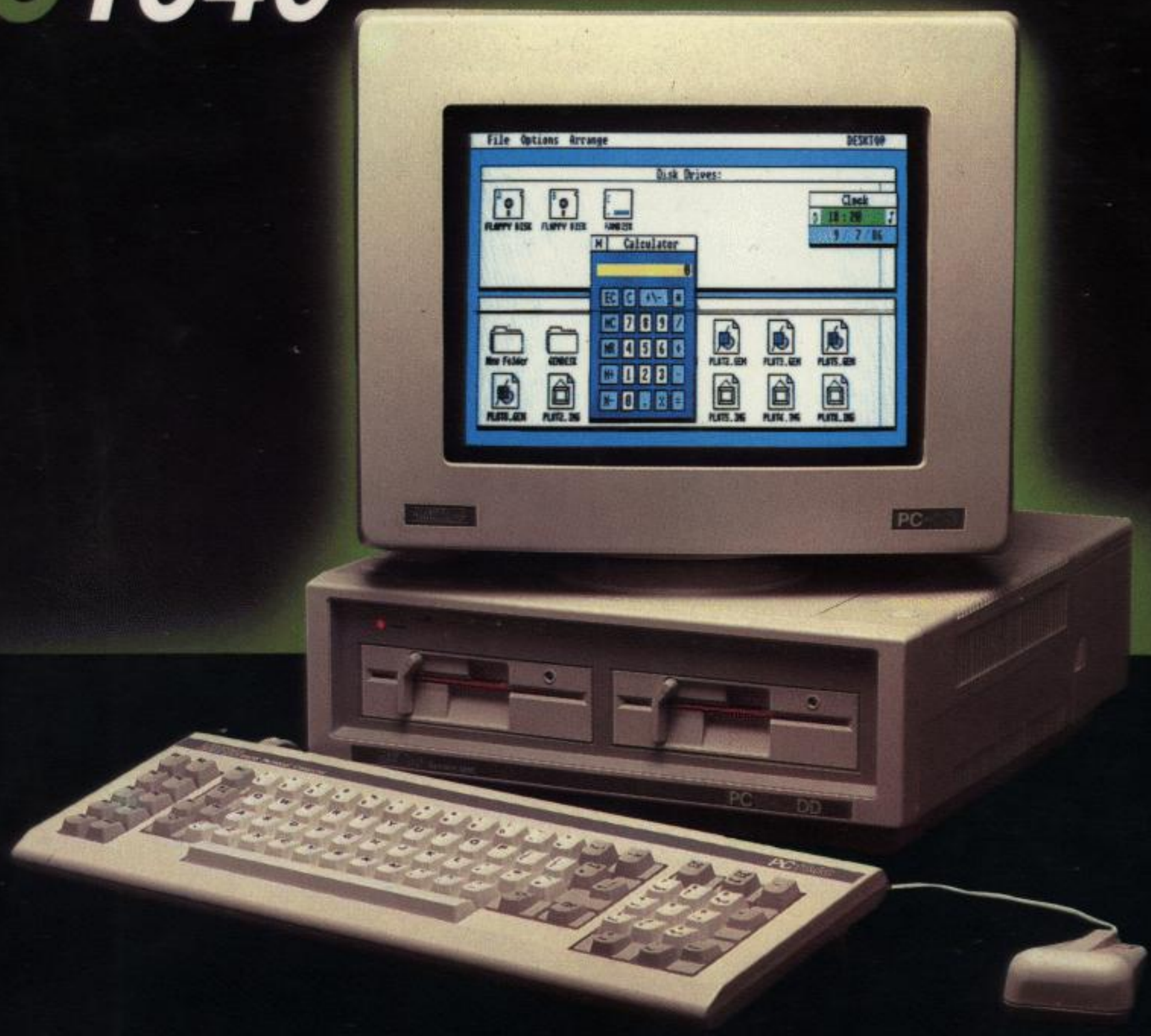
AMSTRAD PERSONAL COMPUTER

PC1640



Technical Reference Manual

AMSTRAD

Preface

This Technical Reference Manual is intended primarily to assist writers of software for the Amstrad PC1640, although in conjunction with the PC1640 Service Manual it will be of interest to designers of add-on hardware.

It is assumed that the reader has a working knowledge of the Industry Standard architecture comprising of an 8086 (or 8088) with DMA, PIT, RTC and Interrupt Controller support chips; plus Extended Graphics Adapter (or Color Graphics Adapter) with Floppy Disk, Serial and Parallel Adapters.

The information contained herein is largely unique to this document, with the exception of parts of the appendices which expand on the information contained in the PC1640 User Instructions and the Microsoft MSDOS Reference Manual.

Whilst the PC1640 implements a superset of the Industry Standard, this manual makes no attempt to identify those areas of the PC1640 specification which exceed the Industry Standard. Users should, therefore, exercise caution when writing software for a range of manufacturers' PCs and only use the "Lowest Common Denominator" facilities if simple portability is required.

All information of a technical nature and particulars of the product are given by Amstrad in good faith. However, it is acknowledged that there may be errors or omissions in this manual.

All correspondence should be addressed to:

Amstrad Consumer Electronics Plc
Brentwood House
169 Kings Road
Brentwood
ESSEX CM14 4EF

All maintenance and service on the product must be carried out by Amstrad authorised dealers. Amstrad cannot accept any liability whatsoever for any loss or damage caused by service or maintenance by unauthorised personnel. This manual is intended to assist the reader in the use of the product, and therefore Amstrad shall not be liable for any damage or loss whatsoever arising from the use of any information or particulars in, or any error or omission in, this manual or any incorrect use of the product.

Written by Bill Weidenauer, AMSTRAD plc.

Published by Amstrad.

First Published 1987.

Cover Index Contents

Table of Contents:

Section 1 - Hardware

AMSTRAD PC1640 TECHNICAL MANUAL

1.0 Introduction

This manual provides a comprehensive description of the AMSTRAD PC1640 hardware and firmware. General information about the PC1640, GEM Desktop and the delivered operating system software is contained in the AMSTRAD PC1640 USER GUIDE. This manual is intended to satisfy the needs of advanced developers who must have access to the various resources available within the PC1640.

Note that all address constants in this document are hexadecimal. In addition hexadecimal quantities are noted with small letter 'h' terminator to denote that they are in hexadecimal form. Address quantities are not usually annotated this way since they are clearly hexadecimal. Values are presented in hexadecimal form when they are logically bit oriented quantities rather than of purely numerical significance.

1.1 Central Processing Unit (CPU)

The CPU is an 8086-2 microprocessor with 1 Megabyte memory addressing capability (See Figure 1), running at a clock frequency of 8MHz. The CPU is connected to an on-board 16-bit system memory bus requiring four 125nS timing cycles (T-States) per access resulting in a 500nS memory cycle for 16-bit memory. The CPU is also connected on an on-board 8 bit I/O and memory peripheral bus with a 4 MHz clock, which in turn connects to an external expansion bus. Operations on the 8-bit bus automatically incur 125nS wait states as follows:

Operation	Wait States	Bus Cycle
8-bit (Memory)	4	1.0 uS
16 to 8-bit convert (Memory)	12	2.0 uS
8-bit (I/O)	6	1.25 uS
16 to 8-bit convert (I/O)	16	2.5 uS

The CPU is configured to run in maximum mode and the instruction set may be optionally extended by the addition of an 8087-2 Numeric Data Coprocessor. The 8087 BUSY output is connected directly to the 8086 NOT TEST input.

1.2 MEMORY LAYOUT

The main board memory consists of 640K bytes of system RAM with parity checking and 16K bytes of system ROM without parity checking.

The 640K byte user RAM starts at CPU memory address 00000 and extends to 9FFFF.

The 128K byte address space from A0000 to BFFFF is reserved for video regeneration buffers, and is not used by CPU programs. The PC1640 Internal Graphics Adapter (IGA) uses the full 128K byte range or segments of this memory range depending on the display mode (see section 2, Video I/O Int 16). External display adapters also use this memory address range for their display buffers.

00000			
	9FFFF	ON-BOARD DYNAMIC RAM	640K BYTE SYSTEM MEMORY RANGE
	A0000 BFFFF	128K BYTES VIDEO DISPLAY BUFFERS	
1M BYTE ADDR RANGE	C0000	192K BYTES EXPANSION ROMS	
		C8000 - C9FFF : HD ROM	
	EFFFF	C0000 - C7FFF : IGA ROM	
	F0000 FBFFF	48K BYTES ROS ROM BLOCK REPEATS	64K BYTE SYSTEM ROM AREA
	FC000	16K BYTES (ROS)	

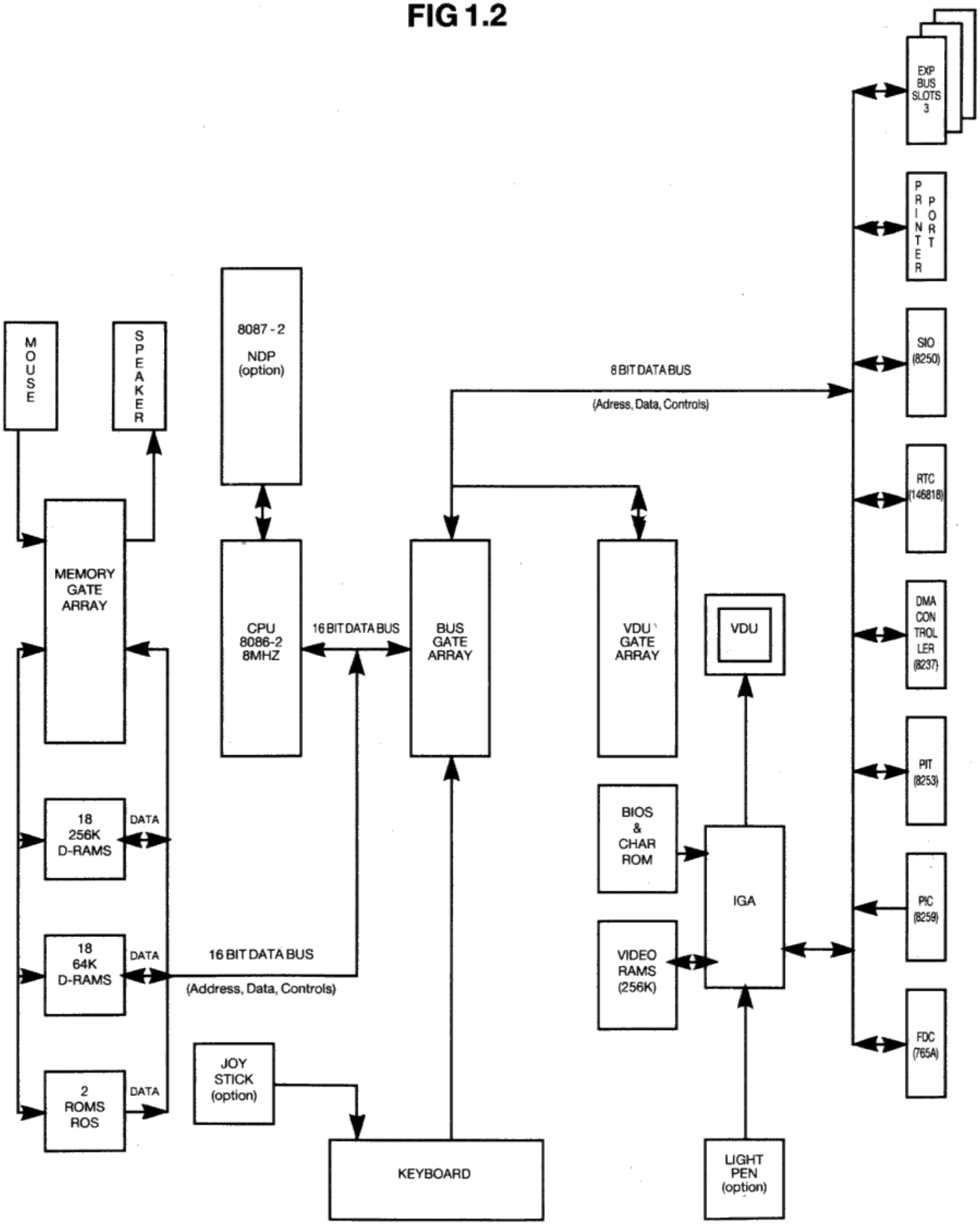
RESIDENT OPERATING SYSTEM ROM

FFFFF

The 192K byte address space from C0000 to EFFFF is reserved for external expansion ROM address space. When the Internal Graphics Adapter is enabled, it uses the space from C0000 to C1FFF for its bios (executable) code and the area from C2000 to C7FFF for its fonts. The optional Hard Disk controller uses the range from C8000 to C9FFF. Additional hard disk controllers may also use the area from CA000 to CD000. The PC1640 test board uses the ROM area from E0000 to E7FFF.

The 16K byte system ROM is at FC000 to FFFFF and contains the Resident Operating System (ROS) firmware. The 48K byte address range from F0000 to FBFFF is reserved for ROM space expansion. The 16K byte ROS area address bits are partially decoded such that the ROS ROM repeats four times in the F0000 to FFFFF address range.

FIG 1.2



1.3 MAIN BOARD I/O CHANNELS

The interfaces on the main board occupy the 8086 I/O addresses as follows:

ADDRESS(hex)	OUTPUT USE	INPUT USE
000 - 00F	8237 DMA Controller	8237 DMA Controller
010 - 01F	Do Not Use	Do Not Use
020 - 021	8259 Interrupt control	8259 Interrupt control
022 - 03F	Do Not Use	Do Not Use
040 - 042	8253 PIT Load Count (0-2)	8253 PIT Read Count (0-2)
043	8253 PIT Load Mode	Undefined
044 - 05F	Do Not Use	Do Not Use
060	No Effect	Port A - Keyboard Code or System Status 1
061	Port B - System Control	Port B - (Readback)
062	No Effect	Port C - System Status-2
063	No Effect	Do Not Use
064	Write System Status-1	Do Not Use
065	Write System Status-2	Do Not Use
066	System Reset	Do Not Use
067 - 06F	Do Not Use	Do Not Use
070	146818 RTC Address	Do Not Use
071	146818 RTC Data	146818 RTC Data
072 - 077	Do Not Use	Do Not Use
078	Clear Mouse X-Coordinate	Mouse X-Coordinate
079	Do Not Use	Do Not Use
07A	Clear Mouse X-Coordinate	Mouse X-Coordinate
07B - 07F	Do Not Use	Do Not Use
080	Do Not Use	Do Not Use
081	DMA Page Register Ch 2	Do Not Use
082	DMA Page Register Ch 3	Do Not Use
083	DMA Page Register Ch 0,1	Do Not Use
084 - 09F	Do Not Use	Do Not Use
0A0	NMI Mask Control	Do Not Use
0A1 - 0BF	Do Not Use	Do Not Use
0C0 - 0FF	Reserved	Reserved
378	Printer Data Latch	Printer Data Latch
379	Do Not Use	Printer Status
37A	Printer Control Latch	Printer Control Latch
37B - 37F	Do Not Use	Do Not Use
3B0 - 3BF	Mono Mode CRTC Registers	Mono Mode CRTC Registers
3C0 - 3CF	Video Controller Registers	Video Controller Registers
3D0 - 3DF	Color Mode CRTC Registers	Color Mode CRTC Registers
3F0 - 3F1	Do Not Use	Do Not Use
3F2	Drive Selection	Do Not Use
3F3	Do Not Use	Do Not Use
3F4	Do Not Use	765 FDC Status
3F5	765 FDC Data	765 FDC Data
3F6 - 3F7	Do Not Use	Do Not Use
3F8 - 3FF	8250 UART Tx Data/Control	8250 UART Rx Data/Control

1.4 EXPANSION BUS I/O CHANNELS

The 8086 CPU I/O addresses on the expansion bus are as follows:

ADDRESS(hex)	USE
200 - 20F	External Game Control Interface
210 - 217	External Bus Expansion Unit
220 - 24F	Reserved
278 - 27F	External Printer Port
2F0 - 2F7	Reserved
2F8 - 2FF	External Asynchronous Serial RS232C Port

ADDRESS(hex)	USE
300 - 31F	External Prototyping Card
320 - 32F	External Hard Disk Controller
380 - 38F	External SDLC Serial RS232C Port
3A0 - 3AF	Reserved
3B0 - 3BB	External Monochrome VDU Controller
3BC - 3BF	Printer Port
3C0 - 3CF	External Graphics Controller
3D0 - 3DF	External Color/Graphics Controller

I/O address above 03FFh, if accessed, wrap around and are mapped onto the range 0000h-03FFh.

External cluster controllers at 0790h-0793h, 0B90h-0B93h, 1390h-1393h and 2390h-2393h wrap around to I/O addresses 0390h-0393h respectively.

1.5 Direct Memory Access (DMA)

The Amstrad PC 1640 supports four DMA channels on the system board, using an 8237-4 DMA controller and programmable page registers to extend its addressing range from 64k bytes to the full 1M byte processor address range. Each channel is able to transfer data in blocks of up to a maximum of 64K bytes within a page. The DMA channels are for 8-bit data transfers between (8-bit) I/O devices and 8-bit or 16-bit memory.

In peripheral (slave) mode, CPU I/O address lines A0 - A3 are connected conventionally so that 16 command codes appear in the order described in the 8237 data sheets (See section 3.5).

The DMA controller CLK is driven at 4MHz (+/- 0.1%). In master mode during DMA transfers on channels 1,2 and 3, one wait state is added resulting in a five-clock DMA bus cycle of 1.25uS. Channel 0 transfers have a four-clock bus cycle of 1uS.

The DMA channel request signals are as follows:

DMA Channel	USE
0	8253 Timer/Counter OUT1 output - for memory refresh.
1	Spare for use by expansion bus. Used by external SDLC Serial Port.
2	765 Floppy Disk Controller DRQ output. Available on the expansion bus.
3	Spare for use by expansion bus. Used by external Hard Disk Controller.

1.5.1 DMA Page Registers

DMA channels 1, 2 and 3 can address the entire 1M byte addressing range of the CPU through the use of their associated DMA page registers. There are three DMA registers, one each for channels 1 through 3. Each page register defines for its channel which one of sixteen 64K byte pages in the 1M byte address range DMA transfers are to occur. The page registers are static so that modulo 64K byte addressing occurs at page boundaries.

The DMA page register bit assignments are as follows:

Bit	Output Use
7-4	Not Connected
3	Address bit A19
2	Address bit A18
1	Address bit A17
0	Address bit A16

1.5.2 DMA Initialisation

Following a reset, system (ROS) initialisation firmware (in the ROS) sets up the 8237 DMA controller for channel 0 (dynamic refresh) operation as follows:

Function	Initialised State
Word Count	64K Transfers
Mode Register	Read Autoinitialise

Function	Initialised State
Command Register	Increment Single Mode Disable Memory to Memory Enable Controller Normal Timing Fixed Priority Late Write DREQ Active High DACK Active Low
Mask Register	Clear Channel 0 Mask Bit

After power-up or system reset the DMA page registers are undefined and are initialised to zero by the ROS firmware and all 8237 internal locations for channels 1-3 are initialised to a state comparable to the channel zero initialisation above.

Following industry compatibility, memory to memory DMA is not supported on the PC1640. It is prohibited due to timing considerations.

1.6 System Interrupts

Nine levels of hardware interrupt are provided for in the system by the CPU Non Maskable Interrupt (NMI) and by an 8259A-2 Interrupt Controller. All levels including NMI, are maskable under software control.

CPU I/O address line A0 is connected conventionally so that the command codes appear in the order described in the 8259 data sheets. The SP/EN pin is tied high signifying that the device is to be hardware un-buffered and designated as a master, not a slave.

1.6.1 Interrupt Levels

The interrupt levels are assigned as follows:

Level	Assigned Function
NMI	Memory Parity Error and 8087 NDP INT output.
0	8253 Timer/Counter Out0 output.
1	Keyboard Scan Code Receiver.
2	146818 Real Time Clock IRQ output. Available on the expansion bus.
3	Spare for use by expansion bus. Used by external (secondary) Asynchronous Serial Port and external SDLC Serial Port.
4	8250 UART INTRPT output. Available on the expansion bus. Used by external SDLC Serial Port.
5	Hard Disk Controller. Available on the expansion bus.
6	765 Floppy Disk Controller INT output. Available on the expansion bus.
7	Parallel Printer Port. Available on the expansion bus. Used by external Printer Port (secondary) and Printer Port (ternary) on external Monochrome VDU Controller.

The system (ROS) firmware initialises the 8259 address bits such that IRQ0 through IRQ7 appear in the CPU interrupt vector space at interrupts 8 through 15 repectively. NMI appears at CPU interrupt vector 2.

1.6.2 Interrupt Controller Initialisation

Following a reset, the initialisation firmware in the ROS sets the 8259 Interrupt Controller to operate as follows:

8086 system, Single (not cascaded),
Normal fully nested (not special),
Edge-triggered,
Buffered mode - slave,
Normal EOI (not auto),
Fixed priority - level 0 highest, level 7 lowest.

The system (ROS) firmware initialises the 8259 address bits such that IRQ0 through IRQ7 appear in the CPU interrupt vector space at interrupts 8 through 15 respectrively. NMI is configured to CPU interrupt vector 2.

1.6.3 NMI Mask Control

The NMI Mask Control is a write only register at I/O address 0A0h and allows the CPU non-maskable interrupt (NMI) input to be enabled or disabled by software. The Bit assignments are as follows:

Bit	Output Use
7	Enable NMI.
6 - 0	Not Connected

Following a reset NMI is disabled.

NMI can be connected to the 8087 NDP, the on-board memory parity check circuit, and the expansion bus I/OCHCK (I/O Channel Check).

1.7 Programmable Interval Timers

Three programmable timer/counters are provided at I/O Addresses 040 - 043 by an 8253 Programmable Interval Timer (PIT) device. They are defined as follows:

Counter	Use
0	General Purpose Timer.
1	Used by DMA channel 0 (for dynamic ram refresh).
2	Tone Generation for Speaker.

1.7.1 Timer Configuration

The 8253 timers are configured as follows:

Function	Configuration
CLK 0,1,2	1.193 MHz +/- 0.1% (54.925493 ms per count)
GATE 0,1	Always 'ON'.
GATE 2	Controlled via Port B (System Control Channel) Speaker Modulate output.
OUT 0	Interrupts on 8259 PIC IR0 input.
OUT 1	Requests on 8237 DMA DREQ0 input.
OUT 2	Logical 'AND' with Port B (System Control Channel) Speaker Drive output. Also goes to Port C (System Status-2 Channel) as an input.

1.7.2 Counter 1 initialisation

Following a reset, the system initialisation firmware in the ROS programs the 8253 PIT for counter 1 (dynamic ram refresh) operation as a rate generator producing a signal with a period of 15.13 uS. There are no restrictions requiring the initialisation and programming of counters 0 and 2.

1.8 System Status and Control

Two system status input channels and four output channels are provided on-board. Ports A, B and C emulate a pre-programmed 8255 PPI device. They are located in the I/O address space in the range 060h - 06Fh. Port B is programmed for control output, Port A is programmed either for Status-1 input or for receiving data from the keyboard, and Port C is programmed for Status-2 input.

Ports A, B and C emulate an 8255 PPI that has been set up as follows:

Group A Mode 0, Group B Mode 0,
Port A = input, Port B = output,
Port C(U) = input. Port C(L) = input.

Unlike an 8255, power-up and reset do not affect this configuration.

1.8.1 Port B - System Control

The System Control channel is located at I/O Address 061h. Its bit assignment is as follows:

Bit (PBn)	Output Use
7	Enable Status-1/Disable Keyboard Code on Port A.
6	Enable incoming Keyboard Clock.
5	Prevent external parity errors from causing NMI. (Also Disable any pending NMI).
4	Disable parity checking of on-board system Ram.
3	Undefined (Not Connected).
2	Enable Port C LSB / Disable MSB. (See 1.8.3)
1	Speaker Drive.
0	8253 GATE 2 (Speaker Modulate).

When bit 7 is set high, Status-1 data is enabled on Port A, the keyboard data path and keyboard interrupts are disabled. When bit 7 is set low, keyboard input data is enabled on port A, the keyboard data path and keyboard interrupts are enabled. Applications software which sets PB7 must restore it to the cleared state else the keyboard may be rendered inoperable.

The keyboard interface operates as follows: Each incoming keycode is latched on-board, causing a keyboard interrupt (on level 1). While the interrupt remains pending, the incoming keyboard data signal is forced low as an acknowledgement to the keyboard that the keycode has been received. As soon as the interrupt has been cleared, the keyboard may use the Data signal to transmit the next keycode.

PB5 when set prevents an external parity error (ie. an I/OCHCK condition on the expansion bus) from causing NMI, even if NMI is unmasked. When NMI has been triggered and latched it may be cleared by pulsing PB5 (if the external device has removed its I/OCHCK signal).

PB2 when set enables the reading of the four LS bits of the RAM fitted indicator on Port C. When PB2 is clear the top (MS) bit of the RAM fitted indicator is read (see 1.8.3).

PB1 may be toggled to drive the speaker with a corresponding pulse train. The speaker may also be driven by a wave form from the 8253 PIT OUT2 output (simultaneously with PB1).

PB0 may be toggled to drive the 8253 gate input, hence modulate counter 2operations and therefore driving the speaker which may all be performed simultaneously to create various audio effects.

1.8.2 Port A - Status-1 Input/Keyboard Code

Port A is a read only location located at I/O Address 060h. The bit assignments for Port A are as follows:

Bit (PAn)	Status-1	Keyboard Input
7	Always 0.	KBD7
6	Second Floppy disk drive installed.	KBD6
5	DDM1 - Default Display Mode bit 1.	KBD5
4	DDM0 - Default Display Mode bit 0.	KBD4
3	Always 1.	KBD3
2	Always 1.	KBD2
1	8087 NDP installed.	KBD1
0	Always 1.	KBD0

When Port B, Bit 7 (PB7) is set to high, reading Port A loads Status-1. When PB7 is set low, reading Port A loads keyboard data.

The Default Display Mode bits (DDM1, DDM0) are set up by the ROS during system initialisation as follows:

DDM1	DDM0	Default Display Mode Selected
0	0	Internal Graphics Adapter (IGA) Enabled or Extended Adapter Installed in Expansion Bus.
0	1	Colour Graphics Adapter Installed in the Expansion Bus with alpha, 40 X 25 chars, bright white on black.
1	0	Colour Graphics Adapter Installed in the Expansion Bus with alpha, 80 X 25 chars, bright white on black.
1	1	External Monochrome Controller, 80 X 25 chars.

When the Internal Graphics Adapter (IGA) is enabled then switches 1 - 5 determine what the default display mode is and whether the IGA is the primary or the secondary adapter. See section 1.22 for these details.

When the Internal Graphics Adapter (IGA) is disabled then the ROS uses the position of display selector switches SW6, SW7 and SW10 to determine the default display mode.

When an EGA or another adapter is installed then its particular User's/Hardware manual is the best guide to its display modes or other features.

Following a reset, the ROS then sets the initial video state is based on the DDM value. Section 2.3.7 gives additional details of the ROS Video Mode settings.

1.8.3 Port C - Status-2 Input

Port C is a read only location located at I/O Address 062h. Its bit assignments are as follows:

Bit (PCn)	Input Use	
7	On-board system RAM parity error.	
6	External parity error (I/OCHCK from expansion bus).	
5	8253 PIT OUT2 output.	
4	Undefined.	
	LSB or MSB (depends on PB2)	
3	RAM3	Undefined
2	RAM2	Undefined
1	RAM1	Undefined
0	RAM0	RAM4

PC7 is forced to the zero state when on-board system RAM parity checking is disabled by PB4.

When the I/OCHCK condition (external parity error) from the expansion bus is disabled from causing NMI (by PB5 set high), PC6 reflects the state of the I/OCHCK input else it reflects the latched state of I/OCHCK.

The value of RAM4-RAM0 denotes the amount of system RAM fitted to the system as follows:

RAM4	RAM3	RAM2	RAM1	RAM0	
0	1	1	1	0	512K bytes.
0	1	1	1	1	544K bytes.
1	0	0	0	0	576K bytes.
1	0	0	0	1	608K bytes.
1	0	0	1	0	640K bytes.

Since the PC1640 comes with 640K of RAM fitted the expected value should always be 640K bytes. Any other value indicates an installed memory configuration error.

See section 1.8.1 for the Control Port B setting for reading RAM fitted segment bits.

1.8.4 Write System Status-1

The Write System Status-1 register (WSS1) is a write only register at I/O Address 064h and is initialised by the Resident Operating System (ROS) firmware based on values obtained from configuration switches 4 and 5. It is used in conjunction with the 8255 PPI Port A emulation. The bit assignments are as follows:

Bit	Output Use
7	No effect.
6	PA6 - Second Floppy disk drive installed.
5	PA5 - DDM1.
4	PA4 - DDM0.
3	No effect.
2	No effect.
1	PA1 - 8087 NDP installed.
0	No effect.

1.8.5 Write System Status-2

The Write System Status-2 register is a write only register at I/O Address 065h and is initialised by the Resident Operating System (ROS) firmware based on the memory size check perfomed during Power On Self Test. It is used in conjunction with the 8255 PPI Port C emulation. The bit assignments are as follows:

Bit	Output Use
7	PC2 (MSB) - Undefined.
6	PC1 (MSB) - Undefined.
5	PC0 (MSB) - Undefined.
4	PC3 (MSB) - RAM4.
3	PC3 (LSB) - RAM3.
2	PC2 (LSB) - RAM2.
1	PC1 (LSB) - RAM1.
0	PC0 (LSB) - RAM0.

Since the PC1640 comes with 640K on the main board the expected value of PC3 - PC0 is binary 10010.

1.8.6 System Reset

Any write access to I/O Address 066h regardless of the value written will cause the hardware to generate an immediate 512uS system reset and pulse the reset line on the expansion bus. The contents of the on-board system RAM is preserved following a system reset.

1.9 Real Time Clock

A HD146818 Real Time Clock plus RAM device is installed and backed up by a set of four non-rechargable size AA batteries. The clock device provides a time of day clock with alarm, a one hundred year calendar, a programmable periodic interrupt, and 50 bytes of static RAM. The static RAM is called the Non-Volatile RAM (NVR) and used to store system configuration data such as number of disk drives, memory size, serial I/O parameters, and default VDU screen mode. The ROS firmware maintains a checksum of the NVR and will reset the configuration data to 1sensible values1 during startup whenever the checksum value is incorrect (thus destroying your actual configuration). Even though direct hardware access to the NVR is possible it is recommended that the programs make use of the ROS Enhanced Function Interrupt (Interrupt 21) to access the NVR because these properly maintain the NVR checksum value.

When system power is off and the 146818 is on battery backup power, the functions which remain active are the clock and the retention of RAM data. No battery power is used while the system power is on.

The input crystal oscillator runs at 32.768 KHz and the 146818 interrupt request is connected to the 8259 system interrupt controller on level 2 (which is also available on the expansion bus). The 146818 power-sense input PS is connected to a battery condition sensor. When the backup battery voltage is sufficiently low, the VRT bit in register D becomes set indicating that the time, the calendar and the NVR data are no longer valid. When this condition is noted during startup, the firmware outputs the message "Please fit new batteries" and resets the NVR to default values (See section 2.4).

All the features described in the 146818 data sheet are available with the exceptions that the CKOUT (clock output) and SQW (square wave output) pins are not connected on the main board.

Writing or reading the NVR involves a two step sequence for each byte that is accessed. The RTC Address channel (I/O Address 070) is first loaded with the NVR location to be accessed. Then the RTC Data channel (I/O Address 071) is either written or read to complete the I/O operation. This facility should be used with caution in order to avoid disturbing the system configuration data.

1.10 Parallel Printer Port

The printer port provides an interface for driving 8-bit and 7-bit Centronics compatible printers. The timing of the signals to the printer is under direct software control. There is a read/write control latch for sending control signals to the printer, an unlatched read-only printer status channel, and a read/write data latch for sending printer data.

In addition the printer control latch can be read to obtain system type and switch information.

1.10.1 Printer Data Latch

The printer data latch is a read/write record at I/O address 378 and its layout is as follows:

Bit (Dn)	Output/Input Use	Cable Polarity
7	Data 7	Hi
6	Data 6	Hi
5	Data 5	Hi
4	Data 4	Hi
3	Data 3	Hi
2	Data 2	Hi
1	Data 1	Hi
0	Data 0	Hi

The contents of the data latch are undefined following a power-up or system reset.

1.10.2 Printer Control Latch

The printer control latch is a read/write record at I/O address 37A and its layout is as follows:

Bit	Output/Input Use	Reset State	Cable Polarity
7	PC1640 SW7 [RO]		
6	PC1640 SW6 [RO]		
5	OPT (1640)/1 (1512) [RO]		
4	Enable Int on ACK	False	
3	Select Printer	False	Low
2	Not Reset Printer	True	Low
1	Select Auto Feed	False	Low
0	Data Strobe	False	Low

Bit D7 is a read-only bit which reflects the state of switch 7 (at the back of the machine) and returns a logic "1" when the switch is in the "on" position and a logic "0" if in the the "off" position.

Bit D6 is a read-only bit which reflects the state of switch 6 (at the back of the machine) and returns a logic "1" when the switch is in the "on" position and a logic "0" if in the the "off" position.

Bit D5 is the option (OPT) bit and can return one of three different pieces of information. Although not documented as such on the PC1512, Bit D5 was always a "1", however on the PC1640 it will always be a zero if immediately prior to the read of channel 037Ah the software performs an I/O read of an I/O channel implemented on the PC1512 main board, having address line A7 high (for example, the CGA channels). This is a simple test for software to detect whether it is running on a PC1512 or a PC1640. A PC1512 will give a 1, whereas a PC1640 will give a 0.

In addition to being a test of machine type the OPT bit, D5, can also reflect the state of either SW9 or SW10. The OPT bit will reflect the state of switch SW9 by an I/O read operation to an I/O channel not implemented on the main board and having address lines A14 and A7 both low (for example channel 0278h) immediately prior to the reading of channel 037Ah. The OPT bit is set to the state of switch SW10 by an I/O read operation to an I/O channel not implemented on the main board having address lines A14 high and A7 low (for example channel 4278h). Software testing OPT bit should disable interrupts before the initial (dummy) channel read and the I/O read of channel 037A in order to avoid additional (interrupt based) I/O operations between the setting and the testing of the information read back in the OPT bit. For switches SW9 and SW10, a logic "1" is returned when the switch is on the "on" position and a logic "0" if the switch is in the "off" position.

When Interrupt on ACK is enabled an incoming Printer Acknowledge condition will cause a system interrupt on level 7 (which is also available on the expansion bus).

If the printer control lines normally driven via latched bits D0 - D3 are driven externally, the data read on input to this channel will be the logical OR of the latched bits and the externally driven bits, e.g. If a data bit is false and the corresponding cable bit is driven true by the external driver, the bit input will be true.

Following power-up or system reset, the control latch contents assume reset conditions as shown.

Note that this is a general purpose printer interface and that not all printers require all the control signals, hence the provision for non-standard printers to be able to drive some of the control signals as inputs to the main board. The timing requirements on Centronics compatible printers generally specify that data must be present at 1uS before the strobe is made active, and must remain valid for at least 1uS after strobe goes inactive. The strobe duration must be between 1uS and 500 uS. Printer Busy status can be inspected as soon as the strobe is inactive in order to determine when more data can be sent.

1.10.3 Printer Status Channel

The Printer Status Channel is a read only register at I/O Address 0379h. Its layout is as follows:

Bit	Input Use	Cable Polarity
7	-Printer Busy	High
6	-Printer Acknowledge	Low
5	Paper Out	High
4	Printer Selected	High
3	-Printer Error	Low
2	-LK3 fitted	
1	-LK2 fitted	
0	-LK1 fitted	

LK1 - LK3 are general purpose factory installed option links on the main board which are used by the system ROM Operating System (ROS) firmware to distinguish national variant machine configurations. The ROS will produce its sign-on message and error messages in one of seven languages. The first seven states (0 - 6) are used for language variants and the eighth (7) state is used extended diagnostic mode testing (See section section 2.2). Since the link state is inverted, the value obtained from the lower three bits of the printer port must be exclusive or'ed (XOR) with 1's to obtain the language number.

LK1	LK2	LK3	ROS Language
OFF	OFF	OFF	English.
OFF	OFF	ON	German.
OFF	ON	OFF	French.
OFF	ON	ON	Spanish.
ON	OFF	OFF	Danish.
ON	OFF	ON	Swedish.
ON	ON	OFF	Italian.
ON	ON	ON	Diagnostic Mode. (English)

Note that this is a general purpose printer interface and that not all printers implement all the status lines, nor do they all attach the same meanings to the error conditions.

Printer Busy normally indicates that a printer cannot receive data, for example during data entry, printing, when offline, or during a printer error condition.

Printer Acknowledge, if implemented is generally asserted by a printer to indicate that data has been received and the printer is ready to receive the next data. Note that Printer Acknowledge (ACK) can also be set to cause interrupts (See 1.10.3).

Section 1.14 contains the printer connector pin assignments.

1.11 The Internal Graphics Adapter.

The Internal Graphics Adapter (IGA) is a gate array on the main board which provides either an Extended Graphics Adapter (EGA) mode, or a Color Graphics Adapter (CGA) mode or a Monochrome Display Adapter (MDA) mode. Additional Hercules monochrome adapter modes and Plantronics (Color) adapter functions are also supported.

The video screen memory (or regeneration buffer) is the 128K byte range from A0000 to BFFFF and its configuration varies depending on the selected display mode. The regeneration buffer origin (ie the starting address) may be configured to either A0000, B0000 or B8000 with sizes varying from 2K bytes to 32K bytes. (See 1.11.2.5).

When an Extended Color (PC-ECD) Display is fitted the IGA is capable of displaying up to 64 different colors in EGA mode with a resolution of 640 dots x 350 lines. The CGA and Plantronics display modes supports 16 colors (for Extended Color Displays and standard Color Displays) with a resolution of up to 640 dots by 200 lines.

For the PC-MD monochrome display, the IGA supports black and white text in normal, intense, blink, or underline with a resolution of 720 by 350 in text modes. The maximum monochrome Graphics resolution is 640 x 350 or 720 x 348 resolution graphics for Hercules compatible monochrome.

For color systems, when an initial mode change is set up via the ROM BIOS, a set of sixteen color palette registers are loaded with standard values such that a standard IRGB color selection is available for sixteen total colors available on the display. The color attribute fields (discussed later) and the color plane selections relate to the standard sixteen color palette format as follows:

Intensity	Red	Green	Blue	Colour

Intensity	Red	Green	Blue	Colour
0	0	0	0	Black
0	0	0	1	Blue
0	0	1	0	Green
0	0	1	1	Cyan
0	1	0	0	Red
0	1	0	1	Magenta
0	1	1	0	Brown
0	1	1	1	White
1	0	0	0	Grey
1	0	0	1	Light Blue
1	0	1	0	Light Green
1	0	1	1	Light Cyan
1	1	0	0	Light Red
1	1	0	1	Light Magenta
1	1	1	0	Yellow
1	1	1	1	Intense White

Note that the Extended color display (PC-ECD) supports 64 colors using a 6-wire rgbRGB color scheme (where the capital letters represent 2/3 intensity primary color signals and the small letters represent 1/3 intensity secondary color signals). Since each of the sixteen palette registers is a 6-bit register with the bits representing rgbRGB (MSB through LSB respectively) only 16 of 64 colors can be displayed on the screen at any one time.

1.11.1.1 Color Alpha Display

In Color mode, two Alpha modes are available: either (medium resolution) 40 characters by 25 rows or (high resolution) 80 characters by 25 rows. The display RAM requirement is 2K bytes and 4K bytes of display RAM for 40 and 80 column modes respectively. The display regeneration buffer is from B8000h to BBFFFh for these modes and the display ROM bios supports up to 8 or 4 separate display pages for 40 and 80 column modes respectively.

The character set is formed by a RAM loadable character generator and for 200 line resolution systems, each of the 256 characters is made up of a 8 by 8 pixel matrix.

When an Extended Color Display (PC-ECD) is being used in full 350 line resolution, the 40 and 80 column text modes are still supported but the character generator is programmed differently and a 8 by 14 pixel matrix is used. The display RAM mapping and regen buffer origin and size requirements are exactly the same as the 200 line display modes.

The starting address in the display RAM is programmed via the CRT (Cathode Ray Tube) Controller (CRTC). The starting address is on an even address boundary and it addresses the first (leftmost) character position in the top row of the display. The CRTC starting address register is a 16-bit register and it specifies the offset in two byte pairs from the display mode origin. This means for each change of one in the CRTC starting address register, the next even address is selected in the display RAM as the current regeneration buffer origin.

In order to display a single character, two bytes of display RAM are required, and for each pair of display RAM bytes, the even address is for the character code and the odd address is for the attribute byte. Subsequent characters are displayed along the row from left to right. When the end of a row is reached the next pair in the display RAM appears in the first character position of the next row down. Appendix 8 gives the 256 character codes and their respective default character representations. Note that the ROM bios for EGA mode supports reprogramming the character generator with user supplied character matrices.

The attribute byte allows a choice of either 16 foreground and 8 background colors per character, plus blinking, or a choice of 16 colors for both foreground and background without blinking. In CGA mode, the display border may be programmed for any one of the 16 colors.

The attribute byte for each is as follows:

Bit (ATn)	Definition
7	Intensity or Enable Blink (Background)
6	Red (Background)
5	Green (Background)
4	Blue (Background)
3	Intensity (Foreground) or Character Map A/B Select
2	Red (Foreground)

Bit (ATn)	Definition
1	Green (Foreground)
0	Blue (Foreground)

Bit 7, the Intensity or Enable Blink Bit, changes function based the Mode Control Register. In EGA mode, Mode Control Register (I/O address 03C0h, index 10), Bit 3 selects between Intensity or Blink in standard alphanumeric display modes. For CGA compatible mode, Mode control Register (I/O address 03D8h) bit 5 selects between Intensity or Blink.

In EGA mode bit 3, the foreground intensity bit, can be used as an alternate character map select bit to obtain up to 512 displayable characters for a given screen. The extended character map option is explained in section 1.11.3, IGA Extended Graphics Mode Registers, in the section which deals with the Character Map Select Register which is contained in the sequencer logic.

Note that in EGA mode, the bit positions listing IRGB actually selects one of the 16 paletter registers and depending on the contents of the selected palette register the (extended six-wire) color signals are produced accordingly. If the palette registers are left unchanged from the initial values loaded at mode change the color selections will be given the sixteen standard colors listed in the color table at the beginning of this section. The table below gives the palette register settings which produce the standard 16 color palette.

Reg	r g b	R G B	Color
0	0 0 0	0 0 0	Black
1	0 0 0	0 0 1	Blue
2	0 0 0	0 1 0	Green
3	0 0 0	0 1 1	Cyan
4	0 0 0	1 0 0	Red
5	0 0 0	1 0 1	Magenta
6	0 0 0	1 0 0	Brown
7	0 0 0	1 1 1	White (Grey1)
8	1 1 1	0 1 0	Grey (Grey2)
9	1 1 1	0 0 1	Light Blue
A	1 1 1	0 0 0	Light Green
B	1 1 1	0 1 1	Light Cyan
C	1 1 1	1 1 0	Light Red
D	1 1 1	1 0 1	Light Magenta
E	1 1 1	1 1 0	(Light) Yellow
F	1 1 1	1 1 1	Intense White

The small (lower case) letters represent the secondary colors at 1/3 intensity each and the big (upper case) letters represent the primary colors at 2/3 intensity each.

1.11.1.2 Monochrome Alpha Display

When driving a Monochrome (PC-MD) Display, there is only one Alpha mode available, 80 characters by 25 rows with a resolution of 720 dots by 350 lines. The display RAM requirement is in 4K increments in display regeneration buffer area from B0000h to B7FFFh. The display ROM bios supports up to 8 separate display pages. Buffer 0 is from B0000 to B0F9F, buffer 1 is from B1000 to B1F9F with the remaining 6 buffers starting on even 1000h boundaries to B7000.

In EGA mode the character set is formed by a RAM loadable character generator and each of the 256 characters is made up of a 9 by 14 pixel matrix.

The CRTC Starting Address is programmed in the same way as when in color modes and the two byte character and attribute pairs are arranged in the display RAM just as in the color modes. The attribute byte, however assumes different functions from the color attributes since there are no IRGB signals sent to the monitor, but Video & Intensity are produced. The monochrome attribute byte is as follows:

Function	Bits	7	6	5	4	3	2	1	0
Blanked	Bkg	I/B	0	0	0	I	0	0	0
Underlined	Bkg	I/B	0	0	0	I	0	0	1
Normal	Bkg	I/B	0	0	0	I	1	1	1
Inverse	Bkg	I/B	1	1	1	I	0	0	0

Bit 7, the Background Intensity/Blink Enable (B/I) Bit, changes function based the Mode control register. In EGA mode, Mode Control Register (I/O address 03C0h, index 10), Bit 3 selects between Intense background when inverted or Blinking. For MDA compatible

mode, Mode control Register (I/O address 03B8h) Bit 5 selects between Intensity or Blink Functions. The Hercules modes follow the same scheme in text mode.

Bit 3 is the foreground intensity bit and controls the intensity when not in inverse video or blanked. It is also necessary to turn the contrast down on the PC-MD display in order to observe the difference in video levels.

1.11.2.1 Colour Graphics Display

PC-CD (Standard Color Display) systems support 200 scan lines with a choice of two graphics resolutions, either 320 pixels per scan line with four colors per pixel or 640 pixels per scan line with a two colors and these are termed 'CGA' compatible modes. Additional sixteen color (EGA) modes are also available in the above two resolutions. In the CGA compatible modes, the regeneration buffer for the 320x200 4-color mode and 640x200 2-color mode starts at B8000 and requires 16K bytes per display page. The IGA ROM BIOS supports either 8 or 4 display buffer pages for the 320 or 640 resolution modes respectively.

For EGA 16 color modes, the regen buffer starts at A0000 and is divided into four 64K byte planes, one each for the Blue, Green, Red and Intensity bits. Each plane may be individually read from or written to by the CPU, and two or more planes may be selected by the CPU for writing simultaneously with the same data. Individual data bits can be either enabled or disabled during CPU writes to the graphics memory. The section on registers (1.11.3 below) will explain the methods available for graphics control.

PC-ECD (Extended Color Display) systems support the 200 line resolution modes described above with an additional 16 or 64 color, 640x350 resolution mode. For the high resolution (350 line) mode the regen buffer starts at A0000 and requires 28K bytes per display page. The IGA ROM BIOS supports 2 display pages for the 350 line mode. The 16 color selection is by way of four color planes as described above. The 16 of 64 possible colors is achieved by reprogramming the palette registers introduced in the color text description.

1.11.2.2 Low Resolution (320 x 200) Graphics

In Low Resolution Graphics Mode, the display memory for one scan line (320 pixels) consists of 80 bytes. Each pixel requires two bits so that four pixels are specified by each byte. The leftmost pixel is contained in the two MS bits of the byte and the two bit pairs for the remaining pixels follow on logically from left to right. The two bit field for each pixel specifies one of four colors and can be in one of three palettes as follows:

Colour	Palette 0	Palette 1	Palette 2
0	Background	Background	Background
1	Green	Cyan	Cyan
2	Red	Magenta	Red
3	Yellow	White	White

The display regeneration buffer for medium resolution graphics modes is mapped a split buffer configuration with the even scan lines (0, 2, 4, ... 198) contained in the graphics memory space from B8000 to B9F3F and the odd scan lines (1, 3, 5, ... 199) in the memory address range from BA000 to BBF3F. The memory map is as follows:

	320 Pixels (2 Bits Per)	
B8000 -	Scan Line 0 (80 Bytes)	- B804F
B8050 -	Scan Line 2	- B809F
B80A0 -	Scan Line 4	- B80EF
	.	
	.	
	.	
B9EF0 -	Scan Line 198	- B9F3F
BA000 -	Scan Line 1	- BA04F
BA050 -	Scan Line 3	- BA09F
BA0A0 -	Scan Line 5	- BA0EF
	.	
	.	
	.	
BBEF0 -	Scan Line 198	- BBF3F

The mapping of a byte of graphics RAM in low resolution mode is as follows:

RAM Bit:	7	6	5	4	3	2	1	0
Pixel:	0		1		2		3	

Pixel Bit:	1	0	1	0	1	0	1	0

1.11.2.3 Medium Resolution (640 x 200) Graphics Mode

In Medium Resolution Graphics Mode, the display memory for for one scan line consists of 80 bytes. Each pixel requires one bits so that eight pixels are specified by each byte. The leftmost pixel is contained in the MS bit of the byte and the remaining pixels follow from left to right. In high resolution mode the two colors are either black (pixel bit off) or pixel bit on with video in one of the 16 colors as selected by a foreground palette register.

One byte of graphics RAM in medium resolution graphics is as follows:

RAM Bit:	7	6	5	4	3	2	1	0
Pixel:	0	1	2	3	4	5	6	7

The address mapping of the scan lines in display RAM for high resolution graphics is the split buffer configuration depicted for medium resolution mode. - All (100) even scan lines from B8000 to B9F3F followed by all (100) odd scan lines from BA000 to BBF3F.

1.11.2.4 High Resolution (640 x 350) Graphics Mode

In High Resolution Graphics Mode, the display memory required for each scan line is 80 bytes which is the same as the 640 pixel scan lines of the medium resolution mode. The major difference is that there are 350 scan lines which are mapped into a contiguous block of display memory starting at A0000 and extending to AFFFF. The IGA ROM BIOS supports two display pages by reprogramming the CRTC starting address register. Page 0 is from A0000 to A6D5F and page 1 is from A8000 to AED5F.

In high resolution mode the display RAM mapping is very straight forward, the 350 display lines are in contiguous 80 byte blocks in the regeneration buffer. The internal pixel to RAM bit mapping is the same as depicted above for medium resolution graphics mode.

This display graphics mapping applies for both EGA Monochrome (BIOS Mode 15) and for EGA High Resolution Color (BIOS mode 16) graphics. The Hercules 720 by 348 graphics mapping is not so straightforward and requires a graphics map which is segmented into four pieces. This mapping scheme is covered in section 1.11.7 with the Hercules Mode control registers.

1.11.2.5 IGA BIOS Modes

The IGA BIOS sets up the hardware to support twelve different modes for the various displays available on the PC1640 range. The following table gives the modes supported by the BIOS ROM. Mode numbers between 8 & 12 are dummy mode numbers for other graphics adapters not supported by the IGA ROM BIOS.

BIOS Mode	0	1	2	3	4	5	6	7	13	14	15	16
Type	Text	Text	Text	Text	Graph	Graph	Graph	Text	Graph	Graph	Graph	Graph
Columns	40	40	80	80	320	320	640	80	320	640	640	640
Rows	25	25	25	25	200	200	200	25	200	200	350	350
Colour(s)	16	16	16	16	4	B/W	2	Mono	16	16	Mono	16/64
Char Cell Size	8x8	8x8	8x8	8x8	8x8	8x8	8x8	9x14	8x8	8x8	9x14	8x14
Regen Origin	B8000	B8000	B8000	B8000	B8000	B8000	B8000	B0000	A0000	A0000	A0000	A0000
Regen Size	32768	32768	32768	32768	32768	32768	32768	32768	65536	65536	65536	65536
Page Size	2048	2048	4096	4096	16384	16384	16384	4096	8192	16384	32768	32768
Number Pages	16	16	8	8	2	2	2	8	8	4	2	2

The Regeneration Buffer Origin is stated in hexadecimal notation since it is an address quantity. All other values are in decimal notation in order to give a numerical perspective to the quantitites.

The IGA ROM BIOS supports multiple display pages and can be called to select an alternate page. The default (base page) upon initial mode selection is always zero and it begins at the origin address. The successive pages are located higher by the page size increment in the table. The equation for page origin is: Page Origin = Regen Origin + (Page Number - Page Size). Since the 'Page Size' quantity is a pure binary multiple it becomes a shift factor for the selected page number. Page are numbered from 0 to n-1 where 'n' is the number of pages available.

The maximum display pages for modes 0 and 1 is listed to be 16 for each. While this is true from a hardware point of view, the IGA ROM BIOS only supports the first eight and the additional pages though accessible via direct access to the memory map, the video character output routines in the ROM BIOS will not correctly access alternate page numbers 8 - 15.

1.11.3 IGA Control Registers

The control registers available in the IGA are somewhat complex but allow a versatile alphanumerics (text) and graphics display environment. The PC1640 ROM Operating System (ROS) and the IGA ROM BIOS allow a simplified set of software interfaces to this hardware and this should be the preferred method of implementing programs with some measure of transportablility to future hardware which may not be compatible with this environment at the hardware level.

Special IGA Registers

There are three special registers which control the overall IGA operational mode and setup characteristics of these modes. The IGA ROM BIOS and the DISPLAY utility program supplied with the PC1640 are the usual programs which sets up these registers. The IGA does not auto-switch, that is, it will stay in a selected hardware emulation mode until some program such as the DISPLAY utility manipulates the control registers. In addition applications programs with drivers for particular hardware will use these registers for their control purposes. The Special IGA Registers are as follows:

Name	Port Address
IGA Extended Mode Control	3DB/3BB
Hercules Control Register	3BF
Plantronics Control Register	3DD

Extended Mode Control Register

The IGA Extended Mode Control Register controls the overall operational mode of the IGA such that it can be in EGA, CGA, MDA, Hercules or Plantronics emulation modes. The Extended mode control register is a write only 8-bit register located at I/O address 3DB/3BB and it can only be written to after two successive I/O reads of address 3D8/3B8. This is a protection feature which prevents accidental modifications from taking place. In order to determine whether to use I/O addresses 3B8/3BB or 3D8/3DB (Mono or Color addresses) it is necessary to know whether the system is configured in monochrome or color mode. The standard method for doing this is to use the system ROS's CRTC I/O address pointer located at RAM address 00463. This word (16-bit) will contain either 3B4 or 3D4 depending whether the system is in monochrome mode or color display mode respectively.

The format of the IGA Extended Mode Control register is as follows:

Bit	Output Function
7	Vsync Polarity, Border Blanking.
6	Enable Special Modes.
5	Disable Palette and Overscan Registers.
4	Lock CRTC Timing Registers.
3	Enable Alternate Character Sets on plane 3.
2	Disable Blanking.
1	Enable 132 Character Mode.
0	Enable Color Simulation Modes.

Bit 7 when set forces negative polarity of vertical sync and also blanks the screen border.

Bit 6 when set enables the 6845 compatible modes, CGA, Monochrome, Hercules and Plantronics modes.

Bit 5 when set locks out the Palette registers and the Overscan register.

Bit 4 when set prevents modification of the CRT Controller registers which determine sync signal timing.

Bit 3 when set provides an additional character set option of four 8Kb character sets from plane 3. (See page 40 for additional details.)

Bit 2 when set disables the screen blanking in the CGA Color Alpha mode.

Bit 1 when set enables the 132 Character Mode.

Bit 0 when set enables color simulation modes.

Hercules Mode Register

The Hercules Mode Register is active when Hercules Monochrome Graphics is enabled and is a write only register at I/O address 3BF. It controls the configuration of the graphics memory map and protects against accidental setting of the graphics mode bits in

the mode control port at 3B8. The layout of this register is as follows:

Bit	Function
7 - 2	Not Used.
1	Enable Mode Reg Bit 7. (The Full/Half Graphics bit)
0	Enable Mode Reg Bit 1. (The Text/Graphics bit)

Since Hercules mode resembles MDA mode but with graphics extensions, applications software built for the MDA may inadvertently set the Hercules graphics control bits. This register prevents such accidents by forcing the respective bits to zero so that text mode is maintained and the display buffer remains at B0000.

Plantronics Mode Register

Plantronics mode is an additional color mode which allows CGA-like color but with extensions for four colors in 640x200 resolution and 16 colors in 320x200 resolution through use of an additional color plane residing in the BC000 memory range. The Plantronics Mode Register (PLR) is a write only register located at I/O address 3DD. Its format is as follows:

Bit	Function
7	Not Used.
6	Color Plane 0/1 Position.
5	Enable Extended color palette 1
4	Enable Extended color palette 2
3 - 0	Not Used.

Bit 6 when clear enables color plane 0 to start at B8000 and color plane 1 to start at BC000. Setting bit 6 swaps the two plane's starting addresses so that color plane 0 starts at BC000 and color plane 1 starts at B8000. Each color plane is 16K bytes in length.

The Extended color palettes enable combinations of bits set is planes 0 and 1 to select one of four different colors for 640 dot resolution or one of 16 colors in 320 dot resolution modes.

When both bits 4 and 5 are cleared standard CGA mode is enabled. When both bits 4 and 5 are set together then bit 4 overrides bit 5 but it is recommended that either bit 4 or bit 5 be set individually else unusual effects may occur. Bit 6 has no effect unless one of bit 4 or bit 5 is set.

EGA Mode Compatible Registers

There are sixty-five EGA Mode compatible registers which control the characteristics of the Extended Graphics Environment. These registers are are grouped into five logical groups, EGC (External) Control, Attribute Controller, Sequencer, Graphics Controller, and the CRTC Controller. They are listed in their respective groupings below:

Register Name	Port	R/W	Index
EGC Control	3C2	WO	-
EGC Status	3C2	RO	-
Attribute Controller Address	3C0	WO	-
Palette Registers (0-F)	3C0	WO	00-0F
Mode Control Register	3C0	WO	10
Extended Graphics Border	3C0	WO	11
Color Plane Enable	3C0	WO	12
Horizontal Panning	3C0	WO	13
Sequencer Address	3C4	WO	-
Clock Mode	3C5	WO	01
Color Plane Write	3C5	WO	02
Character Set Select	3C5	WO	03
Memory Mode Select	3C5	WO	04
Graphics Controller Address	3CE	WO	-
Set/Reset	3CF	WO	00
Enable Set/Reset	3CF	WO	01
Color Plane Compare	3CF	WO	02
Data Rotate	3CF	WO	03
Color Plane Read	3CF	WO	04

Register Name	Port	R/W	Index
Graphics Mode Register 1	3CF	WO	05
Graphics Mode Register 2	3CF	WO	06
Color No Care	3CF	WO	07
Write Mask	3CF	WO	08
CRT Controller Address	3B4/3D4	WO	-
Horizontal Total	3B5/3D5	WO	00
Horizontal Display End	3B5/3D5	WO	01
Start Horizontal Blanking	3B5/3D5	WO	02
End Horizontal Blanking	3B5/3D5	WO	03
Start Horizontal Retrace	3B5/3D5	WO	04
End Horizontal Retrace	3B5/3D5	WO	05
Vertical Total	3B5/3D5	WO	06
CRTC Overflow	3B5/3D5	WO	07
Preset Row Scan	3B5/3D5	WO	08
Maximum Scan line	3B5/3D5	WO	09
Cursor Start	3B5/3D5	WO	0A
Cursor End	3B5/3D5	WO	0B
Start Address High	3B5/3D5	R/W	0C
Start Address Low	3B5/3D5	R/W	0D
Cursor Address High	3B5/3D5	R/W	0E
Cursor Address Low	3B5/3D5	R/W	0F
Start Vertical Retrace	3B5/3D5	WO	10
Light Pen High	3B5/3D5	RO	10
End Vertical Retrace	3B5/3D5	WO	11
Light Pen Low	3B5/3D5	RO	11
Vertical Display Enable End	3B5/3D5	WO	12
Offset	3B5/3D5	WO	13
Underline Location	3B5/3D5	WO	14
Start Vertical Blanking	3B5/3D5	WO	15
End Vertical Blanking	3B5/3D5	WO	16
CRTC Mode Control	3B5/3D5	WO	17
Line Compare Register	3B5/3D5	WO	18
Status Port	3B5/3D5	RO	-

The CRT Controller registers will either reside in the 3B- range for monochrome display mode or in the 3D- range for color display modes. This variable address selection is by way of bit 0 in the EGC Control register at I/O address 3C2.

EGC External Control Registers

The EGC External Control registers comprise a group of two registers which enable reading of information such as switches and enable a number of other hardware setup functions.

EGC Control Register (3C2 Out)

The EGC Control Register is a write only register which resides at I/O address 3C2. Its format is as follows.

Bit	Function
7	VSYNC Polarity.
6	HSYNC Polarity.
5	Alternate (64K) Text page Select.
4	External Video Enable.
3-2	Clock Rate Select / Switch Sense Select.
1	Display RAM Enable.
0	CRTC 3BX/3DX I/O Address Select.

Bit 7 = '0' for positive VSYNC polarity, and bit 7 = '1' for negative VSYNC polarity.

Bit 6 = '0' for positive HSYNC polarity, and bit 6 = '1' for negative HSYNC polarity.

Bit 5 enables the selection of an alternate 64K display RAM for text modes (BIOS Modes 0 - 3 and 7). Bit 5 = '0' for the default (Low) 64K Text Page and bit 5 = '1' for the alternate (High) 64K Text Page select.

Bit 4 when set disables internal video and enables a data path for video from an external (Features) connector. However no such connector is provided on the PC1640 main board IGA. Bit 4 should always be written as zero.

Bits 2 and 3 form a two-bit field which selects one of two clock rates. When both bits are zero (bit 3 = bit 2 = '0') then 14MHz clock is used as the clock source and when bit 3 is zero and bit 2 is one (bit 3 = '0', bit 2 = '1') then 16MHz is used as the IGA clock source. The other two combinations are not valid for the PC1640. In addition to selecting the clock rate Bits 2 and 3 form a two bit field for reading the system board switches 1 - 4 and comprise a ones complement switch select field (See EGC Status - page 35).

Bit 1 = '0' disables CPU access to the display RAM and bit 1 = '1' enables CPU access to the display RAM.

Bit 0 maps the CRTC for Monochrome I/O addressing or Color I/O addressing. When bit 0 = '0' the CRTC resides at 3B4 & 3B5 for monochrome mode and when bit 0 = '1' the CRTC resides at 3D4 & 3D5 for color mode. The Status Port register is also remapped from I/O address 3BA in monochrome mode to I/O address 3DA in color mode.

EGC Status Register (3C2 In)

The EGC Status Register is a read only register which resides at I/O address 3C2. Its format is as follows.

Bit	Function
7	VSYNC Interrupt Active.
6 - 5	Not Used.
4	Switch Sense.
3 - 0	Not Used.

Bit 7 = '1' when the VSYNC is waiting and bit 7 = '0' when VSYNC interrupt request has been cleared or is inactive.

Bit 4 works in conjunction with bits 2 and 3 of the EGC Control register to form a switch sense selector. When bit 4 = '0' the selected switch (1 - 4) is closed. The following tables gives the switch select settings.

EGC Control Reg Out (3C2)			EGC Status Reg In(3C2)
Bit	3	2	Bit 4
	1	1	-Switch 1
	1	0	-Switch 2
	0	1	-Switch 3
	0	0	-Switch 4

The IGA ROM BIOS reads these switches during the Power On Self Test (POST) initialization phase and deposits the switch complement value in the system storage location 0:488. Bits 0 - 3 correspond to the ones-complement of switches 1 - 4 respectively. The four MS bits of location 0:488 will always be ones on the PC1640.

Attribute Controller Registers

The Attribute Controller is a hardware grouping within the Extended graphics Controller which provides for color management. The major components of this section of logic are the palette registers which allow for the selection of 16 of 64 colors at any one time.

Attribute Controller Address Register

The Attribute Controller Address Register is a write only register which at I/O address 3C0h. When written to, its contents represent an index into the Attribute Controller register set which also resides at I/O address 3C0. The processor must first output the Index value followed immediately by the register value to be written. Any divergence from this two byte output scheme will cause following operations to confuse Index and data values.

Palette Registers (00h - 0Fh)

The sixteen Palette Registers reside at index positions 00h through 0Fh in the attribute controller and the bit assignments for each is as follows:

Bit	Function	Video Pin
7 - 6	Not Used.	-

Bit	Function	Video Pin
5	Secondary Red Video	2 (r)
4	Secondary Green/Intensity	6 (g/I)
3	Secondary Blue/Mono Video	7 (b/V)
2	Primary Red	3 (R)
1	Primary Green	4 (G)
0	Primary Blue	5 (B)

The Palette registers are each six bits wide and the bits are arranged such that the output color signals will correspond (from MS to LS bits) in the order rgbRGB. The major point to note is that secondary green and Intensity are on the same video output pins. The IGA ROM BIOS will initially load the palette registers such that the secondary green bit (bit 4) will be set for all the high order registers which require Intensity set for the four wire IRGB 16 color displays. This scheme allows the color selection to resemble a CGA's color selection. Any of the color fields which contain four bits (i.e. attribute bytes and color planes) actually select their respective palette register for the selection of color signals are to be output.

Mode Control Register (10h)

The Mode Control Register is a write only register which resides at index 10h in the attribute controller at I/O address 3C0. This register specifies the major operational mode and some other characteristics of selected modes. The Mode Control Register format is as follows:

Bit	Function
7 - 4	Not Used.
3	Background Intensity / Enable Blink.
2	Enable Line Graphics.
1	Display Type.
0	Graphics/AlphaNumeric (AN) Mode.

In Alphanumeric mode bit 3 ties up with the MS bit of the text attribute byte to enable MS attribute bit to either be the Blink Enable bit when bit 3 is set or to enable the MS attribute bit to be that Background Intensity bit when bit 3 is reset. The blink rate is 16 display frames ON and 16 display frames OFF which is half the cursor blink rate. In Color Graphics Mode, setting bit 3 allows for a specific pixel to alternate between two colors by toggling COL3 at the blink rate.

Bit 2 when set enables the special line graphics characters originally designed for the MDA by causing the ninth dot to be the same as the eight dot for an effective 9x14 character cell.

Bit 1 when set selects the monochrome display attributes and when clear selects color display attributes.

Bit 0 selects graphics mode when set and Alphanumeric mode when reset.

Extended Graphics Border Register (11h)

The Extended Graphics Border Register is located at index 11h in the attribute controller registers at I/O address 3C0. It is also called the Overscan register and is a 6-bit wide write only register. Its function is to specify the Extended graphics mode border color. It can be locked out by setting bit 5 in the IGA Extended Mode Control Register (See page 29).

The format of the Extended Graphics Border register is the same as the Palette Registers (see page 36). When in monochrome mode the Extended Graphics Border register should be set to zero.

Color Plane Enable Register (12h)

The Color Plane Enable Register is a write only register residing at index 12h in the attribute controller at I/O address 3C0. Its function is to enable or disable which color planes are active for producing video, thus acting as an overall palette selector. The format is as follows:

Bit	Function
7 - 6	Not Used.
5 - 4	Video Status Multiplex (MUX)
3	Enable Color Plane 3
2	Enable Color Plane 2
1	Enable Color Plane 1
0	Enable Color Plane 0.

When bits 0-3 are set to '1's, the corresponding color plane is enabled.

The two bits 4 and 5 allow for diagnostics to be run on the hardware by selecting which two color plane's outputs will be gated to bits 4 and 5 of the Status Port Register at 3BA / 3DA. The Following table gives the correspondence between the video status MUX bits and the Status Port.

Video Status MUX (3C0)			Status Port Reg (3XA)		
Bits	5	4	Bits	5	4
	0	0		R	B
	0	1		r	g
	1	0		b	G

The upper case letters represent the primary colors and the lower case letters represent the secondary colors. The 1, 1 case is not used.

Horizontal Panning Register (13h)

The Horizontal Panning Register is a write only register residing at index 13h in the attribute controller at I/O address 3C0. Its function is to select the number of pixels to shift the video date horizontally left. Panning is available in both alphanumeric and graphics modes. The monochrome alphanumeric shift factor is a maximum of 9 pixels. The maximum shift factor for all other modes is 8 pixels. The bit layout of the Horizontal Panning Register is as follows:

Bit	Function
7 - 4	Not Used.
3 - 0	Horizontal Panning Value. (Bits 3-0)

For the 9-bit maximum monochrome alphanumeric mode the shift sequence is pixel 8, then pixels 0 through 7. For all other modes the pixel shift sequence is pixels 0 through 7.

Sequencer Registers

The Sequencer is a hardware grouping within the Extended graphics controller which controls video memory accessing, character clocking and character generator mapping. The Sequencer registers consist of an address register with four control registers which are indexed by the address register. Index position zero though listed to select the Reset register is not required in the PC1640 IGA and is not implemented. The remaining registers corresponding to index values 1 through 4 respectively are the Clocking Mode register, the Plane Mask register, the Character Set Select Register, and the Memory Mode register.

Sequencer Address Register

The Sequencer Address Register is a write only register at I/O address 3C4 which selects which sequencer register is configured to I/O address 3C5. The bit layout of the Sequencer address register is as follows:

Bit	Function
7 - 3	Not Used.
2 - 0	Sequencer Address (Bits 2 - 0)

Clock Mode Register (3C5, 1)

The Clock Mode Register is a write only register which resides at I/O address 3C5 when the index register contains 1. Its format is as follows:

Bit	Function
7 - 4	Not Used.
3	Dot Clock Rate.
2	Shift Register Load.
1	Not Used.
0	8/9 Dot Clocks.

Bit 3 controls whether the dot clock will be a direct function of the master clock rate or divided by two. Setting bit 3 causes the input clock to be divided by two. This is used for 320 pixel wide graphics.

Bit 2 when cleared causes the serial video shift registers to be loaded every character clock. Setting bit 2 causes the load rate to be every second character clock. Bit 2 should be cleared when two of the shift registers are chained together for 16-bit character columns.

Bit 0 controls the number of dot clocks generated by character row. It should be cleared for monochrome mode (9-bit wide characters) and set for color mode characters (8-bits wide).

Color Plane Select Register (3C5, 2)

The Color Plane Select Register is a write only register which resides at I/O address 3C5 when the index register contains 2. Its format is as follows:

Bit	Function
7 - 4	Not Used.
3	Write Enable Memory Plane 3.
2	Write Enable Memory Plane 2.
1	Write Enable Memory Plane 1.
0	Write Enable Memory Plane 0.

Setting Bits 0 - 3 write enables the respective memory plane. The CPU can write to any combination of display memory planes in one write cycle by setting the respctive bits. Bits 0 &1 and bits 2 & 3 should have the same values when odd/even modes are selected. (See page page 42 for Odd/Even mode details.)

Character Set Select Register (3C5, 3)

The Color Plane Select Register is a write only register which resides at I/O address 3C5 when the index register contains 3. Its format is as follows:

Bit	Function
7 - 4	Not Used.
3 - 2	Character Set Select A.
1 - 0	Character Set Select B.

There are four alternate character sets per character map each containing 128 characters each thus allowing up to 512 characters to be accessible for any given screen. In addition the IGA Extended Mode register (bit 3) enables an alternate set of character maps to be selected for an extended total of 1024 available characters. The alternate map character selections is disabled then character map select A equals character map select B (that is, bits 0 & 1 equal bits 2 & 3. When enabled, attribute bit 3 selects when reset selects character map A and when set selects character map B. The following table gives the character set selections for the combinations of bits 0 - 3.

Character Set				Selected Map vs	
Sel A		Sel B		Attribute Bit 3	
3	2	1	0	1	0
0	0	0	0	--	--
0	0	0	1	00	01
0	0	1	0	00	02
0	0	1	1	00	03
0	1	0	0	01	00
0	1	0	1	--	--
0	1	1	0	01	02
0	1	1	1	01	03
1	0	0	0	02	00
1	0	0	1	02	01
1	0	1	0	--	--
1	0	1	1	02	03
1	1	0	0	03	00
1	1	0	1	03	01
1	1	1	0	03	02
1	1	1	1	--	--

The dashes mean disabled and that attribute bit 3 becomes the intensity select bit.

When the IGA Extended Mode Register bit 3 is set then an alternate set of character maps are enabled from plane 3 and in this case the map numbers in the table should be logically incremented by 4. The following figure illustrates the complete PC1640 character set organization.

	Plane 2	Plane 3
8 Kb	Char Set 0	Char set 4
8 Kb	Not Used.	Not Used.
8 Kb	Char Set 1	Char set 5
8 Kb	Not Used.	Not Used.
8 Kb	Char Set 2	Char set 6
8 Kb	Not Used.	Not Used.
8 Kb	Char Set 3	Char set 7
8 Kb	Not Used.	Not Used.

Memory Mode Select Register (3C5, 4)

The Memory Mode Select Register is a write only register which resides at I/O address 3C5 when the index register contains 4. Its format is as follows:

Bit	Function
7 - 3	Not Used.
2	Odd/Even.
1	Extended Memory. (1)
0	Not Used.

Bit 1 indicates that 256Kb of display RAM is present and should always be set on the PC1640.

Bit 2 when set selects chained addressing mode whereby even CPU address access character sets 0 and 2 while odd CPU addresses access character sets 1 and 3. When bit 1 is cleared, unchained addressing mode is selected and CPU accesses sequentially access data within the character bit maps.

Graphics Controller Registers

The Graphics Controller is a hardware grouping within the Extended graphics Controller which directs memory data to the attribute controller and to the CPU. In graphics modes serialised memory data is sent to the attribute controller. The Graphics Controller and the Attribute Controller are the two logic groupings which make up a total functional logic grouping called the Video Controller. The Graphics Controller registers consists of an address register and nine write only data registers.

Graphics Controller Address (3CE)

The Graphics Controller Address Register is a write only register at I/O address 3CE which serves as an address pointer for nine data registers at I/O address 3CF. The bit layout is as follows:

Bit	Function
7 - 4	Not Used.
3 - 0	Graphics Address Pointer.

Bits 0 - 3 select the active register at I/O address 3DF.

Set/Reset (3CF, 0)

The Set/Reset Register is a write only register which resides at I/O address 3CF when the Graphics Controller Address Register contains 0. Its format is as follows:

Bit	Function
7 - 4	Not Used.
3	Set/Reset Bit 3
2	Set/Reset Bit 2
1	Set/Reset Bit 1

Bit	Function
0	Set/Reset Bit 0

The value in Bits 0 - 3 is written to plane 0 - 3 respectively, providing that the corresponding bit in the Enable Set/Reset register (below) is also set and that Graphics Mode Register 1 is programmed for write mode 0 and that the CPU data bit being written contains a 1.

For example if it were desired to write bits 7, 5 and 0 of a particular display location with light cyan then the Set/Reset register bits 3-0 would be set to 1011 binary (the pattern for light cyan), then the Enable Set/Reset register (page 44) is set to 0Fh and the particular display memory location would be written with 1010001 binary. The bit positions containing zeroes would remain unchanged and the pixels corresponding to bits 7, 5, and 0 would be light cyan. The equation for bit number to pixel number translation is: Pixel Number = (7 - Bit Number).

Enable Set/Reset (3CF, 1)

The Enable Set/Reset Register is a write only register which resides at I/O address 3CF when the Graphics Controller Address Register contains 1. Its format is as follows:

Bit	Function
7 - 4	Not Used.
3	Enable Set/Reset Bit 3
2	Enable Set/Reset Bit 2
1	Enable Set/Reset Bit 1
0	Enable Set/Reset Bit 0

Setting Bits 0 - 3 will qualify the corresponding Set/Reset register bit to be written as described above. If write mode is 0 and Set/Reset is not enabled on a color plane, the plane's original contents are preserved.

Color Plane Compare Register (3CF, 2)

The Color Plane Compare Register is a write only register which resides at I/O address 3CF when the Graphics Controller Address Register contains 2. The Color Plane Compare register is activated when the Graphics Mode Register 1 (page 47) bit 3 is set enabling compare mode. When color compare mode is active, CPU reads of the display RAM area will return the results of a comparison between the Color Plane Compare Register and the color planes rather than the actual memory contents.

The Color Plane Compare Register is as follows:

Bit	Function
7 - 4	Not Used.
3	Color Plane Compare 3.
2	Color Plane Compare 2.
1	Color Plane Compare 1.
0	Color Plane Compare 0.

The color pattern in bits 0 - 3 is compared with the memory location being read by the CPU. The bits returned represent which bits actually match the color pattern. The Color No Care register (described below) specify which planes participate in the comparison.

In the example for writing light cyan using the Set/Reset register, loading the color compare register with 1011 binary and reading the same display memory location would return the 1010001 binary value written. This is interpreted to mean that the pixels corresponding to bits 7, 5 and 0 are light cyan and the other pixels are some other color(s). To determine what other colors are in the same byte would require either performing a color compare scan of the 15 other color combinations or discreetly reading each of the color planes and considering what color the four bits for each pixel actually represent.

Data Rotate Register (3CF, 3)

The Data Rotate Register is a write only register which resides at I/O address 3CF when the Graphics Controller Address Register contains 3. This register controls a number of logical operations which can be performed when the CPU writes to the display RAM. Its layout is as follows:

Bit	Function
7 - 5	Not Used.
4 - 3	Function Select.

Bit	Function
2 - 0	Rotate Count.

The Function select field (bits 3 & 4) select one of four functions which can be performed between the existing contents of a memory plane and the new data being written. These operations are as follows:

Bit 4 3	Selected Function
0 0	Replace Existing Data.
0 1	Logical 'AND'.
1 0	Logical 'OR'.
1 1	Logical 'XOR'.

When bits 3 and 4 are both reset, data from the CPU replaces the existing data and no logical operation is performed.

When a logical operation is enabled, the planes which are write enabled (see Color Plane Enable register) will be read and the specified logical operation performed between the existing contents and the data from the CPU and the resultant value stored in the respective display memory plane.

Bits 0 - 3 control a Rotate Right function on CPU data being written to the display planes. When write mode zero is programmed in the Graphics Mode register 1 (See Below), the contents of bits 0 - 3 represent the shift count which will be performed on the data from the CPU. The rotate is end around, the LS bits are rotated to the MS bits and the resultant value is written to the selected memory plane(s).

Color Plane Read (3CF, 4)

The Color Plane Read Register is a write only register which resides at I/O address 3CF when the Graphics Controller Address Register contains 4. Its format is as follows:

Bit	Function
7 - 2	Not Used.
1 - 0	Read Select.

The CPU reads from whichever of the four color planes in display RAM selected according to the Read Select field. If the original contents of the color paletter registers are maintained then the selection relates to the IRGB color signals as follows:

Bit 1 0	Selected Color Plane
0 0	Blue. (Plane 0)
0 1	Green. (Plane 1)
1 0	Red. (Plane 2)
1 1	Intensity. (Plane 3)

Bit 2 is also connected with the read select decoding and should be maintained reset else no plane will be selected for reading.

Graphics Mode Register 1 (3CF, 5)

The Graphics Mode Register 1 is a write only register which resides at I/O address 3CF when the Graphics Controller Address Register contains 5. Its format is as follows:

Bit	Function
7 - 6	Not Used.
5	Shift Register Format.
4	Not Used.
3	Read Mode.
2	Not Used.
1 - 0	Write Mode

Bit 5 controls the configuration of the four serializer shift registers within the graphics controller. In the normal case when bit 5 is reset, data from planes 0 - 3 are shifted out of shift registers 0 - 3 respectively going out MS bit first. When bit 5 is set to one then the even numbered bits are shifted out of the even numbered shift registers and the odd numbered bits are shifted out of the odd shift registers. This means that shift register 0 shifts bits 6, 4, 2 & 0 of Plane 0, followed by bits 6, 4, 2 & 0 of plane 1. Shift register 1

shifts bits 7, 5, 3 & 1 of Plane 0 followed by bits 7, 5, 3 & 1 of Plane 1. Shift registers 2 and 3 perform the same even/odd shifting pattern for color planes 2 & 3 as described above.

Bit 3 when clear enables the Color Plane Read register to control the color plane selection of CPU reads. When bit 3 is set then the Color Plane Compare register controls the CPU data read back (as described on page 44).

Bits 0 and 1 form a two bit Write Mode field which specifies the manner in which the IGA handles graphics data written to the memory planes. The following table gives the valid options:

Bits 1 0	Selected Write Mode Description
0 0	CPU data written to planes 0 - 3 is controlled by Color Plane Write register, the Data Rotate Register, the Set/Reset Registers as described in the register descriptions.
0 1	When a CPU write cycle is performed Planes 0 - 3 are written with data from the previous CPU read operation.
1 0	Planes 0 - 3 are written with 1's or 0's based on bits 0 - 3. For example, if the value of data bit 3 = 1 then plane 3 would be written with FFh.

The 1,1 state is not legal and should not be used.

Graphics Mode Register 2 (3CF, 6)

The Graphics Mode Register 2 is a write only register which resides at I/O address 3CF when the Graphics Controller Address Register contains 6. Its format is as follows:

Bit	Function
7 - 4	Not Used.
3 - 2	Memory Mapping Mode.
1	Enable Odd/Even Chaining.
0	Not Used.

Bits 2 and 3 form a two bit field which specifies the regeneration buffer origin and size parameters as follows:

Bit 3 2	Origin	Size
0 0	A000h	128K Bytes
0 1	A000h	64K Bytes
1 0	B000h	32K Bytes
1 1	B800h	32K Bytes

Setting bit 1 causes the processor address bit A0 control the selection of odd/even memory planes rather than contiguous odd/even addresses. The usual function performed by address bit A0 is shifted to the high order address bit.

Color No Care Register (3CF, 7)

The Color No Care Register is a write only register which resides at I/O address 3CF when the Graphics Controller Address Register contains 7. Its format is as follows:

Bit	Function
7 - 4	Not Used.
3	Color Plane 3 No Care.
2	Color Plane 2 No Care.
1	Color Plane 1 No Care.
0	Color Plane 0 No Care.

This register ties up with the Color Compare register and can be considered a color compare mask in that setting bits in the 'No Care' register inhibits the corresponding color plane from participating in the comparison process when a color compare read is performed.

Write Mask Register (3CF, 7)

The Write Mask Register is a write only register which resides at I/O address 3CF when the Graphics Controller Address Register contains 8. Its format is as follows:

Bit	Function
7 - 0	Write Mask. (Bits 7 - 0)

Resetting bits in the Write Mask register disables the respective bits from being written to in the display planes. The Write Mask register is programmed to all ones during display mode initialization so that the all eight bits are normally stored into display memory.

The Write Mask register affects all data written by the CPU including the rotate and logical operations. The hardware performs a read-before-write operation in order to preserve the protected bits.

EGA Mode CRT Controller Registers

The CRT Controller is a hardware grouping within the IGA which controls horizontal and vertical synchronization as well as cursor, underline and blink timing. It also generates addressing for the display regeration buffer and dynamic RAM refresh controls.

The CRT Controller contains data 27 registers which are accessed through an index register which must be loaded prior to accessing a particular CRTC data register. The CRTC data registers reside at either I/O address 3B5 or I/O address 3D5 depending on whether the IGA is operating in monochrome mode or color mode respectively.

This alternate I/O addressing scheme is so that an alternate adapter such as a MDA or a CGA can be fitted in the expansion slots and there will be no I/O address conflicts with the other display adapter's CRTC. This means that the system can only support one color adapter and one monochrome adapter at the same time. If the IGA is driving a monochrome monitor then a CGA can be fitted in the expansion slots. Conversely, if the IGA is driving a color display then only a monochrome adapter can be fitted in the expansion slots. In no case can an EGA be fitted in the expansion slots when the IGA is active because the other EGA will overlay the IGA's ROM BIOS as well as the registers in the 3CX I/O address range. It is also possible to configure the IGA to overlay other display adapter's display RAM.

An additional register, the Status PORT Register, is not technically part of the CRTC but is closely associated with it also explained with this hardware grouping.

CRT Controller Address Register

The CRT Controller Address Register is a write only register which resides at either I/O address 3B4 or I/O address 3D4. When written to, its contents represent an index into the CRTC data registers at I/O addres 3X5 (X=B for Monochrome and X=D for color).

Bit	Function
7 - 5	Unused.
4 - 0	CRT Controller Address. (00h - 18h)

Values greater than 18h are not valid and should not be used.

Horizontal Total Register (3B5/3D5, 0)

The Horizontal Total Register is a write only register which resides at I/O address 3X5 when the CRT Controller Address Register contains 00h. Its format is as follows:

Bit	Function
7 - 0	Horizontal Total Value. (-2)

The Horizontal Total Register specifies the number of characters (minus 2) in the horizontal scan interval inclusive of the retrace period. This value is the basis of all horizontal (and vertical) timing.

Horizontal Display End Register (3B5/3D5, 0)

The Horizontal Display End Register is a write only register which resides at I/O address 3X5 when the CRT Controller Address Register contains 01h. Its format is as follows:

Bit	Function
7 - 0	Horizontal Display End Value. (-1)

The Horizontal Display End value is the number of characters to be displayed per horizontal line. The actual number of characters displayed per horizontal line is one less than the contents of this register.

Start Horizontal Blanking Register (3B5/3D5, 2)

The Start Horizontal Blanking Register is a write only register which resides at I/O address 3X5 when the CRT Controller Address Register contains 02h. It is formatted as follows:

Bit	Function
7 - 0	Start Horizontal Blanking Value.

The Horizontal blanking signal becomes active when the horizontal character count is equal to the value in this register.

End Horizontal Blanking Register (3B5/3D5, 3)

The End Horizontal Blanking Register is a write only register which resides at I/O address 3X5 when the CRT Controller Address Register contains 03h. It is formatted as follows:

Bit	Function
7	Unused.
6 - 5	Display Enable Skew Value.
4 - 0	End Horizontal Blanking Value.

The Horizontal blanking signal becomes inactive when the lower 5 bits of the horizontal character count is equal to the value stored in bits 0 - 4 of this register.

Because of the sequential access to display memory by the CRTC, the video data is skewed relative to the horizontal timing. The Display Enable Skew value corrects for this skew by causing a delay of a number of character clocks equal to the value stored in bits 5 and 6.

The following equation specifies the value for bits 0 - 4: End Horizontal Blanking = (Start Blanking Value + Width of Blanking) Modulo 32.

Start Horizontal Retrace Register (3B5/3D5, 4)

The Start Horizontal Retrace Register is a write only register which resides at I/O address 3X5 when the CRT Controller Address Register contains 04h. Its format is as follows:

Bit	Function
7 - 0	Start Horizontal Retrace Value.

The Horizontal retrace signal becomes active when the horizontal character count is equal to the value in this register.

End Horizontal Retrace Register (3B5/3D5, 5)

The End Horizontal Retrace Register is a write only register which resides at I/O address 3X5 when the CRT Controller Address Register contains 5. Its bit assignment is as follows:

Bit	Function
7	Even/Odd Starting Address.
6 - 5	Horizontal Retrace Skew.
4 - 0	End Horizontal Retrace Value.

The Horizontal Retrace signal becomes inactive when the lower 5 bits of the horizontal character count is equal to the value stored in bits 0 - 4 of this register.

Setting bit 7 specifies that the CRT memory address after a horizontal retrace be an odd memory address. Resetting bit 7 specifies an even starting address. Generally this bit should be reset but it is useful in applications where horizontal pixel panning is required.

The value bits 5 and 6 specify the horizontal retrace skew (0 - 3) in character clock counts. The horizontal retrace signal will be delayed by this value. A number of internal timing signals are generated by the falling edge of horizontal retrace. To guarantee proper latching, retrace is started before the end of display enable, and then skewed by several character clocks for correct screen centering.

The following equation specifies the value for bits 0 - 4: End Horizontal Retrace = (Start Horizontal Retrace Value + Width of Retrace)

Modulo 32.

Vertical Total Register (3B5/3D5, 6)

The Vertical Total Register is a write only register which resides at I/O address 3X5 when the CRT Controller Address Register contains 06h. Its format is as follows:

Bit	Function
7 - 0	Vertical Total Value. (-2)

The Vertical Total Register specifies the low 8 bits of a 9 bit value. The 9th bit is located in the CRTC Overflow register. The value in this register represents the total number of scan lines per frame minus two.

CRTC Overflow Register (3B5/3D5, 7)

The CRTC Overflow Register is a write only register which resides at I/O address 3X5 when the CRT Controller Address Register contains 07h. It is formatted as follows:

Bit	Function
7 - 5	Unused.
4	Line Compare Bit 8.
3	Start Vertical Blanking Bit 8.
2	Start Vertical Retrace Bit 8.
1	Vertical Display Enable End Bit 8.
0	Vertical Total Bit 8.

This register specifies the 9th bit of several other control registers and will be set for a value specifying a scan line greater than 255.

Preset Row Scan Register (3B5/3D5, 8)

The Preset Row Scan Register is a write only register which resides at I/O address 3X5 when the CRT Controller Address Register contains 08h. Its format is as follows:

Bit	Function
7 - 5	Unused.
4 - 0	Row Scan Preset Value.

Bits 0 - 4 specify the value of the starting row scan count after a vertical retrace.

Maximum Scan Line Register (3B5/3D5, 9)

The Maximum Scan Line Register is a write only register which resides at I/O address 3X5 when the CRT Controller Address Register contains 09h. Its bit assignment is as follows:

Bit	Function
7 - 5	Unused.
4 - 0	Maximum Scan Line. (-1)

The value programmed into bits 0 - 4 specifies the number of scan lines per character row minus one.

Each horizontal retrace increments the horizontal row scan counter. The horizontal row scan counter is cleared when it equals the Maximum Scan Line Register.

Cursor Start Register (3B5/3D5, 0A)

The Cursor Start Register is a write only register which resides at I/O address 3X5 when the CRT Controller Address Register contains 0Ah. Its bit assignment is as follows:

Bit	Function
7 - 6	Unused.
5	Cursor Off.

Bit	Function
4 - 0	Row Scan Cursor Begins. (-1)

Setting bit 5 turns the cursor off and resetting bit 5 enables the cursor.

Bits 0 - 4 specifies the row scan number of a character line where a cursor is begin. Programming Cursor Start greater than Cursor End disables the cursor.

Cursor End Register (3B5/3D5, 0B)

The Cursor End Register is a write only register which resides at I/O address 3X5 when the CRT Controller Address Register contains 0Bh. Its bit assignment is as follows:

Bit	Function
7	Unused.
6 - 5	Cursor Skew Control.
4 - 0	Row Scan Cursor Ends. (+1)

The value stored in bits 0 - 4 represents the last line in each character for displaying the cursor plus one.

Bits 5 & 6 specify the number of character clocks to delay the cursor. For each clock count the cursor moves right by one position.

Programming Cursor Start and Cursor End may not always produce the expected results especially in instances where cursor end is greater than max line scan.

Start Address High Register (3B5/3D5, 0C)

The Start Address High Register is a read/write register which resides at I/O address 3X5 when the CRT Controller Address Register contains 0Ch. Its bit assignment is as follows:

Bit	Function
7 - 0	Display Start Address. (High Order Bits)

The Start Address High and the Start Address Low registers form a 16 bit value stored which is added to the display origin programmed by Graphics Mode Register 2 to form current display buffer starting address. In text modes the IGA ROM BIOS sets up the CRTC such that the programmed display start address value specifies character/attribute byte pairs from the display origin address for an effective two to one (word oriented) offset ratio. In graphics modes the value is a straight one to one byte to offset ratio.

Start Address Low Register (3B5/3D5, 0D)

The Start Address Low Register is a read/write register which resides at I/O address 3X5 when the CRT Controller Address Register contains 0Ch. Its bit assignment is as follows:

Bit	Function
7 - 0	Display Start Address. (Low Order Bits)

This register specifies the 8 low order bits of the display start address value.

Cursor Location High Register (3B5/3D5, 0E)

The Cursor Location High Register is a read/write register which resides at I/O address 3X5 when the CRT Controller Address Register contains 0Eh. Its format is as follows:

Bit	Function
7 - 0	Cursor Location. (High Order Bits)

The cursor location is a 16 bit quantity similar to the start address register. The Cursor Location High register contains the 8 high order bits of the cursor location.

Cursor Location Low Register (3B5/3D5, 0F)

The Cursor Location Low Register is a read/write register which resides at I/O address 3X5 when the CRT Controller Address Register contains 0Fh. Its format is as follows:

Bit	Function
7 - 0	Cursor Location. (Low Order Bits)

The Cursor Location Low register specifies the 8 low order bits of the cursor location.

Start Vertical Retrace Register (3B5/3D5, 10 Write)

The Start Vertical Retrace Register is a write only register which resides at I/O address 3X5 when the CRT Controller Address Register contains 0Fh. It is formatted as follows:

Bit	Function
7 - 0	Vertical Retrace Position. (Lower 8 bits)

The Start Vertical Retrace register is a 9-bit register and the value stored specifies the position of the leading edge of the vertical retrace signal in terms of scan lines. This register contains the 8 low order bits of the Vertical Retrace start value and the 9th bit is located in the CRTC Overflow register.

End Vertical Retrace Register (3B5/3D5, 11 Write)

The End Vertical Retrace Register is a write only register which resides at I/O address 3X5 when the CRT Controller Address Register contains 0Fh. It is formatted as follows:

Bit	Function
7 - 6	Unused.
5	Vertical Interrupt Enable.
4	Vertical Interrupt Clear.
3 - 0	End vertical Retrace Value.

The vertical retrace signal becomes inactive when the lower four bits of the horizontal scan count equal bits 0 - 3 of this register.

Bit 5 when set to 1 causes the IRQ pin to go to a tri-stated high-Z state and when it is zeroed the IGA generates IRQ2 when vertical retrace goes active.

Bit 4 is reset to zero to clear the vertical retrace interrupt signal. The interrupt handler routine for the CRT Interrupt should set this bit low then high in order to clear the interrupt.

The following equation specifies the value for bits 0 - 3: End Vertical Retrace = (Start Vertical Retrace Value + Width of retrace) Modulo 16.

Light Pen High Register (3B5/3D5, 10 Read)

The Light Pen High Register is a read only register which resides at I/O address 3X5 when the CRT Controller Address Register contains 10h. Its format is as follows:

Bit	Function
7 - 0	Light Pen Address. (High Order Bits)

The Light Pen Address is a 16-bitvalue representing the value of the CRTC address register when the light pen was triggered. This register returns the eight high order bits.

Cursor Location Low Register (3B5/3D5, 11 Read)

The Light Pen Low Register is a read only register which resides at I/O address 3X5 when the CRT Controller Address Register contains 11h. Its format is as follows:

Bit	Function
7 - 0	Light Pen Address. (High Order Bits)

An I/O read of this register returns the eight low order bits of the light pen address value.

Vertical Display End Register (3B5/3D5, 12)

The Vertical Display End Register is a write only register which resides at I/O address 3X5 when the CRT Controller Address Register contains 12h. Its format is as follows:

Bit	Function
7 - 0	Vertical Display End Value. (-1) (8 Low Order Bits)

The vertical display end enable value is the represents the number of scan lines displayed minus one. This register contains the 8 low order bits of the vertical display end value and the 9th bit is located in the CRTC Overflow register.

CRTC Offset Register (3B5/3D5, 13)

The CRTC Offset Register is a write only register which resides at I/O address 3X5 when the CRT Controller Address Register contains 13h. Its format is as follows:

Bit	Function
7 - 0	CRTC Offset Value.

The CRTC Offset value represents the logical line width of the display screen. The value in this register determines the starting address of the next character row and is the number of character/attribute pairs in a display line.

Underline Location Register (3B5/3D5, 14)

The Underline Location Register is a write only register which resides at I/O address 3X5 when the CRT Controller Address Register contains 14h. Its format is as follows:

Bit	Function
7 - 5	Unused.
4 - 0	Underline Location Value. (-1)

The value in bits 0 - 4 represents the horizontal row scan count at which the underline will occur. The specified value is one less than the desired scan line number.

Start Vertical Blanking Register (3B5/3D5, 15)

The Start vertical Blanking Register is a write only register which resides at I/O address 3X5 when the CRT Controller Address Register contains 15h. Its format is as follows:

Bit	Function
7 - 0	Start Vertical Blanking Value (8 Low Order Bits)

This register contains the 8 low order bits of the horizontal scan line count at which the vertical blanking signal becomes active. The 9th bit of this value is located in the CRTC Overflow register.

End Vertical Retrace Register (3B5/3D5, 16)

The End Vertical Blanking Register is a write only register which resides at I/O address 3X5 when the CRT Controller Address Register contains 16h. Its format is as follows:

Bit	Function
7 - 5	Unused.
4 - 0	End Vertical Blanking Value.

The vertical blanking signal becomes inactive when the lower 5 bits of the horizontal scan count match bits 0 - 4 of this register.

The following equation specifies the value for bits 0 - 4: End Vertical Blanking = (Start Vertical Blanking Value + Width of Blanking) Modulo 32.

CRTC Mode Register (3B5/3D5, 17)

The CRTC Mode Register is a write only register which resides at I/O address 3X5 when the CRT Controller Address Register

contains 17h. Its is formatted as follows:

Bit	Function
7	H/V Retrace Enable.
6	Word/Byte Mode.
5	64K Address Wrapping.
4	Not Used.
3	Memory Address Count Mode.
2	Horizontal Retrace Clock Select.
1	Row Scan Counter Select.
0	200 Line Compatibility Mode Select.

Bit 7 controls whether horizontal and vertical retrace is enabled or disabled. Resetting bit 7 to zero clears horizontal and vertical retrace. Setting bit 7 to one enables H & V retrace signals.

Bit 6 when zero selects word mode and a one selects byte mode. This bit should set to zero in non EGA modes (IGA Extended Mode register bit 6 = 1). In word mode the memory is addressed in two byte pairs and the MS bit of the address counter is tied to address bit 0.

Bit 5 selects address wrap at 64K byte boundaries. Since 256K bytes of display memory are installed on the main board this bit should be set to a one.

Bit 3 when set to one causes the memory address counter to be clocked by character clock input to be divided by two. When bit 3 is zero the memory address counter is clocked by straight character clock input.

Bit 2 selects whether the vertical timing counter is to be clocked by horizontal retrace or by horizontal retrace divided by two. Setting bit 2 selects the divide by two mode allowing double vertical resolution.

Bit 1 selects Row Scan counter output configuration. A zero selects row scan counter bit 1 on memory address output bit 14. A one selects row scan counter bit 14 to memory address output bit 14.

Bit 0 allows CGA compatibility mode support by causing alternate display lines to be displaced by 8K bytes. When bit 0 is zero compatibility mode is selected and row scan bit 0 is connected to memory address bit 13 during active display time. When bit 0 is one memory address bit 13 is connected address counter bit 13.

Line Compare Register (3B5/3D5, 18)

The Line Compare Register is a write only register which resides at I/O address 3X5 when the CRT Controller Address Register contains 18h. Its format is as follows:

Bit	Function
7 - 0	Line Compare Value. (8 Low Order Bits)

This register contains the lower eight bits of the line compare target value and the 9th bit is in the CRTC Overflow register. The line compare target value is used to implement a split screen function whereby some areas of the screeen can be made immune to scrolling. When the vertical scan counter equals the line compare target value the memory address generator is cleared to zero. Each subsequent row address is determined by the 16-bit addition of the Start of line latch and the contents of the Offset register.

Status PORT Register (3BA / 3DA)

The Status Port Register resides at either I/O address 3BA or I/O address 3DA depending on the CRTC Mono/Color (bit 0) setting in the IGA External Control Register. Its format is as follows:

Bit	Function
7	Not Used (Color) / -VSYNC (Mono).
6	EGA Mode.
5 - 4	Color Diagnostic. (MUX)
3	-VSYNC (Color) / Mono Video.
2	Light Pen Switch. (-LPSW)
1	Light Pen Strobe.
0	Display Enable.

Bit 7 is always a one in EGA emulation mode (IGA Extended Mode Reg bit 6 = 0) or when mapped into color adapter range (3DA).

In MDA emulation mode (at 3BA) bit 7 = 0 when VSYNC is active and bit 7 = 1 when VSYNC is inactive.

Bit 6 is a one in EGA emulation mode and a zero otherwise.

Bit 4 and 5 are used by diagnostic software for testing the color output signals and are used in conjunction with the Video Status MUX in the Color Plane Enable Register. Refer to page 38 for the Video Status MUX details. The following is a repeat of the Video Mux Table.

Video Status MUX (3C0)		Status Port Reg (3XA)	
5	4	5	4
0	0	R	B
0	1	r	g
1	0	b	G

The primary colors are represented the by capital (RGB) letters and the secondary colors are represented by the small (rgb) letters.

Bit 3 in EGA or Color Mode (3DA) is zero when VSYNC is active and one when VSYNC is inactive. In MDA mode (3BA) is zero when Mono Video is active and one when Mono Video is inactive.

Bit 2 = 0 when the Light Pen switch is closed and bit 2 = 1 when the Light Pen switch is open.

Bit 1 = 0 when the Light Pen strobe has not been triggered and bit 1 = 1 when the Light Pen strobe has been triggered.

Bit 0 = 0 when the Video signal is enabled and bit 0 = 1 when either the vertical or horizontal retrace signals are active. This is a real time indication of the raster scan line status.

1.11.4 IGA BIOS EGA Mode Initialization

The IGA ROM BIOS contains initialization tables for the EGA Mode compatible registers. The IGA BIOS modes were introduced in 1.11.2.5 and consist of a standard set of hardware setups which the BIOS ROM supports. Obviously other hardware setup configurations could be conceived of by the well informed software engineer. However the ROM BIOS modes are the subset of usable capabilities which will ensure transportability to other compatible hardware. The BIOS setups are as follows:

BIOS Mode		0	1	2	3	4	5	6	7	13	14	15	16
EGC Ctl (3C2)		23	23	23	23	23	23	23	A6	23	23	A2	A7
Attribute	(0)	00	00	00	00	00	00	00	00	00	00	00	00
Controller	(1)	01	01	01	01	13	13	17	08	01	01	08	01
Registers	(2)	02	02	02	02	15	15	17	08	02	02	00	02
(3C0)	(3)	03	03	03	03	17	17	17	08	03	03	00	03
	(4)	04	04	04	04	02	02	17	08	04	04	18	04
	(5)	05	05	05	05	04	04	17	08	05	05	18	05
	(6)	06	06	06	06	06	06	17	08	06	06	00	14
	(7)	07	07	07	07	07	07	17	08	07	07	00	07
	(8)	10	10	10	10	10	10	17	10	10	10	00	38
	(9)	11	11	11	11	11	11	17	18	11	11	08	39
	(0A)	12	12	12	12	12	12	17	18	12	12	00	3A
	(0B)	13	13	13	13	13	13	17	18	13	13	00	3B
	(0C)	14	14	14	14	14	14	17	18	14	14	00	3C
	(0D)	15	15	15	15	15	15	17	18	15	15	18	3D
	(0E)	16	16	16	16	16	16	17	18	16	16	00	3E
	(0F)	17	17	17	17	17	17	17	18	17	17	00	3F
	(10)	08	08	08	08	01	01	01	0E	01	01	0B	01
	(11)	00	00	00	00	00	00	00	00	00	00	00	00
	(12)	0F	0F	0F	0F	03	03	01	0F	0F	0F	05	0F
	(13)	00	00	00	00	00	00	00	08	00	00	00	00
Sequencer	(1)	0B	0B	01	01	0B	0B	01	00	0B	01	01	01
Registers	(2)	03	03	03	03	03	03	01	03	0F	0F	0F	0F
(3C5)	(3)	00	00	00	00	00	00	00	00	00	00	00	00
	(4)	03	03	03	03	02	02	06	03	06	06	06	06
Grapics	(0)	00	00	00	00	00	00	00	00	00	00	00	00

BIOS Mode		0	1	2	3	4	5	6	7	13	14	15	16
Controller	(1)	00	00	00	00	00	00	00	00	00	00	00	00
Registers	(2)	00	00	00	00	00	00	00	00	00	00	00	00
(3CF)	(3)	00	00	00	00	00	00	00	00	00	00	00	00
	(4)	00	00	00	00	00	00	00	00	00	00	00	00
	(5)	10	10	10	10	30	30	00	10	00	00	00	00
	(6)	0E	0E	0E	0E	0F	0F	0D	0A	05	05	05	05
	(7)	00	00	00	00	00	00	00	00	0F	0F	0F	0F
	(8)	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
CRTC	(0)	37	37	70	70	37	37	70	60	37	70	60	5B
Registers	(1)	27	27	4F	4F	27	27	4F	4F	27	4F	4F	4F
(3B5/3D5)	(2)	2D	2D	5C	5C	2D	2D	59	56	2D	59	56	53
	(3)	37	37	2F	2F	37	37	2D	3A	37	2D	3A	37
	(4)	31	31	5F	5F	30	30	5E	51	30	5E	50	52
	(5)	15	15	07	07	14	14	06	60	14	06	60	00
	(6)	04	04	04	04	04	04	04	70	04	04	70	6C
	(7)	11	11	11	11	11	11	11	11	11	11	1F	1F
	(8)	00	00	00	00	00	00	00	00	00	00	00	00
	(9)	07	07	07	07	01	01	01	0D	00	00	00	00
	(0A)	06	06	06	06	00	00	00	0B	00	00	00	00
	(0B)	07	07	07	07	00	00	00	0C	00	00	00	00
	(0C)	00	00	00	00	00	00	00	00	00	00	00	00
	(0D)	00	00	00	00	00	00	00	00	00	00	00	00
	(0E)	00	00	00	00	00	00	00	00	00	00	00	00
	(0F)	00	00	00	00	00	00	00	00	00	00	00	00
	(10)	E1	E1	E1	E1	E1	E1	E0	5E	E1	E0	5E	5E
	(11)	24	24	24	24	24	24	23	2E	24	23	2E	2B
	(12)	C7	C7	C7	C7	C7	C7	C7	5D	C7	C7	5D	5D
	(13)	14	14	28	28	14	14	28	28	14	28	28	28
	(14)	08	08	08	08	00	00	00	0D	00	00	0D	0F
	(15)	E0	E0	E0	E0	E0	E0	DF	5E	E0	DF	5E	5F
	(16)	F0	F0	F0	F0	F0	F0	EF	6E	F0	EF	6E	0A
	(17)	A3	A3	A3	A3	A2	A2	C2	A3	E3	E3	E3	E3
	(18)	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF

1.11.5 Color Graphics Adpater Compatible Registers.

When the PC1640 IGA is in 6845 compatible mode and attached to a PC-CD display (or to a PC-ECD display) it supports the older style 200-line mode text and graphics. Its CRT controller and memory mapping are are different from the EGA environment in a number of important ways and the software support environment is different in that the system ROS ROM supports the video interrupt 16 calls. The IGA ROM BIOS initializes the IGA for the CGA mode emulation and then sets the video int 16 vectors so that the standard ROS calls will be used for CGA mode support.

In CGA emulation mode there are two programmable registers for CGA mode and color selection. These consist of the CGA Mode Control Register and the CGA Color Select Register.

1.11.5.1 CGA Control Register

The CGA Mode Control Register is a write only register located at I/O address 3D8h. It is used to control the state of the video circuitry, selecting Alpha or Graphics mode and the various sub options available within Alpha and Graphics modes.

The layout of the CGA Mode control register is as follows:

Bit	Output Use
7	No effect
6	No effect
5	Enable Blinking Chars (disable intensified backgrounds)
4	Select Graphics Mode 2 (de-select graphics mode 1)
3	Enable Video Display

Bit	Output Use
2	Select Palette 2 (de-select palettes 0,1)
1	Select Graphics modes (de-select Alpha modes)
0	Select Alpha 80 Char mode (de-select 40 Char mode)

When bit 5 is set in Alpha modes, the foreground will blink for all displayed characters with attribute bit 7 also set. Bit 5 has no effect in Graphics modes.

Bit 4 (Select Graphics Mode 2) has no effect in Alpha modes.

Clearing bit 3 causes the display to be blanked if bit 2 in the IGA Extended Mode Control register is cleared. The standard CGA emulation mode initialization process sets bit 2 of the IGA Extended Mode control register so that blanking is disabled. The DISPLAY program has a command line parameter, CGAB, which allows the standard blanking function to be enabled.

The Select Palette 2 bit (bit 2) has no effect in Alpha modes or in Graphics mode 2. It is used in conjunction with bit 5 of the CGA colour select register to control graphics mode 1 palette. To select palette 2, bit 5 of the CGA colour select register (below) should be reset and bit 2 of theGA mode control register should then be set.

Bit 0 (Select Alpha 80 Char mode) has no effect in Graphics modes.

To avoid unsightly effects on the screen, this register should be updated during frame flyback time. Any kind of mode changing should preferably be done with video disabled. Mode changing involves the use of bits 1 and 0 and usually some re-programming of the CRTC.

1.11.5.2 CGA Color Select Register

The CGA Color Select Register is a write only register located at I/O address 03D9h and is used for controlling border colour in alpha modes and for selecting palette, border and pixel colour options in the graphics modes. The layout of the CGA Colour select register is as follows:

Bit	Alpha Modes	Graphics Mode 1	Graphics Mode 2
7,6	No Effect	No Effect	No effect
5	No Effect	Select Palette 1 (Deselect palette 0)	No effect
4	No Effect	Foreground Intensity for palettes 0, 1 & 2	No effect
3	Intensity (Border)	Intensity (Backgnd and Border)	Intensity (Pixel)
2	Red (Border)	Red (Background and Border)	Red (Pixel)
1	Green (Border)	Green (Background and Border)	Green (Pixel)
0	Blue (Border)	Blue (Background and Border)	Blue (Pixel)

In 640 x 200 two-color graphics mode the overall screen palette is controlled by setting the CGA Color Select register (which is initialized to 07h by the ROS on selection of mode 6). This means that the colour resolution is really one of 16 colours for foreground on a black background. Most applications software however control 640 x 200 mode graphics as black and white graphics.

To avoid unsightly effects on the screen this register should only be updated during frame flyback time.

1.11.5.3 CGA Status Register

The CGA Status Register is a read only register located at I/O address 03DA. It may be read at any time to determine the following:

Bit	Input Use
7 - 4	Not used.
3	Frame Flyback (VSYNC) Time.
2	Light-pen switch open.
1	Light-pen latch set.
0	Display Enabled.

Frame flyback time starts at the same time as the bottom border and lasts for 46 horizontal scan periods, ending 16 scans before the end of the subsequent top border.

Bit 2 reflects the state of the light-pen push button switch. Bit 2 = 0 when the light-pen switch is closed and bit 2 = 1 when the light-pen switch is open.

When bit 1 is set, it indicates that the light pen latch is set, triggered either by a pulse from the light pen or by writing data to set the

light pen channel. Writing any data to the Clear Light Pen channel clears the latch.

Bit 0 = zero when the Video signal is enabled and bit 0 = "1" when either the vertical or horizontal retrace signals are active. This is a real time indication of the raster scan line status.

1.11.5.4 CGA Mode 6845 CRTC emulation.

The CRT Controller is an emulation of a MC6845 CRT Controller device. All 6845 registers are programmable in the AMSTRAD PC1640 and function the same as the actual device.

The registers must be programmed according to the CGA mode of operation required in conjunction with the CGA Mode and Color Select Registers previously described. A mode changing operation should be performed in the following sequence: Disable video, reprogram the CRTC as required, reprogram the Mode and Color select registers as required, (maintaining video disabled), initialize display ram as required, enable video.

The 6845 CRTC is controlled by way of two I/O addresses, the CRTC Address register and the CRTC Data I/O location. The CRTC Address register is a write only register located at I/O address 3D4 and the lower 5 bits are used select the data register at I/O address 3D5. Addresses greater than 17 (11h) produce no results.

The CRTC registers are initialized as follows:

Register Number (Hex)	Register Name	Alpha 40 Char Mode (Decimal)	Alpha 80 Char Mode (Decimal)	Graphics Modes (Decimal)	R/W Type
00	Horizontal Total (-1)	56	113	56	WO
01	Horizontal Displayed	40	80	40	WO
02	Horiz. Sync Position	45	90	45	WO
03	Horiz. Sync Width	10	10	10	WO
04	Vertical Total	31	31	127	WO
05	Vertical Total Adjust	6	6	6	WO
06	Vertical Displayed	25	25	100	WO
07	Vertical Sync Position	28	28	112	WO
08	Interlace	2	2	2	WO
09	Max. Scan Address	7	7	1	WO
0A	Cursor Start	6	6	6	WO
0B	Cursor End	7	7	7	WO
0C	Start Address High	0	0	0	WO
0D	Start Address Low	0	0	0	WO
0E	Cursor Location High	0	0	0	RW
0F	Cursor Location Low	0	0	0	RW
10	Light Pen Posn. High	-	-	-	RO
11	Light Pen Posn. Low	-	-	-	RO

Values greater than 31 in register 0Ah turn the cursor off. This is because bit 5 is the cursor off bit. Bit 6 in register 0Ah selects alternate blink rate.

CRTC Display Addressing

In Alpha modes, the CRTC register values for start address and light pen position are in the 8K range 0000h to 1FFFh. The register value corresponding to a character position in display RAM must be derived from the even byte address in the 16K range B8000h to BBFFFh by subtracting B8000h and halving.

In Graphics modes, the CRTC register values for start address and light pen position are in the 4K range 0000h to 0FFFh, and wraparound occurs above this range. A register value corresponds to two pairs of pixel bytes in display RAM on word boundaries, one pair displayed on an even scan and the other pair displayed on the following odd scan in the same horizontal position.

The register corresponding to the pixel byte pair position in display RAM must be derived from the even byte address in the 8K range B8000h to B9FFFh (for an even scan) by subtracting the address offset B8000h and halving. Similarly for the odd scan line the offset BA000h is subtracted from an even byte address in the range BA000h to BBFFFh and halved.

1.11.6 Monochrome Display Adapter Compatible Registers.

When the PC1640 IGA is in 6845 compatible mode and attached to a PC-MD display it supports the monochrome mode text. Its CRT controller and memory mapping are are different from the EGA environment in a number of ways and the software support environment is different in that the system ROS ROM supports the older VIDEO Int 16 calls. The IGA ROM BIOS initializes the IGA for the MDA mode emulation and then sets the video int 16 vectors so that the standard ROS calls will be used for MDA mode support.

1.11.6.1 MDA Mode Control Register

The MDA Mode Control Register is a write only register located at I/O address 3B8h. It is used to control the state of the video circuitry.

The format of the MDA Mode control register is as follows:

Bit	Output Use
7	No effect
6	No effect
5	Enable Blinking Chars (disable intensified background)
4	No effect
3	Enable Video Display
2	No effect
1	No effect
0	No effect

When bit 5 is set in the foreground will blink for all displayed characters with the blink (7) bit set in their attribute bytes.

Bit 3 must be set in order to enable the video output and when bit 3 is zero the display is blanked.

1.11.6.2 MDA Status Register

The MDA Status Register is a read only register located at I/O address 03BA. It may be read at any time to determine the following:

Bit	Input Use
7 - 4	Not used (0).
3	Video.
2 - 1	Not used (0).
0	Display Enabled.

Bit 0 = zero when the Video signal is enabled and bit 0 = one when either the vertical or horizontal retrace is active.

Bit 3 = one when the mono video output pin (7) is active. This is a real time status of the raster scan line output.

1.11.6.3 6845 CRTC Emulation.

The CRT Controller is an emulation of a MC6845 CRT Controller device. All MC6845 are programmable in the AMSTRAD PC1640 and function the same as the actual device.

The CRTC is controlled by way of two I/O addresses, the CRTC Address register and the CRTC Data I/O location. The CRTC Address register is a write only register located at I/O address 3B0 and the lower 5 bits are used select the data register at I/O address 3B1. Addresses greater than 17 (11h) produce no results.

The CRTC registers names and initial values are as follows:

Register Number (Hex)	Register Name	Initial Value (Decimal)	R/W Type Type
00	Horizontal Total (-1)	97	WO
01	Horizontal Displayed	80	WO
02	Horiz. Sync Position	82	WO
03	Horiz. Sync Width	15	WO
04	Vertical Total (-1)	25	WO
05	Vertical Total Adjust	6	WO
06	Vertical Displayed	25	WO
07	Vertical Sync Position	25	WO

Register Number (Hex)	Register Name	Initial Value (Decimal)	R/W Type Type
08	Interlace	2	WO
09	Maximum Scan	13	WO
0A	Cursor Start	11	WO
0B	Cursor End	12	WO
0C	Start Address High	0	WO
0D	Start Address Low	0	WO
0E	Cursor Location High	0	RW
0F	Cursor Location Low	0	RW
10	Light Pen Address High	-	RO
11	Light Pen Address Low	-	RO

Setting bit 5 in register 0Ah turns the cursor off. Bit 6 in 0Ah selects alternate blink rate.

1.11.7 Hercules Compatible Emulation.

For the most part, the Hercules Monochrome (HMGA) emulation mode resembles the MDA emulation in text modes and the control/status register bits have the same meanings. There are, however extra functions available in the HMGA emulation which must be considered.

1.11.7.1 HMGA Mode Control Register

The HMGA Mode Control Register is a write only register located at I/O address 3B8h. It is used to control the state of the video circuitry.

The format of the MDA Mode control register is as follows:

Bit	Output Use
7	Display Page Select.
6	No effect.
5	Enable Blinking Chars (disable intensified background)
4	No effect.
3	Enable Video.
2	No effect.
1	Text/Graphics Mode Select.
0	No effect.

Bit 7 = '0' selects display page 0 (B0000-B7FFF), which is the Power-Up default state. Bit 7 = '1' selects display page 1 (B8000 - BFFFF). Note that this will overlay an alternate color mode (CGA) adapter's display buffer which ranges from B8000 to BBFFF. This bit is disabled from being set to '1' by the Hercules Mode register (bit 1) at I/O address 3BFh. See page 30.

When bit 5 is set in the foreground will blink for all displayed characters with the blink (7) bit set in their attribute bytes.

Bit 3 must be set in order to enable the video output and when bit 3 is zero the display is blanked.

Bit 1 = '0' selects text mode, which is the Power-Up default state. Bit 1 = '1' selects graphics mode. This bit is disabled from being set to '1' by the Hercules Mode register (bit 0) at I/O address 3BFh. See page 30.

In Text mode display buffer spans from B8000 to B0FFF and the PC1640 ROS supports 4 display pages in 4K increments from the origin. See 1.11.2.5 - BIOS Mode 7. The character/attribute pairs are as described under Monochrome Text description.

In Graphics Mode there two 32K display pages as controlled by bit 7 above. The byte/pixel mapping is segmeted into four groups of scan lines in the display buffer with a x and y range of 0 to 719 and 0 to 347 respectively. The following 'C' language program illustrates the display mapping.

```
/*--------------------------* HERCMAP.C *-----------------------------------*/

#include <stdio.h>

int HGR_Offset(x,y)
int x,y;
```

```
{
        return (0x2000 * (y % 4)) + (90 * (y/4)) + (x/8);
}

main()
{
        int x,y;

        for (x = y = 0; y < 10; y++)
                printf(1x=%d, y=%3d: Offset = %4xh\r\n1,x,y,HGR_Offset(x,y));
        for (y = 338; y < 348; y++)
                printf(1x=%d, y=%3d: Offset = %4xh\r\n1,x,y,HGR_Offset(x,y));
}
/*--------------------------* HERCMAP.C *-----------------------------------*/
```

And this program produces the following output:

```
        x = 0, y =   0: Offset = 0000h
        x = 0, y =   1: Offset = 2000h
        x = 0, y =   2: Offset = 4000h
        x = 0, y =   3: Offset = 6000h
        x = 0, y =   4: Offset = 005Ah
        x = 0, y =   5: Offset = 205Ah
        x = 0, y =   6: Offset = 405Ah
        x = 0, y =   7: Offset = 605Ah
        x = 0, y =   8: Offset = 00B4h
        x = 0, y =   9: Offset = 20B4h
        x = 0, y = 338: Offset = 5D88h
        x = 0, y = 339: Offset = 7D88h
        x = 0, y = 340: Offset = 1DE2h
        x = 0, y = 341: Offset = 3DE2h
        x = 0, y = 342: Offset = 5DE2h
        x = 0, y = 343: Offset = 7DE2h
        x = 0, y = 344: Offset = 1E3Ch
        x = 0, y = 345: Offset = 3E3Ch
        x = 0, y = 346: Offset = 5E3Ch
        x = 0, y = 347: Offset = 7E3Ch
```

This output illustrates the first 10 display lines and the last 10 display lines in the display buffer. Each 720 pixel display line occupies 90 (5Ah) bytes. And every fourth line starts at the next contiguous 90th byte position. The four groupings are separated by 2000h in the buffer (as the first four output lines (0-3) illustrate. The final line in each grouping ends at a byte offset of 1E95h from its group origin. The (016Ah byte) void space after each line is not used.

Note the '+ (x/8)' term in the equation for HGR Offset. It represents the fact that there are eight pixels per byte and they're numbered from 0 to 7 which is inverse to the bit numbering. The equation for the pixel's bit number within a byte would then be: Bit Number = (7 - (x Mod 8))

1.11.7.2 HMGA Status Register

The HMGA Status Register is a read only register located at I/O address 03BA. It may be read at any time to determine the following:

Bit	Input Use
7	-Vertical SYNC.
6 - 4	Not used (0).
3	-Mono Video.
2 - 1	Not used (0).
0	Display Enabled.

Bit 7 = '0' indicates that the vertical SYNC signal is active and the display is blanked. Bit 7 = '1' indicates that the display is active. It is a good idea to wait until bit 7 is clear before updating the screen.

Bit 3 = '0' when the mono video output pin (7) is active. This is a real time status of the raster scan line output.

Bit 0 = '0' when the Video signal is enabled and bit 0 = '1' when either the vertical or horizontal retrace signals are active (and the display is blanked).

1.11.7.3 6845 CRTC Emulation.

The CRT Controller is an emulation of a MC6845 CRT Controller device. All MC6845 are programmable in the AMSTRAD PC1640 and function the same as the actual device.

The CRTC is controlled by way of two I/O addresses, the CRTC Address register and the CRTC Data I/O location. The CRTC Address register is a write only register located at I/O address 3B0 and the lower 5 bits are used select the data register at I/O address 3B1. Addresses greater than 17 (11h) produce no results.

The CRTC registers names and initial values are as follows:

Register Number (Hex)	Register Name	Alpha (80) Char Mode (Decimal)	Graphics Mode (Decimal)	R/W Type
00	Horizontal Total (-1)	97	53	WO
01	Horizontal Displayed	80	45	WO
02	Horiz. Sync Position	82	46	WO
03	Horiz. Sync Width	15	7	WO
04	Vertical Total (-1)	25	91	WO
05	Vertical Total Adjust	6	2	WO
06	Vertical Displayed	25	87	WO
07	Vertical Sync Position	25	87	WO
08	Interlace	2	2	WO
09	Maximum Scan	13	3	WO
0A	Cursor Start	11	0	WO
0B	Cursor End	12	0	WO
0C	Start Address High	0	0	WO
0D	Start Address Low	0	0	WO
0E	Cursor Location High	0	0	RW
0F	Cursor Location Low	0	0	RW
10	Light Pen Position High	-	-	RO
11	Light Pen Position Low	-	-	RO

In text mode, setting bit 5 in register 0Ah turns the cursor off. Bit 6 in 0Ah selects alternate blink rate.

The PC1640 ROS ROM only supports text mode (7) and contains no setup tables for HMGA graphics. The values given are therefore a typical setup and represent a setup for graphics page zero. Consult the MC6845 CRTC data sheets for additional device details to derive alternate setups.

1.12 Floppy Disk Controller

The floppy disk controller is based on the NEC uPD765A single chip controller, and supports one or two 5.25 inch single or double sided, MFM double density floppy disk drives with a data rate of 250 kilobits per second.

The FDC is controlled by way of the Drive Selection register at I/O Address 3F2. It is defined as follows:

Bit (Dn)	Output Use
7 - 6	No effect
5	Switch motor(s) on and enable drive 1 selection
4	Switch motor(s) on and enable drive 0 selection
3	Allow 765A FDC to interrupt and request DMA
2	- 765A reset
1	Drive Select Bit 1 (DS1)
0	Drive Select Bit 0 (DS0)

The Drive Select bits (DS1, DS0) are only valid for values of 00 and 01 for drives 0 and 1 respectively. The drive selection qualification is only completed when either bit 4 (for drive 0) or bit 5 (for drive 1) is set. In addition setting bits 4 or 5 will have no effect until the value of DS1, DS0 is correspondingly set.

Bit 2 (- 765A reset) must brought low (output as '0') and held low for at least 3.5 uS in order to reset the 765A. It must then be set high in order to release the reset signal.

On power-up or following a system reset, all bits in this register are cleared to zero.

1.12.1 FDC Hardware Conditions

The hardware imposes the following conditions on the use of the 765A controller and disk drives:

1. The clock frequency of the 765A FDC is fixed at 4.0 MHz.
2. Disk data transfers are done by DMA using the on-board DMA controller. The 765A DRQ output may connected to or disconnected from the DMA controller DRQ2 input by software using Drive Selection Register bit 3.
3. An interrupt level is available for use by the 765A to signal command completion and attention status to the CPU. The 765A INT output may be connected to or disconnected from the interrupt controller IRQ6 input by software using Drive Selection Register bit 3.
4. Drive 0 is always present. Drive 1 is optional. Drives 2 and 3 are not implemented and can never be accessed. (No select signal decode is implemented for these drives.) Drive Ready output signal from the currently selected drive is connected to the 765A RDY input. For drives which do not a drive ready output the 765A RDY input may be optionally fixed to the true condition.
5. The 765A Drive Select outputs US1 and US0 are not used to select the drives. This function together with motor control is done via the Drive Selection Register which is external to the FDC 765A.
6. The FLT (Fault) input 765A is forced permanently false.
7. A Two-Sided status signal from the drive(s) is not provided but interface to the drives allows the use of double sided drives.
8. Write precompensation of 250 ns is provided.
9. The 765A may be individually reset by software using Drive Selection Register bit 2.

1.13 RS232C Asynchronous Serial Port

The asynchronous serial port is configured to I/O addresses 3F8h - 3FFh and is based on the National INS8250 ACE (or UART), single channel device.

The clock frequency input of the 8250 is 1.8432 MHz (± 0.1%).

The 8250 BAUD OUT output is connected to the RCLK input.

An interrupt level is available for use by the 8250. When the 8250 OUT2 output is driven low (i.e. a '1' is written to bit 2 of the 8250 MODEM Control Register) then the INTRPT signal is connected to the interrupt control IRQ4 input.

1.13.1 Serial Channel Interface

The serial interface uses a 25-way subminiature D type plug (male) connector emulating a DTE (Data Terminal Equipment).

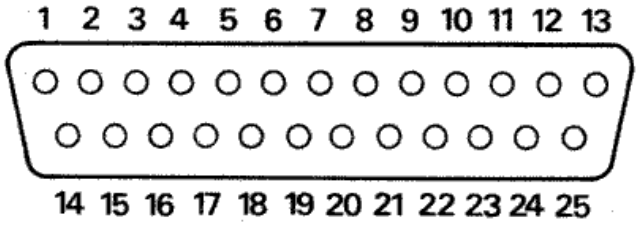
The electrical levels of signal lines on this interface conform with EIA (Electronics Industry Association) standard RS-232C (and the equivalent CCITT V.24 interface standard).

The RS232C drivers and receivers between the 8250 and the serial channel connector are all inverting.

1.13.2 Serial Channel Pin Arrangement

Pin	EIA	CCITT	Description
2	BA	103	TxD - Serial Data Output
3	BB	104	RxD - Serial Data Input
* 4	CA	105	RTS - Request to Send Output
5	CB	106	CTS - Clear to Send Output
6	CC	107	DSR - Data Set Ready Input
7	AB	102	Signal Ground (Common Return)
8	CF	109	DCD - Data Carrier Detect Input
* 20	CD	108.2	DTR - Data Terminal Ready Output
22	DE	125	RI - Ring Indicator Input

* These interchange circuits, where implemented, shall be used to detect either a power off condition in the equipment across the interface, or the disconnection of the interconnecting cable. The terminator for these circuits shall interpret the power off condition or the disconnection of the interconnecting cable as an OFF condition.



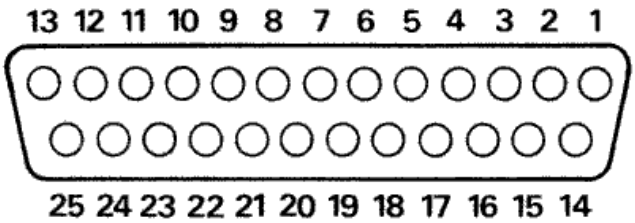
See Appendix 4 for additional details of serial signals and cable connections.

1.14 Parallel Printer Interface

The parallel printer port is described in Section 1.10 and is a general purpose 'Centronics' style 8-bit interface. The printer interface uses a 25-way subminiature 'D' socket (female) connector located at the back of the PC1640.

The Pin assignments for the printer connector is as follows:

Pin	Assignment
1	Not Data Strobe
2	Data Bit 0
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
10	Not Printer Acknowledge
11	Printer Busy
12	Paper Out
13	Select Printer
14	Not Select Auto Feed
15	Not Printer Error
16	Not Reset Printer
17	Not Printer Selected
18	GND
19	GND
20	GND
21	GND
22	GND
23	GND
24	GND
25	GND



Appendix 5 contains the Amstrad PL-2 printer lead specification for the DMP3000 printer.

1.15 Keyboard Interface

Keyboard data input to the CPU is via the 8255 PPI Port A, and the keyboard interrupt (level 1) of the 8259A PIC. Both of these have been previously described in sections 1.6 and 1.8.

1.15.1 Serial Clock and Serial Data

The Serial Clock and Serial Data signals are used for keyboard interface. These two bidirectional signals are used by the keyboard microcontroller to send keycodes to the main electronics board. The main electronics board also uses the same two signals to indicate readiness to receive another keycode back to the microcontroller. In addition these two signals are used to reset the microcontroller under hardware or software control.

1.15.2 Keyboard to Main Board Interface

The quiescent state for both Serial Clock and Serial Data is high. A minimum of 5 us must separate a transition of one signal from another transition of the same signal, or any transition of the other signal.

Keycodes are sent from the keyboard microcontroller to the main board in 8-bit serial form MS bit first. Keycode data received by the main board is clocked into a shift register as either a "1" bit sequence or as a "0" bit sequence. To be interpreted as a "1" bit, the Serial Data signal must remain high during the time period when Serial Clock goes low and returns to the high state. To be interpreted as a "0" bit, Serial Data must be low prior to the Serial Clock transition from high to low, Serial Data will then go high followed by Serial Clock. The "1" bit or the "0" bit is clocked into the shift register on the falling edge of Serial Clock.

1.15.3 Main Board to Keyboard Interface

Upon receiving a keystroke from the microcontroller, within 5 us of the last clock falling edge, the main board electronics drives the Serial Data line low and maintains it low until it is ready to receive a new keystroke. When the main board returns the Serial Data signal to the high state the microcontroller is free to send another keystroke. This response to the reception of a keycode is termed the ACKNOWLEDGE sequence.

The mainboard electronics causes a RESET to the keyboard microcontroller by driving the Serial Clock line low for 10 milliseconds or more. The state of the Serial Data signal does not affect the reset sequence.

1.15.4 Keycodes

The 8-bit keyboard data is capable of 128 'make' codes correspondingly 128 'break' codes. For any key which is pressed, the 'make' keycode produced is in the range of 0 - 127 decimal. When a key is released, the 'break' keycode produced is the same as the make keycode except that the top bit is set so that the value is in the range of 128 - 511 (decimal).

After a key is pressed and the keycode has been sent to the main board electronics, if no new keys are pressed and the key has remained pressed for more that one half second, then the keyboard microcontroller re-sends the keycode every 83 milliseconds provided that the main board indicates by an Acknowledge sequence that is ready to accept a new keycode. The Pause/Break key does not repeat.

The keycode 0AAh is sent after a reset to indicate successful completion of power-up tests.

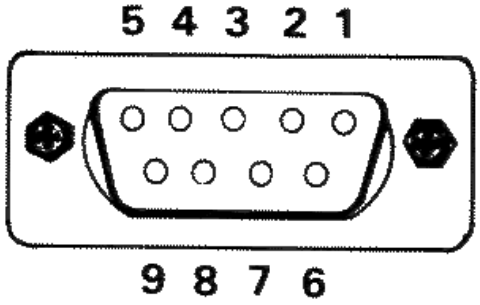
The PC1640 ROM BIOS receives the keycodes via an interrupt subroutine and it 'tokenizes' them into a two-bytes value for further conversion to ascii values by applications software which use the tokenized keyboard information. The keycodes and their corresponding token values are covered in the ROS firmware (Section 2.3.5). Programs such as KEYBUK.EXE replace the ROM BIOS keyboard vectors and perform their own tokenization process. Since certain of the keycodes are actually Mouse Button and Joystick information, non AMSTRAD versions of KEYBUK will ignore these 'extraneous' keycodes resulting in a loss of a certain level of functionality in these areas.

1.15.5 Keyboard Connector

The Keyboard connector is a 6-way Din socket. The pin assignment is as follows:

Pin	Assignment
1	KBCLK
2	KBDATA
3	M1 (Mouse Button 1)
4	GND
5	+5 Volts DC
6	M2 (Mouse Button 2)

Viewed from left hand side of machine



The M1 and M2 signals are connected directly to the keyboard controller in order to produce keycodes.

1.16 Mouse Interface

The mouse interface consists of two switch inputs from push buttons and two 8-bit X & Y coordinate counters. The two mouse switches (M1 & M2) are arranged to form part of the keyboard matrix and are handled as keyboard data (producing low level keycodes 7E and 7D respectively).

The Mouse X-Coordinate (I/O Address 078h) is an 8-bit counter which can be read by the CPU. Any write access regardless of the value written to the X-Coordinate location clears the counter. Similarly the Mouse Y-Coordinate (I/O Address 07Ah) can be read by the CPU or cleared by any write access to its I/O address.

The counters are incremented or decremented according to the direction of movement of the mouse, and their values indicate the mouse movement since last read or cleared. The X-Coordinate counter increments for "Right" motion and decrements for "Left" motion. The Y-Coordinate counter increments for "UP" motion and decrements for "Down" motion. In order to properly track mouse motion, software should read and clear the coordinate counters at a rate high enough to prevent overflow from positive values to negative values or negfative values to positive values for a fairly high rate of mouse movement. The scaling of mouse movement is such that one increment of the counter represents 1/8 mm of physical mouse motion.

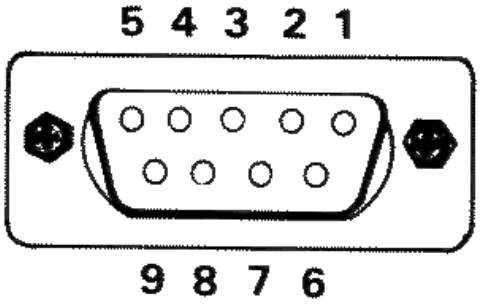
The delivered operating systems have AMSTRAD specific mouse drivers which actively perform the Read-and-Clear operation (every 18 ms) using the ticker interrupt. This can cause the appearance of no mouse motion to the casual observer sampling the mouse coordinate counters. See Appendix 1 for additional details concerning Mouse Software Interfaces.

1.16.1 Mouse Connector

The mouse connector is a 9 way D type (female) connector located on the left hand side of the case and it has an AMSTRAD specific pinout. Attaching any other manufacturer's hardware (even though the connector may be similar) to the PC1640 mouse connector may cause serious damage to either the main board electronics or to the alternative (mouse) hardware.

The mouse connector pin assignments are as follows:

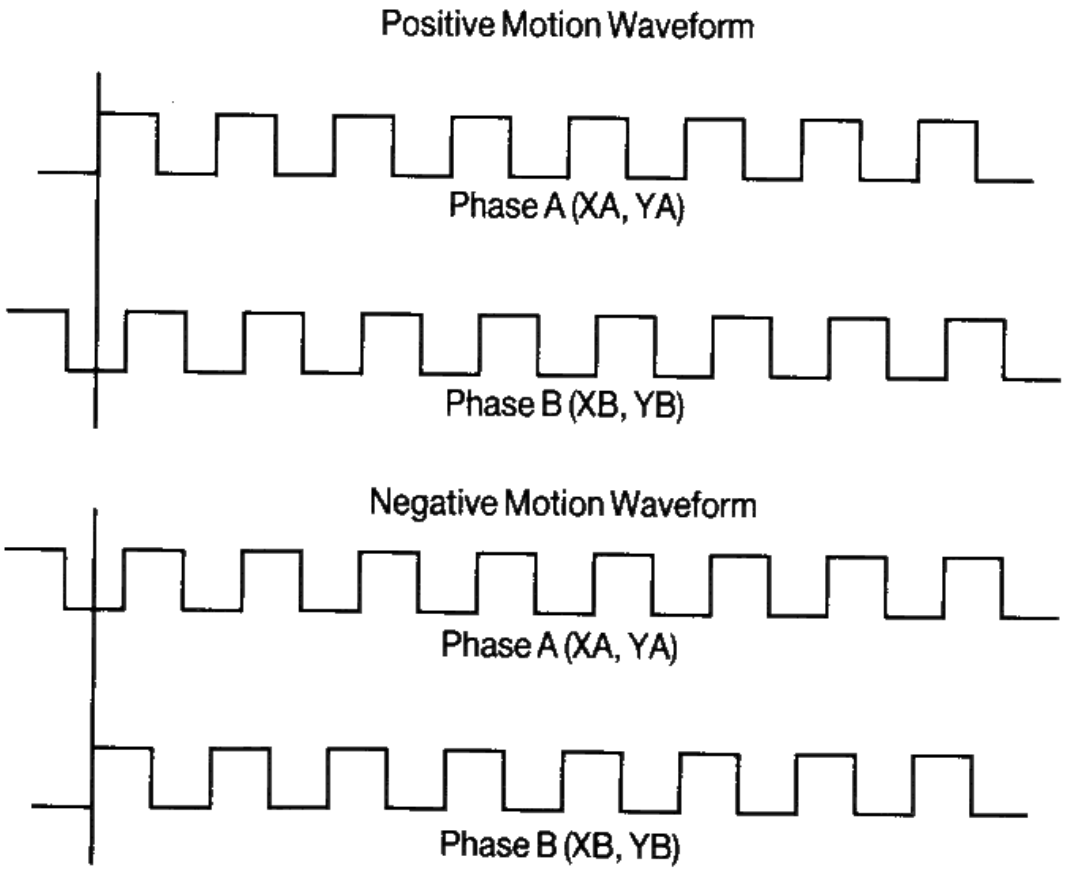
Pin	Assignment
1	XA
2	XB
3	YA
4	YB
5	Spare
6	M1
7	+5 Volts DC
8	GROUND
9	M2



mouse connector

The first four pins contain optically encoded phase XA, XB, YA and YB square waves. For positive motion the square wave on the A phase leads the B phase by 90 degrees with the reverse being true for negative motion.

The remaining pins carry Mouse Button 1 (M1), 5V power, Ground and Mouse Button 2 (M2) signals.



1.17 Joystick Interface

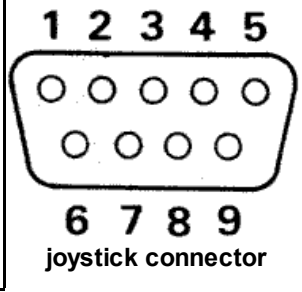
The AMSTRAD PC supports an industry standard joystick interface. The joystick inputs are handled as keycodes from the keyboard interface. The low level keycodes are in the range of 7C down to 77 (hexadecimal) corresponding to Up, Down, Left, Right, Fire1 and Fire2 respectively. The ROS firmware (See section 2) translates the directional codes to cursor key tokens and the Fire buttons can be assigned variable tokens depending on NVR settings.

1.17.1 Joystick Connector

The joystick connector is a 9 way D type (male) connector with an industry standard pinout. Attaching an incorrect device (even though the connector may be similar) to the PC1640 joystick connector may cause serious damage to either the main board electronics or to the incorrect (joystick) hardware.

The Joystick Socket is located on the rear left corner of the keyboard. Its pinout is as follows:

Pin	Assignment
1	Up
2	Down
3	Left
4	Right
5	Spare
6	Fire 2
7	Fire 1
8	Common
9	Not Connected



joystick connector

Viewed from rear of keyboard

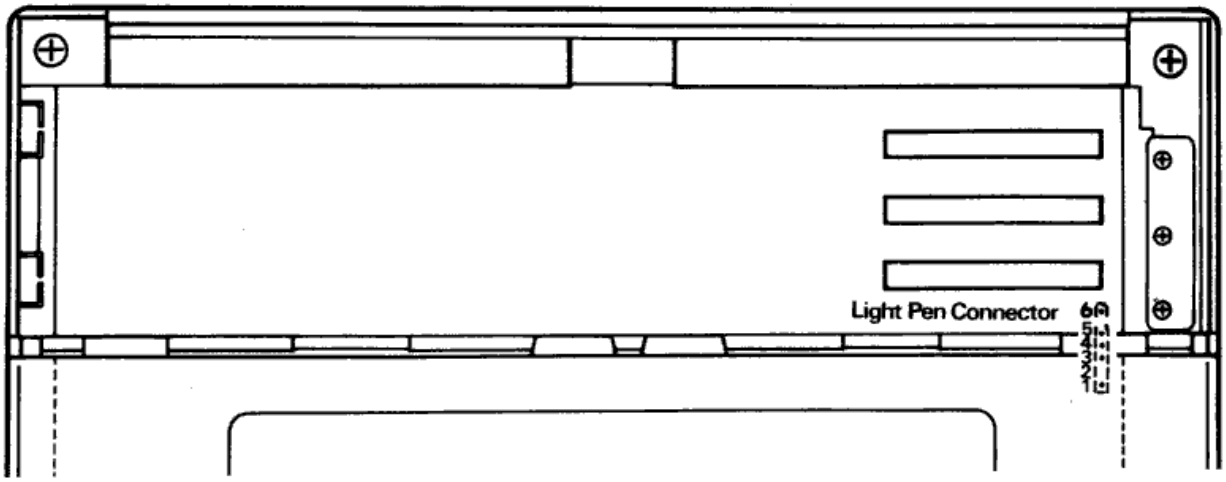
1.18 Light Pen Connector

The AMSTRAD PC1640 Supports a standard light pen interface via the emulated MC6845. The Light Pen connector is located by

removing the expansion slot cover at the rear of the machine. The connector is located inside the PC case on the right hand edge of the main board just forward of the expansion card connectors. It consists of a 6-way berg strip and is labeled PL8 (LIGHT PEN) in large letters. Pin 1 is the forward most pin viewed from in front of the machine (the disk drive end).

The pin assignment is as follows:

Pin	Assignment
1	-Light Pen Input.
2	(Keyway)
3	-Light Pen Switch.
4	Ground.
5	+5 Volts DC.
6	+12 Volts DC.

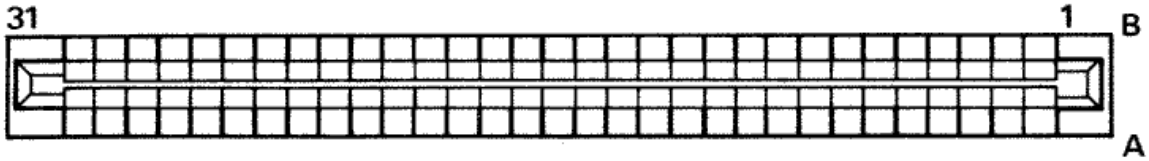


1.19 Expansion Card Interface

The AMSTRAD PC1512 has three slots for additional peripheral cards. These consist to a set of connectors in the right rear of the main board. The Pin numbering of the each connector is the same and is such that the left (ground plane) side is numbered B1 - B31 top to bottom and the right (component) side is numbered A1 - A31 top to bottom. The following table defines the pin assignments of the expansion interface:

Pin	Signal	In/Out
A01	Not I/O CHCK	In
A02	I/O Data Bit D7	In/Out
A03	I/O Data Bit D6	In/Out
A04	I/O Data Bit D5	In/Out
A05	I/O Data Bit D4	In/Out
A06	I/O Data Bit D3	In/Out
A07	I/O Data Bit D2	In/Out
A08	I/O Data Bit D1	In/Out
A09	I/O Data Bit D0	In/Out
A10	I/O RDY	In
A11	AEN - Address Enable	Out
A12	I/O + Mem/Address Bit A19	Out
A13	I/O + Mem/Address Bit A18	Out
A14	I/O + Mem/Address Bit A17	Out
A15	I/O + Mem/Address Bit A16	Out
A16	I/O + Mem/Address Bit A15	Out
A17	I/O + Mem/Address Bit A14	Out
A18	I/O + Mem/Address Bit A13	Out
A19	I/O + Mem/Address Bit A12	Out
A20	I/O + Mem/Address Bit A11	Out

Pin	Signal	In/Out
A21	I/O + Mem/Address Bit A10	Out
A22	I/O + Mem/Address Bit A09	Out
A23	I/O + Mem/Address Bit A08	Out
A24	I/O + Mem/Address Bit A07	Out
A25	I/O + Mem/Address Bit A06	Out
A26	I/O + Mem/Address Bit A05	Out
A27	I/O + Mem/Address Bit A04	Out
A28	I/O + Mem/Address Bit A03	Out
A29	I/O + Mem/Address Bit A02	Out
A30	I/O + Mem/Address Bit A01	Out
A31	I/O + Mem/Address Bit A00	Out
B01	Ground	--
B02	RESET	Out
B03	+ 5 Volts DC	--
B04	IRQ2	In
B05	- 5 Volts DC	--
B06	DREQ2	In
B07	- 12 Volts DC	--
B08	Not Connected (Reserved)	In
B09	+ 12 Volts DC	--
B10	Ground	--
B11	-MEW (Memory Write)	Out
B12	-MRD (Memory Read)	Out
B13	-IOW (I/O Write)	Out
B14	-IOR (I/O Read)	Out
B15	-DACK3	Out
B16	DREQ3	In
B17	-DACK1	Out
B18	DREQ1	In
B19	-DACK0	Out
B20	CLK	Out
B21	IRQ7	In
B22	IRQ6	In
B23	IRQ5	In
B24	IRQ4	In
B25	IRQ3	In
B26	-DACK2	Out
B27	T/C	Out
B28	ALE	Out
B29	+ 5 Volts DC	--
B30	CK14	Out
B31	Ground	--



Expansion Bus Connector
Viewed from above while standing in front of machine

The I/O expansion slots are laid out the same as the industry standard 16-bit Personal Computer bus. The translation from the internal 16-bit 8086 bus to the 8-bit I/O bus layout is done by main board circuitry. Any 16-bit CPU I/O transfers will be broken down into two 8-bit cycles (with wait states) by this circuitry.

All signals are TTL compatible and can support a maximum of two low-power schottky (LSTTL) loads per slot. Power supply loading per slot should be limited to a maximum of 700 milliamperes on the + 5 Volt supply, to 100 milliamperes on the - 5 Volt supply, to

700 milliamperes on the + 12 Volt supply and to 100 milliamperes on the - 12 Volt supply.

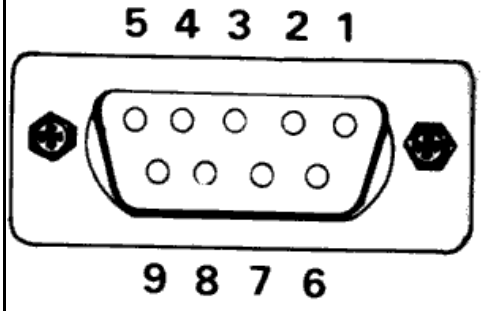
Note that direct access to the on-board 16-bit fast memory bus is not available via the I/O expansion slots.

Additional engineering details for prototyping adapter boards should be supplied as part of the documentation for that particular hardware.

1.20 Video Connector

The video connector is a 9-way D type (female) socket located in the rear of the computer. Its pinout is as follows:

Pin	Assignment
1	Ground
2	Secondary Red (r) or Ground (GND)
3	Primary Red (R)
4	Primary Green (G)
5	Primary Blue (B)
6	Secondary Green (g) or Intensity (I)
7	Secondary Blue (b) or Mono Video (V)
8	Horizontal SYNC
9	Vertical SYNC



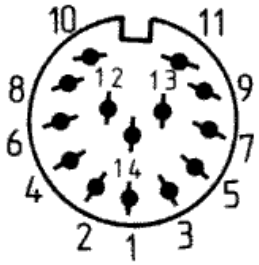
Video Connector

The same pinout is used for all three display types but the interpretation of a particular pin's function varies with the connected device. In addition certain devices require that pin 2 exhibit a full ground characteristic and this is accomplished by switch 8 (sw8) at the rear of the PC1640. When sw8 is in the OFF position pin2 is grounded. When sw8 is in the ON position then the secondary red (r) signal is routed through to pin 2 (See section 1.22 - PC1640 Switch Settings.)

1.21 Power Connector

The power connector is a 14-way Din socket located in the rear of the computer. Power is routed from the power supply located in the monitor to the main board electronics through the power connector. Its pinout is as follows:

Pin	Assignment
1	Not Connected
2	0 Volts DC
3	+ 5 Volts DC
4	0 Volts DC
5	+ 5 Volts DC
6	Not Connected
7	Not Connected
8	0 Volts DC
9	- 12 Volts DC
10	0 Volts DC
11	+ 12 Volts DC
12	0 Volts DC
13	- 5 Volts DC
14	Not Connected



Power Connector

1.22 PC1640 Display Selector Switch Settings.

The Internal Graphics Adapter initial mode switches are as follows:

sw4	sw3	sw2	sw1	Primary Adapter			Secondary Adapter		
OFF	ON	OFF	OFF	MD	(Mono)	Internal	CGA	(CO80)	External
OFF	ON	OFF	ON	MD	(Mono)	"	CGA	(CO40)	"
OFF	ON	ON	OFF	ECD350	(CO80)	"	MDA/HERC	(Mono)	"
OFF	ON	ON	ON	* ECD200	(CO80)	"	MDA/HERC	"	"
ON	OFF	OFF	OFF	* CD	(CO80)	"	MDA/HERC	"	"

sw4	sw3	sw2	sw1	Primary Adapter			Secondary Adapter		
ON	OFF	OFF	ON	CD	(CO40)	"	MDA/HERC	"	"
ON	OFF	ON	OFF	CGA	(CO80)	External	MD	(Mono)	Internal
ON	OFF	ON	OFF	CGA	(CO40)	"	MD	(Mono)	"
ON	ON	OFF	OFF	MDA/HERC	(Mono)	"	ECD350	(CO80)	"
ON	ON	OFF	ON	MDA/HERC	"	"	* ECD200	(CO80)	"
ON	ON	OFF	ON	MDA/HERC	"	"	* CD	CO80	"
ON	ON	ON	ON	MDA/HERC	"	"	CD	(CO40)	"

* See note 3 below.

sw5:
OFF for true EGA mode.
ON for CGA/MDA/HERC emulation modes.
(Also called 6845 compatible modes.)
sw8:
OFF for CD (Standard RGB) Monitors.
ON for ECD (Enhanced RGB) Monitors.

The IGA foreign font switches are as follows:

sw10	sw9	sw7	sw6	
OFF	OFF	--	--	English Font
OFF	ON	ON	ON	English Font
OFF	ON	OFF	OFF	Danish Font
OFF	ON	OFF	ON	Portugese Font
OFF	ON	ON	OFF	Greek Font

Switches sw6, sw7 and sw10 are used by the IGA BIOS ROM to select an alternate set of characters to be loaded into the IGA fonts RAM for characters in the range of 128 to 255.

Switch sw9 (when set to 'OFF') has the effect of switching off the foreign fonts section of the IGA ROM (in the address range C4000h - C7FFFh). This forces the IGA BIOS to select the english fonts which are stored below address C4000. Switching sw9 to OFF also enables any controller device using this address range to place its controller ROM in the C4000 to C7FFF address space and it will not conflict with the foreign fonts section of the IGA ROM.

The IGA can be switched off by turning sw10 to the 'ON' position. When this is done, switches sw1 to sw5, sw8 and sw9 have no effect and in addition, switches sw6 and sw7 assume a different function in that they are used by the PC1640 ROS for setting the default display mode (DDM) as follows:

sw10	sw9	sw7	sw6	
ON	--	OFF	OFF	External EGA installed.
ON	--	OFF	ON	External CGA in 40 Column Mode
ON	--	ON	OFF	External CGA in 80 Column Mode
ON	--	ON	ON	External Monochrome Adapter (MDA or Herc)

Notes:

1. During bootstrap all messages will be sent to the primary adapter.
2. In order to switch text output to the secondary adapter the MODE utility is required. The sw1 - sw4 table gives the required MODE parameter enclosed in parenthesis for text output to the device.
3. The ECD350 and ECD200 switch settings specify whether the default text display will be in high resolution (350 line) mode or in low resolution (200 line) mode. ECD200 appears the same as CD (CO80) in the default text mode. However graphics software may examine the switch settings to determine if 350 line mode is achievable. An incorrect switch setting may cause software to switch the display setup to 350 lines and if a PC-ECD display is not truly fitted, incorrect synchronization will be output. While this may not actually damage the (PC-CD) display the screen will continuously roll and the system may have to be bootstrapped to recover.
4. Secondary Adapter(s) listed in the sw1 - sw4 table section are optional where they are noted as being external and the respective switch settings merely inform software such as the ROM BIOS and MODE.EXE what the desired setup characteristics will be if the device is sensed as being installed. When the primary adapter is listed as an external device then it is required and system initialization will fail if the device is not installed.

2.0 Firmware

This section describes the Amstrad PC1640 Resident Operating System (ROS) and the Internal Graphics Adapter (IGA) ROM BIOS. It defines the interfaces to all the interrupt service routines provided by the Amstrad PC1640 ROS firmware (ROM) and all RAM locations used by the ROS.

The following copyright message is stored at the beginning of the ROS starting at location 0003 (relative to its origin at FC000):

(C) Copyright 1987 AMSTRAD plc

The ROS physically occupies the highest 16K bytes (in the address range FC000 to FFFFF) in the 1 Mega Byte addressing range of the 8086-2 CPU (See Figure 1.1). The total 64K byte address range from hexadecimal F0000 to FFFFF is reserved for system ROM and this contains the reset and initialization address FFFF0. The PC1640 address decoding for the system ROM area is such that the ROS ROM which actually resides at FC000 is repeated four times in the address space starting at F0000.

Note that all address constants in this document are in hexadecimal form unless otherwise noted.

All calls to the ROS and IGA BIOS firmware should be made through the software interrupts disclosed in this manual. Application programs must not attempt to access the locations within the ROM area directly. Amstrad reserves the right to modify the coding within the Resident Operating System ROM and the IGA BIOS ROM as it sees fit.

The firmware provides a set of resident software routines which perform various services and I/O functions:

1. Power-Up initialization and Self Test.
2. Keyboard input.
3. Video display of characters and pixels.
4. Video buffer screen dumping.
5. Character I/O to the printer and serial ports.
6. System clock and real time clock support.
7. Floppy Disk I/O including format, read and write.

To ensure hardware independence of application programs all I/O processes should be done using the ROM firmware software interrupts. This avoids possible problems due to any hardware modifications and/or enhancements.

2.1 Power-Up initialization and Self Test

The Power-Up initialization and Self Test function is entered at location FFFF0, the CPU reset entry point. This is a collection of routines which perform all necessary hardware initialization and self tests, setting up of the BIOS RAM variable area, initialization all the interrupt locations used by the ROS, initialization of expansion slot peripheral ROMs and the running of floppy diskette or hard disk bootstrap.

The ROS does not use the System RAM (User RAM Area) for stack or program variables until it has been successfully tested. If a RAM error is found an error message should be displayed correctly, assuming there is no other fault that may result in incorrect operation of the CPU or Video Circuitry.

The Power-Up initialization and Self Test process proceeds as follows:

1. Disable maskable and non-maskable interrupts.
2. Run self test which include the following:
   Checksum of the ROS.
   Test all system RAM.
   8237 DMA controller.
   8253 Programmable Interval Timer.
   8255 Programmable Peripheral Interface.
   RTC counting.
   The system serial interface.
   The system printer interface.
   8259 programmable interrupt controller.
   Mouse X and Y counter registers.

   After a system reset all the self tests except the RAM tests are rerun. If the three option links in the least significant part of the system printer status register are set to all ones then diagnostic mode is selected and if the diagnostic test ROM pack is found, it is entered, otherwise only the keyboard and disk tests are run before external ROM initialization and system startup is attempted.

   Refer to section 2.2 for the individual power-up self test details.

3. Checksum the NVR.

If the checksum of the NVR is incorrect then it is loaded with its default values (See page 169).

4. Initialise the 8253 Programmable Interval Timer.

Set up counter 0 to interrupt every 54.9337 milliseconds. Set up counter 1 to generate an output signal with a period of 15.13 microseconds. Disable counter 2.

5. Initialise the 8237 DMA controller.

Set up DMA channel 0 for memory refresh. Disable channel 1, 2 and 6.

6. Initialise the 8259 Programmable Interrupt Controller.

Disable (mask) all interrupt levels. Note that levels 0, 1 and 3 are enabled (unmasked) later.

7. Initialise the Write Status Registers.

Write Status-1 is initialised from a byte in the NVR defining the number of drives fitted and the default Video Mode. The ROS also sets or resets bit 1 in the status register depending on whether or not an 8087 NDP is installed. See section 1.8 for further System Status-1 information.

Write Status-2 is initialised according to the amount of memory installed. The ROS assumes a

minimum of 512K bytes and that additional RAM may be added in contiguous 32k byte increments up to the maximum of 640K bytes. The additional memory is sized according to the following procedure. The segment address of each of the four 32K byte RAM blocks is written to the first two bytes of each respective block. The segments are then verified from low to high until a non matching segment address or the last block is encountered. The setting of the Write Status-2 register is according to the RAM0-RAM4 table in section 1.8.3.

8. Initialise the ROS variable area in system RAM.

The ROS uses variables in the address range of 00300 to 00500. Refer to section 2.4 (RAM Variables) for a complete description of these variables and their respective initialised values.

9. Initialise the first 32 Interrupt Vectors.

The first 32 interrupt vectors are set up to reference the ROS routines as listed below. Software interrupt routines which do not perform any function reference a dummy routine that simply does a return from interrupt (IRET) instruction. Hardware service interrupts which do not perform any function reference a dummy (HWIRET) routine which issues a nonspecific end-of-interrupt to the 8259 interrupt controller and then executes an IRET instruction.

Interrupt	Purpose	Type
0	Divide by Zero	Hardware (HWIRET)
1	Single Step	Hardware (HWIRET)
2	Parity error routine (NMI)	Hardware
3	Break	Hardware (HWIRET)
4	Overflow	Hardware (HWIRET)
5	Print Screen	Software
6	Reserved	Software
7	Reserved	Software
8	System Clock interrupt	Hardware
9	Keyboard interrupt	Hardware
10	RTC interrupt	Hardware (HWIRET)
11	COMMS	Hardware (HWIRET)
12	COMMS	Hardware (HWIRET)
13	Hard Disk	Hardware (HWIRET)
14	Floppy Disk interrupt routine	Hardware
15	Printer interrupt	Hardware (HWIRET)
16	Video I/O	Software
17	System Configuration	Software
18	Memory Size	Software
19	Disk I/O	Software
20	Serial I/O	Software
21	Enhanced function(s)	Software
22	Keyboard I/O	Software
23	Printer I/O	Software
24	System Restart	Software
25	Disk Bootstrap	Software

Interrupt	Purpose	Type
26	System Clock and RTC I/O	Software
27	Keyboard Break	Software (IRET)
28	External Ticker interrupt	Software (IRET)
29	Video parameter table	Software Vector
30	Disk Parameter table	Software Vector
31	External 8x8 Char Matrix table	Software Vector

The interfaces to the above routines are detailed in section 2.3.

The 8259 PIC is programmed such that its IRQ0 - IRQ7 interrupt levels use CPU interrupt vectors 8 - 15 as indicated in the above table.

10. Initialise and Test the Disk interface.

The initialise function of interrupt 19 in invoked followed by the disk test (see 2.2.12).

11. Keyboard Self Test.

The Keyboard microcontroller returns 0AAh upon successful completion of its power-up self test (See 2.2.15).

12. Initialise the Video Display.

The initial video testing is done by IGA BIOS ROM if it is enabled else the ROS performs testing as described in section 1.8.2

13. Initialise the 8259 Interrupt controller.

Enable 8259 interrupt controller on levels 0 (8253 counter 0), 1 (keyboard scan code receiver) and 6 (765 floppy disk controller). All other 8259 interrupt levels are masked.

14. Display the ROS sign-on message.

During power-up the ROS checksums the NVR. After the sign-on message has been displayed, the ROS outputs a warning message if the NVR sum was incorrect. In the case that the NVR is OK (and last startup data are valid) the time and date of last switch-on are displayed.

15. Enable the NMI.

If a NMI occurs the default ROS interrupt handler displays a RAM parity error message and hangs the system. This condition can only be rectified by switching the machine off.

16. Initialise all external ROMs.

The ROS checks for external ROMs between addresses C0000 and F4000 in 800h (2k) byte increments. An external ROM which conforms to the following specification will be initialised by the ROS:

1. The first two bytes contain the hexadecimal value 55AA.
2. The next two bytes contain the size in 512 (1/2K) byte increments.
3. The next byte is the initialization routine entry point.
4. The LS byte of the byte sum of the ROM is zero.

When a ROM conforming to this specification is located then the initialization entry is called.

If the checksum test fails then an error message is displayed and initialization is not called.

The IGA BIOS ROM conforms to above standard and if it is enabled, it is entered (at C0000) and its Power-On Self Test (POST) initialisation will perform testing and initialise:

256k of Video RAM.
Video Controller.
Load Fonts from ROM based tables.
Establish initial (default) display mode based on switch settings.
Set up Video Interrupt 16 based on display mode switch settings.

17. Diskette Bootstrap.

The ROS attempts to load the bootstrap sector (from drive A, side 0, track 0, sector 1) into memory at 07C00h. If the bootstrap sector loads successfully it is given control (far jump to segment 0000 offset 7C00). If after 10 retrys the bootstrap sector cannot be loaded then the ROS displays a message prompting the user to insert a system disk into drive A and press a key. The ROS then waits for the key press and repeats the bootstrap procedure.

2.2 Power-Up Self Tests

On Power-Up or following a system reset, the ROS performs a series of self tests on the hardware to verify proper operation. When a test failure occurs, the ROS displays an error message on the primary display device and the system is locked up. The keyboard interface is treated differently in that the ROS repeats keyboard self test until it is successful.

The ROS executes all self tests except when the option links (LK1 - LK3) are all set (See section 1.10.3), the ROS will only run the keyboard interface test and the disk test. If either of these two tests fail an error message is displayed but the error is ignored. This allows the system to be brought up for diagnostic testing.

When a soft reset (Control, Alt and Del) is issued the ROS performs all the self tests except the system RAM (User Area RAM) test and the video RAM test. The program calling the ROS initialization after a soft reset must store the value 1234h in system location 0:478h and this value notifies the ROS firmware that a soft reset was requested. When the 1234h value is recognized a full hardware reset is performed and this will reset all external peripheral cards in the expansion bus. A special entry flag value, 1235h performs the ROS Power-Up testing as described above but inhibits the full hardware reset from occurring. The ROS sets the contents of the word location at 0:488 to 1234h when it completes its testing.

2.2.1 Test Procedure.

Upon entry the ROS performs the necessary functions to start video output from the IGA or any other primary display adapter, and then stores the message ("Please wait") on the first line of the screen to

indicate that self testing is in progress and as each successive self test is started a dot is displayed on the screen.

The tests are run in the following order:

1. ROS checksum test.
2. Direct Memory Access (8237) Controller test.
3. Programmable Interval timer (8253) test.
4. Programmable Peripheral Interface (8255) test.
5. Real Time Clock (HD146818) test.
6. Asynchronous Communications Element (8250) test.
7. Parallel Printer Port test.
8. Mouse X and Y count register test.
9. System RAM test.
10. Programmable Interrupt Controller (8259) test.
11. Disk test.
12. Keyboard Interface test.

The ROS uses the stack during the Disk test, the Keyboard interface test and the Programmable Interrupt Controller test. All other self tests are executed without using either the stack or any RAM variables.

2.2.2 Test Methods.

Most of the device diagnostic tests consist of a Data Path test and a Waveform test as described below:

Data Path test.

> The data path test checks the read/write path between the CPU and a particular device. A pattern is written to a device and then read back to verify the integrity of the data path. The patterns are as follows:
>
> All zeros.
> All ones.
> Sliding single bit and complement across 8 bits.

Waveform test.

> The waveform test detects address decoding errors in a hardware device. The waveform test consists of selecting a specific address in a device, writing a test pattern (usually 0FFFFh) and verifying that the same pattern can be read back. The waveform test is done in both ascending sequential order (upwards) and descending sequential order (downwards) in order to check that the address decoding logic works correctly.

2.2.3 ROS Checksum Test.

All bytes in the Resident Operating System ROM are summed and then checked that the least significant byte of the sum is zero. If the check fails then an error message indicating faulty ROM checksum is displayed.

2.2.4 Direct Memory Access Controller test.

The upwards/downwards waveform test is used to confirm that the registers in the DMA controller chip can be addressed. Any failure will cause the faulty DMA error message to be displayed.

2.2.5 Programmable Interval Timer test.

The fiest 8253 test is a read/write data path test to counter 2 followed by a check that counter 1 counts at the correct rate. If either test fails an interval timer error message is displayed.

2.2.6 Programmable Peripheral Interface test.

The 8255 PPI tests consists of a data path test on each of the two system status channels (Status-1 and Status-2). The 8253 PIT OUT2 (Status-2) bit is also checked for proper operation. If either test fails the faulty real time clock error message is displayed.

2.2.7 Real Time Clock test.

The RTC seconds counter is tested to be counting at the correct rate. Next a data path test on the checksum byte of the NVR is run (and the checksum byte is restored). If either test fails the faulty real time clock error message is displayed.

2.2.8 Asyncronous Communications Element test.

This test confirms that the transmitter and receiver of the 8250 (i. e. the system serial port) are functioning correctly (at least in diagnostic mode).

The 8250 is configured in loop mode, 9600 baud, 8 data bits, 1 stop bit and no parity. Two test patterns are transmitted and the received patterns are checked. The status register is monitored for no parity, framing or overrun errors. If either received pattern does not equal the sent pattern or an error is set in the status register the faulty system serial port error message is displayed.

2.2.9 Printer Parallel Port test.

A data path test is performed on the printer data latch. If any incorrect test pattern is returned, the faulty printer port error message is displayed.

2.2.10 Mouse X and Y Count Register test.

The X and Y registers are cleared and then read to verify that they both contain zero. If the test fails the faulty mouse coordinate register error message is displayed.

2.2.11 System RAM test.

The amount of System (User Area) RAM is determined using the procedure described in section 2.1 (This should always produce 640K as the answer). The data path test is run on all available RAM followed by an upwards/downwards waveform test. If either test fails the faulty RAM error message is displayed.

2.2.12 Programmable Interrupt Controller test.

The 8259 tests consist of a data path test on the interrupt mask register and an interrupt acknowledge

test to confirm that interrupts can occur and be serviced. If the test fails the faulty interrupt controller message is displayed.

2.2.13 Disk test.

The disk test attempts to establish whether the drives fitted to the system seek correctly. The test moves the read/write heads to track 10 on each drive. The ROS does not verify that the correct track was attained. If any errors are reported then a floppy disk controller error message is displayed.

2.2.14 Keyboard Interface test.

Upon power-up or reset, the keyboard self test is performed by the keyboard controller firmware. The keyboard returns keycode 0AAh to signify the successful completion of its testing. If any key code other than 0AAh is returned the keyboard error message is displayed amd keyboard reset is issued (which reruns the keyboard self test). The Keyboard test is repeated until the keyboard test passes. When test pass is received, the error message is removed from the screen and the test is exited as normal. During the keyboard test a short beep is sounded every five seconds to indicate that the test is in progress.

2.3 ROM Firmware Interrupts.

The first 32 interrupt vectors are initialised by Power-Up initialization. The software IRET and hardware HWIRET entries are dummy routines which require no entry or exit conditions and are not detailed here.

Any application program which replaces a default interrupt vector with its own entry point must not invoke any ROS interrupts from within its own interrupt service routine.

2.3.1 Interrupt 2: Parity Error (NMI).

The Interrupt 2 routine deals with system RAM parity error. The screen is switched to the default display mode, cleared and a RAM parity error message is displayed. The machine cannot be used until the power switch is cycled off and on again.

This routine does not use RAM for stack or program variables.

An application program which makes use of the 8087 NDP must supply an interrupt 2 service routine for the 8087 NDP.

CPU registers are used as follows:

Entry:
    No conditions.
Exit:
    Doesn't exist.

2.3.2 Interrupt 5: Print Screen.

The Interrupt 5 routine dumps the screen in character mode to the primary printer port. Since the screen dump is character based, attempting to dump graphic pictures to the printer may produce incorrect results. Characters that cannot be read back from the screen in graphics mode (using the video

interrupt 16 read character function) are printed as spaces.

If a screen print is already in progress the interrupt takes no action.

The Print Screen Status variable (at address 00500) is set to 1 while the screen dump is in progress. When complete the variable is set to zero. If the screen dump is abandoned due to printer port timeout, the variable is set to 255.

CPU registers are used as follows:

Entry:
    No conditions.
Exit:
    All flags and registers preserved.

2.3.3 Interrupt 6: Mouse Button Control.

The ROS interrupt 6 routine provides default mouse button services. The keyboard firmware generates a set of make/break keycodes when either of the two mouse buttons is pressed and released. When the keyboard interrupt routine recognizes a mosue button keycode it invokes interrupt 6. The default ROS routine will either obtain the appropriate keycode from the NVR and return with the carry flag set in the case of a make code or return with the carry flag clear in response to a mouse button break code.

CPU registers are used as follows:

Entry:
    Register AL = Mouse Key Code
    Bit 0 specifies which mouse button:
        0 = Mouse Button M1.
        1 = Mouse Button M2.
    Bit 7 specifies whether make or break:
        0 = Mouse Button make.
        1 = Mouse Button break.
Exit:
    Carry Flag & Register Ax specify action:
    Carry SET:
        Insert a key token into the keyboard buffer.
        AX = Key Token to be inserted.
    Carry CLEAR:
        No action to be taken.
        AX is corrupt.
    All other flags corrupt. (also BX, CX & DX may be corrupt.)
    All other registers preserved.

Note: A key token value of FFFFh is ignored and is not put in the keyboard buffer.

2.3.4 Interrupt 8: System Clock Interrupt.

The interrupt 8 routine is invoked by the system clock (counter 0 of the 8253). The default ROS routine does the following:

1. Increment the 32-bit system clock count held in RAM (location 0046C). If the clock reaches the 24 hour time (0001855000h) then the count is reset to zero and the 24 hour flag (location 00470) is set to 0FFh.
2. If the least significant byte of the system clock count is zero then the current time and date in the real time clock (RTC) is copied to the NVR. The time that is last copied from the RTC before the machine is switched off is displayed when the machine is next switched on.
3. If the disk motor timeout count is not zero then it is decremented by one. If the count reaches zero all the drive motors are turned off.
4. Invoke interrupt 28. Application programs that want to be interrupted by the system clock should use interrupt 28.

CPU registers are used as follows:

Entry:
No conditions.
Exit:
All flags and registers preserved.

2.3.5 Interrupt 9: Keyboard Interrupt.

The ROS Keyboard hardware interrupt reads a key code from the keyboard interface, translates the key code into a 16-bit key token using an internal translation table and the key token is put into the key token buffer. If the buffer is full the key token is discarded and a bleep is output on the speaker. The key tokenization for the most part consists of the high byte being the key number and the lower byte being the ASCII for the keycap. Those keys for which there is no ASCII equivalent the token consists of a unique high byte value with the low byte cleared.

Entry:
No conditions.
Exit:
All flags and registers preserved.

The ROS Keycode translation table is as follows (all values are hexadecimal) and the key names are the USA versions:

Key Code	Key Cap (US)	Normal	Num Lock	ALT	CTRL	SHIFT
01	ESC	011B	01F0	011B	011B	N/A
02	1 and !	0231	7800	Ignored	0221	N/A
03	2 and @	0332	7900	0300	0340	N/A
04	3 and #	0433	7A00	Ignored	0423	N/A
05	4 and $	0534	7B00	Ignored	0524	N/A
06	5 and %	0635	7C00	Ignored	0625	N/A
07	6 and ^	0736	7D00	071E	075E	N/A
08	7 and &	0837	7E00	Ignored	0826	N/A
09	8 and *	0938	7F00	Ignored	092A	N/A
OA	9 and (	0A39	8000	Ignored	0A28	N/A
0B	0 and)	0B30	8100	Ignored	0B29	N/A
0C	- and _	0C2D	8200	0C1F	0C5F	N/A
0D	= and +	0D3D	8300	Ignored	0D2B	N/A

Key Code	Key Cap (US)	Normal	Num Lock	ALT	CTRL	SHIFT
0E	<-Del	0E08	0EF0	0E7F	0E08	N/A
0F	Tab	0F09	A500	9400	0F00	N/A
10	Q	1071	1000	1011	1051	N/A
11	W	1177	1100	1117	1157	N/A
12	E	1265	1200	1205	1245	N/A
13	R	1372	1300	1312	1352	N/A
14	T	1474	1400	1414	1454	N/A
15	Y	1579	1500	1519	1559	N/A
16	U	1675	1600	1615	1655	N/A
17	I	1769	17000	1709	1749	N/A
18	O	186F	1800	180F	184F	N/A
19	P	1970	1900	1910	1950	N/A
1A	[and {	1A5B	1AF0	1A1B	1A7B	N/A
1B	] and }	1B5D	1BF0	1B1D	1B7D	N/A
1C	Return Key	1C0D	1CF0	1C0A	1C0D	N/A
1D	Ctrl	Ignored	Ignored	----	Ignored	N/A
1E	A	1E61	1E00	1E01	1E41	N/A
1F	S	1F73	1F00	1F13	1F53	N/A
20	D	2064	2000	2004	2044	N/A
21	F	2166	2100	2106	2146	N/A
22	G	2267	2200	2207	2247	N/A
23	H	2368	2300	2308	2348	N/A
24	J	246A	2400	240A	244A	N/A
25	K	256B	2500	250B	254B	N/A
26	L	266C	2600	260C	264C	N/A
27	; and :	273B	27F0	Ignored	273A	N/A
28	' and "	2827	28F0	Ignored	2822	N/A
29	# and ~	2960	29F0	Ignored	297E	N/A
2A	Left Shift	Ignored	Ignored	Ignored	Ignored	N/A
2B	\ and |	2B5C	2BF0	2B1C	2B7C	N/A
2C	Z	2C7A	2C00	2C1A	2C5A	N/A
2D	X	2D78	2D00	2D18	2D58	N/A
2E	C	2E63	2E00	2E03	2E43	N/A
2F	V	2F76	2F00	2F16	2F56	N/A
30	B	3062	3000	3002	3042	N/A
31	N	316E	3100	310E	314E	N/A
32	M	326D	3200	320D	324D	N/A
33	, and <	332C	33F0	Ignored	333C	N/A
34	. and >	342E	34F0	Ignored	343E	N/A
35	/ and ?	352F	35F0	Ignored	353F	N/A
36	Right Shift	Ignored	Ignored	Ignored	----	N/A
* 37	*	372A	37F0	9600	372A	N/A
38	Alt	Ignored	----	Ignored	Ignored	N/A
39	Space Bar	3920	3920	3920	3920	N/A

Key Code	Key Cap (US)	Normal	Num Lock	ALT	CTRL	SHIFT
* 3A	Caps Lock	Ignored	Ignored	Ignored	Ignored	N/A
3B	F1	3B00	6800	5E00	5400	N/A
3C	F2	3C00	6900	5F00	5500	N/A
3D	F3	3D00	6A00	6000	5600	N/A
3E	F4	3E00	6B00	6100	5700	N/A
3F	F5	3F00	6C00	6200	5800	N/A
40	F6	4000	6D00	6300	5900	N/A
41	F7	4100	6E00	6400	5A00	N/A
42	F8	4200	6F00	6500	5B00	N/A
43	F9	4300	7000	6600	5C00	N/A
44	F10	4400	7100	6700	5D00	N/A
* 45	Num Lock	Ignored	Ignored	PAUSE	Ignored	N/A
* 46	Scroll Lock	Ignored	Ignored	Ignored	Ignored	N/A
47	Key Pad 7	4700	Ignored	7700	N/A	4737
48	Key Pad 8	4800	Ignored	Ignored	N/A	4838
49	Key Pad 9	4900	Ignored	8400	N/A	4939
4A	Key Pad -	4A2D	Ignored	Ignored	N/A	4A2D
4B	Key Pad 4	4B00	Ignored	7300	N/A	4B34
4C	Key Pad 5	Ignored	Ignored	Ignored	N/A	4C35
4D	Key Pad 6	4D00	Ignored	7400	N/A	4D36
4E	Key Pad +	4E2B	Ignored	Ignored	N/A	4E2B
4F	Key Pad 1	4F00	Ignored	7500	N/A	4F31
50	Key Pad 2	5000	Ignored	Ignored	N/A	5032
51	Key Pad 3	5100	Ignored	7600	N/A	5133
* 52	Key Pad 0	5200	Ignored	Ignored	N/A	5230
53	Key Pad .	5300	Ignored	Ignored	N/A	532E
54 - 6F	Undefined	Ignored	Ignored	Ignored	Ignored	Ignored
* 70	Del ->	N/A	N/A	N/A	N/A	N/A
71 - 73	Undefined	Ignored	Ignored	Ignored	Ignored	Ignored
* 74	Enter	N/A	N/A	N/A	N/A	N/A
75 - 76	Undefined	Ignored	Ignored	Ignored	Ignored	Ignored
* 77	Joy Fire2	N/A	N/A	N/A	N/A	N/A
* 78	Joy Fire1	N/A	N/A	N/A	N/A	N/A
* 79	Joy Right	4D00	4D00	4D00	4D00	4D00
* 7A	Joy Left	4B00	4B00	4B00	4B00	4B00
* 7B	Joy Down	5000	5000	5000	5000	5000
* 7C	Joy Up	4800	4800	4800	4800	4800
* 7D	Mouse M2	N/A	N/A	N/A	N/A	N/A
* 7E	Mouse M1	N/A	N/A	N/A	N/A	N/A
7F	Undefined	Ignored	Ignored	Ignored	Ignored	Ignored

Joystick keys produce their respective cursor keys.

Key codes marked with '*' cause special actions as explained below.

The table positions marked 'Ignored' are physically marked in the table by a value with the MS bit set and this causes the keyboard processor to ignore these keystroke combinations.

2.3.5.1 Special Key Actions.

Some keys or set of keys invoke a special action as detailed below. Unless otherwise stated they do not result in any key tokens being inserted into the buffer.

1. [Ctrl]+[Alt]+[Del]: Reset.
   When reset is detected, a system hardware reset is issued. The power-up initialisation process is entered but System RAM and video RAM tests are not run.
2. [Ctrl]+[Nun Lock]: Pause.
   The ROS waits for another key to be pressed (except [Ctrl]+[Num Lock]), thus suspending any application that is running. Since the keyboard firmware lights the light on the Num Lock key the ROS toggles the Num Lock flag so that the system ROS and the keyboard firmware are in step with each other. Note that programs such as KEYBUK do not perform this extra function so there may be some confusion in these cases.
3. [Ctrl]+[Scroll Lock]: Break.
   When break is detected, interrupt 27 is invoked and the keyboard buffer is cleared. Key token 0000h is then inserted into the buffer.
4. [Shift] + [PrtSc]: Print Screen.
   When print screeen is detected interrupt 5 is invoked, the ROS print screen function. If this entry point is maintained screen dump will continue to work. Some operating systems such as CP/M and Dos Plus install a nul pointer here.
5. [Ins]: Insert Toggle.
   Each time the Ins key (keycode 52) is received, except in Num Lock mode, the Ins key toggle bit (bit 7 of RAM location 0:417) is inverted.
6. [Scroll Lock]: Scroll Toggle.
   Each time the Scroll Lock key is pressed the scroll key toggle bit (bit 4 of RAM location 0:417) is inverted. Note that Ctrl - Scroll Lock (break) does not flip the scroll toggle.
7. [Caps Lock]: Caps Lock Toggle.
   Each time the Caps Lock key is pressed the Caps Lock toggle bit (bit 6 of RAM location 0:417) is inverted.
8. [Num Lock]: Num Lock Toggle.
   Each time the NUM LOCK key is pressed the Num Lock toggle bit (bit 5 of RAM location 0:417) is inverted.
9. [Alt]+[Numeric Key Pad 0 to 9]: Absolute Key Token.
   When the [Alt] key is held down, an absolute key token may be entered via the numeric keypad. Pressing any other key resets the absolute key token to zero (and inserts the associated Alt-key token for the key pressed). When [ALT] is released the absolute key token modulo 256 is placed into the keyboard buffer, unless the token is zero, in which case it is discarded.
10. [Enter] and Fire Buttons.
    When the Enter key code (74) or one of the two Joystick Fire button key codes (78 or 79) is received an associated key token is obtained from the NVR and inserted into the key token buffer.
11. Mouse buttons.
    The keyboard firmware generates four key codes to indicate when the two mouse buttons are pressed or released. When the ROS receives one of these codes from the keyboard it does a far call to the address held in the Mouse Button interrupt vector (Interrupt 6). A default ROS routine is loaded into this vector upon power-up or system reset. This routine requests the ROS to insert a key token held in the NVR into the keyboard buffer, the token used depends on which mouse

button (M1 or M2) is received. The mouse button release codes are ignored.

12. [Del ->]: Forward Delete.
    When the forward delete key code (code 70) is received a key token is obtained from the NVR and placed in the key token buffer.

13 [Alt], [Ctrl], [Shift], [Caps Lock] & [Num Lock].

    The translation of various key codes into their respective tokens is affected by the current states of these keys (which is stored in location 00417). The Shift key, while pressed, reverses the current state of the CAPS LOCK and Num Lock. If more than one of Alt, Ctrl, Shift, Num Lock or CAPS LOCK is active at one time then the order of precedence for key code translation is Alt, then Ctrl, then Shift, then Caps Lock or Num Lock.

    Caps Lock, when active, converts the key tokens for the lower case alphabetic keys (a - z) to their upper case counterparts.

Note that some operating systems (such as DOS Plus) install their own entry points into the interrupt vectors and these interrupt routines may exhibit different characteristics than those of the ROS routines described here.

2.3.6 Interrupt 14: Floppy Disk Controller.

The ROS service routine for interrupt 14 sets bit 7 of the RAM DRIVE RESTORE FLAG, to indicate that the Floppy Disk Controller interrupt has occurred.

CPU registers are used as follows:

Entry:
    No conditions.
Exit:
    All flags and registers preserved.

2.3.7 Interrupt 16: '6845 Compatible' Video I/O (int 10h).

The ROS interrupt 16 service routine provides a set of routines for reading and writing characters in alpha and graphics mode. In graphics mode the characters are constructed using a character matrix table (see section 2.3.21). It also provides facilities for scrolling the screen up or down, reading and writing pixels (CGA graphics only) and reading the light pen.

It is important to note that this interrupt entry point is only active in '6845 Compatible' modes or when an external CGA or MDA or Hercules display adapter is active and the IGA ROM BIOS is not involved. The EGA emulation mode compatible calls are covered in section 2.3.8. It should be noted that the Int 16 call registers are much the same for the video mode calls and video character and string display. This is because the EGA compatible mode which was implemented later is designed to be software compatible with the earlier (this entry point's) call register setups. When the '6845 compatible' display modes are entered either as a result of the sw5 setting (ON) or by the DISPLAY utility program, the video int 16 vector (at 0:40h) is set to point to the ROS's video interrupt's fixed entry point (at F000:F065) and this set of service routines becomes active. Additional hardware setups must also be performed within the IGA to support the 6845 compatibility.

CPU registers for '6845 Compatible' Video Int 16 are used as follows:

Entry:

AH = Function Selector as follows:

0 - Set Video Mode.
1 - Set Cursor Size.
2 - Set Cursor Address.
3 - Get Cursor Address.
4 - Get Light Pen Address.
5 - Set Display Page.
6 - Scroll Screen Up.
7 - Scroll Screen Down.
8 - Read Character and Attributes.
9 - Write Character and Attributes.
10 - Write Character only.
11 - Write Color Select Register.
12 - Write Pixel.
13 - Read Pixel.
14 - Write Character in Teletype Emulation mode.
15 - Get Video Parameters.

All other registers as required by the function.

Exit:

If selector is greater than 15 then carry is set, else carry is clear.

All other flags and registers as specified by the function.

Alpha modes 0 and 1 require 2000 bytes of video RAM while alpha modes 2 and 3 require 4000 bytes of the video RAM. The ROS takes advantage of all the video RAM available in alpha modes by supporting multiple display pages. This means that application programs can set up a number of display pages and switch them as required.

In general parameters passed to ROS routines are not checked and care should be taken when choosing unusual parameters as unexpected results may occur. In particular be careful of boundary conditions such as setting the top of the display window equal to the bottom of the display window (for functions 6 and 7) effectively creating a one line display. In this instance the screen scrolling may not perform as expected.

ROS Video Int 16 Function 0: Set Video Mode.

CPU registers are used as follows:

Entry:

AH = 0
AL = Mode Selector as follows:

0 - Alpha 25 Rows by 40 Columns.
1 - Same as Mode 0.
2 - Alpha 25 Rows by 80 Columns.
3 - Same as mode 2.
4 - Graphics 200 pixels by 320 pixels using palette 1.

5 - Graphics 200 pixels by 320 pixels using palette 2.
6 - Graphics 200 pixels by 640 pixels - 2 colors.
7 - Alpha 25 Rows by 80 Columns using Monochrome Adapter.

Exit:

All flags and registers preserved.

In mode 4 palette 0 may be selected by writing the color select register using Video Int 16 Function 11. The definition of the palettes is contained in Section 1.11.2.2, Low Resolution (320x200) Graphics.

When mode 5 is selected it must be followed by a selection of palette zero (Color select register - See 1.11.5.2) in order to enable palette 2.

If the default display mode indicates an external monochrome adapter, then mode 7 is selected regardless of the mode in AL.

To select the Video mode the ROS does the following:

1. Disable video output.
2. Reset the Cursor Addresses for all pages to row 0 column 0.
3. Output the mode to the Mode select register (3B8/3D8).
4. Reload the 6845 CRTC registers from the Video Parameter table (which is supplied by interrupt 31).
5. Clear the video RAM. If an alpha mode is selected, the video RAM is filled with white space, i.e. ASCII space (020h) and the default attribute byte held in the NVR. The graphics mode fill is zeros.
6. If color display then set up the Color Select register (3D9h):

   Set the border color to the default background color.
   In graphics modes set intensified foreground colors.
   In mode 6 (Graphics 640 Mode) set white foreground color.
7. For text modes select page zero.
8. Set the cursor size to start cursor from the video parameter table.
9. Enable video output.

ROS Video Int 16 Function 1: Set Cursor Size.

This function is only relevant in alpha modes as the cursor is not supported in graphics modes. It sets the start and end scan numbers of the cursor.

CPU registers are used as follows:

Entry:

AH = 1
CH = Starting scan of cursor in range 0 to 31.
CL = Ending scan of cursor in range 0 to 31.

Exit:

All flags and registers preserved.

The ROS interprets a start value of 31 with the special meaning "turn the cursor off". In addition the 6845 hardware cursor disable function is enabled when cursor start bit register 5 is set.

ROS Video Int 16 Function 2: Set Cursor Address.

This function sets the current row and column addresses of the cursor in the specified page.

CPU registers are used as follows:

Entry:
AH = 2
BH = Page number for modes 0 to 3.
(Must be zero for all other modes.)
DH = Cursor Row Address.
DL = Cursor Column Address.
Exit:
All flags and registers preserved.

Refer to table 1.11.2.5 for valid page numbers and page starting address details.

ROS Video Int 16 Function 3: Get Cursor Address.

This function returns the current row and column address of the cursor in the specified page.

CPU registers are used as follows:

Entry:
AH = 3
BH = Page number for modes 0 to 3.
(Must be zero for all other modes.)
Exit:
DH = Cursor Row Address.
DL = Cursor Column Address.
CH = Starting scan of cursor.
CL = Ending scan of cursor.
All flags and other registers preserved.

Refer to table 1.11.2.5 for valid page numbers and page starting address details.

ROS Video Int 16 Function 4: Get Light Pen Address.

This function returns the address of the light pen.

CPU registers are used as follows:

Entry:
AH = 4.
Exit:
If Light Pen switch set then
AH = 1.
DH = Character Row address (0 to 24).
DL = Character Column address (0 to 79).
CH = Pixel Row address (0 to 199).
BX = Pixel Column address (0 to 639).

If Light Pen switch clear then
AH = 1.
BX, CH & DX preserved.
Always
All flags and other registers preserved.

ROS Video Int 16 Function 5: Set Display Page.

This function sets the active display page.

CPU registers are used as follows:

Entry:
AH = 5.
BH = Page number to be displayed.
Exit:
All flags and registers preserved.

Refer to table 1.11.2.5 for valid page numbers and page starting address details.

ROS Video Int 16 Function 6: Scroll Screen Up.

This function scrolls the active display page, or part of the active display page up a specified number of lines.

CPU registers are used as follows:

Entry:
AH = 6.
DH = Bottom Row of area to scroll.
DL = Right most Column of area to scroll.
CH = Top Row of area to scroll.
CL = Left most Column of area to scroll
BH = Attributes for blank lines scrolled onto the bottom of the scroll area.
AL = Number of lines to roll up.
If AL = 0 then blank the specified area.
If AL not zero then roll specified area up by the number of lines in AL.
If DH = CH then AL must be zero.
Exit:
All registers preserved.
Carry is clear and all other flags corrupt.

Scrolling always takes effect on the current active display page.

Hardware scrolling is not supported. Scrolling is achieved by copying areas of Video Display RAM.

In graphics modes blank lines are filled with zeros to display the current background color.

Note this function will fail to operate properly if on entry CH equals DH and AL is not zero. This is also true for all other compatible ROM environments.

ROS Video Int 16 Function 7: Scroll Screen down.

This function scrolls the active display page, or part of the active display page down a specified number of lines.

CPU registers are used as follows:

Entry:
AH = 7.
DH = Bottom Row of area to scroll.
DL = Right most Column of area to scroll.
CH = Top Row of area to scroll.
CL = Left most Column of area to scroll
BH = Attributes for blank lines scrolled onto the top of the scroll area.
AL = Number of lines to roll down.
If AL = 0 then blank the specified area.
If AL not zero then roll specified area down by the number of lines in AL.
If DH = CH then AL must be zero.
Exit:
All flags and registers preserved.

Scrolling always takes effect on the current active display page.

Hardware scrolling is not supported. Scrolling is achieved by copying areas of Video Display RAM.

Note this function will fail to operate properly if on entry CH equals DH and AL is not zero.

ROS Video Int 16 Function 8: Read Character and Attributes.

This function reads the character and its associated attribute byte at the current cursor address in a specified display page.

In graphics modes, the character pixel data is generated either from an internal character matrix table for characters 0 to 127 or from an external character matrix table for characters 128 to 255, the address of which is held in interrupt vector 31. See 2.3.23 for additional details.

CPU registers are used as follows:

Entry:
AH = 8.
BH = Page to read for alpha modes 0 to 3.
(Must be zero for all other modes.)
Exit:
AL = Character. (0 if no match found in Graphics Modes).
AH = Attributes byte. (Unchanged in graphics modes).
All flags and registers preserved.

Refer to 1.11.1 and 1.11.2 for the definition of the character attributes bytes.

ROS Video Int 16 Function 9: Write Character and Attributes.

This function writes a character (or a block of the same character) and its associated attribute byte to the current cursor position in a specified display page.

In graphics modes, the character pixel data is generated either from an internal character matrix table for characters 0 to 127 or from an external character matrix table for characters 128 to 255, the address of which is held in interrupt vector 31. See 2.3.23 for additional details.

CPU registers are used as follows:

Entry:
- AH = 9.
- AL = Character to write.
- BH = Page to write for alpha modes 0 to 3.
  - (Must be zero for all other modes.)
- BL = In alpha modes
  - Attributes of character.
- In graphic modes
  - Write mode as follows:
    - Bit 7 = 0 for character overwrite mode.
    - Bit 7 = 1 for character XOR mode.
    - Bits 0 and 1 = required character color (for modes 4 & 5 only).
- CX = Repeat Count.

Exit:
- All flags and registers preserved.

The repeat count specifies the number of consecutive locations to which the character and attributes are written. In graphics modes all characters must fit on the current line.

In graphics mode if bit 7 of BL is set then the data for the specified character is exclusive ORed with the data already in the display RAM at the cursor address.

ROS Video Int 16 Function 10: Write Character Only.

This function writes a character (or a block of the same character) to the current cursor position in a specified display page. In alpha modes the attribute bytes for all characters written remains unchanged.

In graphics modes, the character pixel data is generated either from an internal character matrix table for characters 0 to 127 or from an external character matrix table for characters 128 to 255, the address of which is held in interrupt vector 31. See 2.3.23 for additional details.

CPU registers are used as follows:

Entry:
- AH = 10.
- AL = Character to write.
- BH = Page to write for alpha modes 0 to 3.
  - (Must be zero for all other modes.)
- BL = In alpha modes
  - BL is not used.
- In graphic modes
  - Write mode as follows:

Bit 7 = 0 for character overwrite mode.
Bit 7 = 1 for character XOR mode.
Bits 0 and 1 = required character color (for modes 4 & 5 only).

CX = Repeat Count.

Exit:

All flags and registers preserved.

The repeat count specifies the number of consecutive locations to which the character is written.

In graphics mode if bit 7 of BL is set then the data for the specified character is exclusive ORed with the data already in the display RAM at the cursor address.

ROS Video Int 16 Function 11: Write Color Select Register.

This function writes the CGA compatible Color Select Register IRGB bits or the Palette select bits.

CPU registers are used as follows:

Entry:

AH = 11.

BH = Function select:

Zero - Set the IRGB bits (3-0) as specified by BL.

Non Zero - Set the Palette (0 or 1) as specified by BL.

Exit:

All flags and registers preserved.

Changing the palette number (BH non-zero) only has effect in modes 4 and 5 (320 pixel graphics mode). Refer to section 1.11.3 for further details.

ROS Video Int 16 Function 12: Write a Pixel.

This function writes an individual pixel (only valid in graphics modes).

CPU registers are used as follows:

Entry:

AH = 12.

DX = Pixel Row (0 to 199)

CX = Pixel Column (0 to 639)

AL = Write Mode:

Bit 7 = 0 for character overwrite mode.

Bit 7 = 1 for character XOR mode.

Bits 0 & 1 = Character Color for 320 graphics modes.

Exit:

All flags and registers preserved.

The pixel color specified in AL should be in the range 0 to 3 in modes 4 and 5 (graphics 320 pixel mode) and in the range 0 to 1 for mode 6 (graphics 640 pixel mode).

ROS Video Int 16 Function 13: Read a Pixel.

This function is used for reading an individual pixel (only in graphics modes).

CPU registers are used as follows:

Entry:
    AH = 13.
    DX = Pixel Row (0 to 199)
    CX = Pixel Column (0 to 639)
Exit:
    AL = Color of the specified pixel.
    All flags and other registers preserved.

ROS Video Int 16 Function 14: Write in TTY Emulation Mode.

This function writes the specified character in Teletype emulation mode at the current cursor address in the active display page.

CPU registers are used as follows:

Entry:
    AH = 14.
    AL = Character to write.
    BL = In alpha modes
        BL is not used.
    In graphic modes
        Write mode as follows:
            Bit 7 = 0 for character overwrite mode.
            Bit 7 = 1 for character XOR mode.
            Bits 0 and 1 = required character color (for modes 4 & 5 only).
Exit:
    All flags and registers preserved.

Upon completion of the write the cursor column is incremented by one. If the column address is greater than the line length then the column address is set to zero and the cursor row address is incremented by one.

If the incremented row address is greater than the last visible line then it is decremented to its original value and the entire page is scrolled up one line. In alpha modes the line added to the bottom of the page is cleared to spaces with the attributes the same as the first character in previous line. In graphic modes the bottom line is cleared to zeroes.

The following display characters are executed rather than displayed symbolically:

BEL (07h)
    Sounds a short (bleep) tone on the speaker.
BS (08h)
    Decrements the cursor column one character position unless the column is already zero in which case it is ignored.
CR (0Dh)
    Sets the cursor column address to zero.
LF (0Ah)

Increments the cursor row address by one and follows the scroll up procedure as detailed in the paragraph above.

All other control characters are displayed.

ROS Video Int 16 Function 15: Get Current Video Parameters.

This function returns the current Video Mode, the current display page and number of visible columns.

CPU registers are used as follows:

Entry:
AH = 15.
Exit:
BH = Current active display Page (or zero if in graphics modes or alpha mode 7).
AH = Number of visible columns (40 or 80).
AL = Current video mode (0 to 7).

All flags and other registers preserved.

2.3.8 IGA BIOS Interrupt 16: 'EGA Compatible' Video I/O (int 10h).

The IGA BIOS interrupt 16 service routine provides an extended set of service routines similar in many respects to those in the ROS ROM but with extra function selections to support the enhancements in the graphics hardware environment.

It is important to note that this interrupt entry point is active for 'EGA Compatible' display modes and that there are some differences in the standard call parameters. It is therefore important to use the correct section when reference is made to video interrupt 16. The CGA and MDA or Hercules Monochrome emulation mode compatible calls are covered in section 2.3.7. It should be noted that the Int 16 call registers are much the same for the video mode calls and video character display. When the 'EGA compatible' display modes are entered either as a result of the sw5 setting (OFF) or by the DISPLAY utility program, the video int 16 vector (at 0:40h) is set to point to the IGA BIOS's video interrupt's entry point and this set of service routines becomes active. Additional hardware setups must also be performed in the IGA to support the EGA compatibility.

CPU registers for 'EGA Compatible' Video Int 16 are used as follows:

Entry:
AH = Function Selector as follows:
0 - Set Video Mode.
1 - Set Cursor Size.
2 - Set Cursor Address.
3 - Get Cursor Address.
4 - Get Light Pen Address.
5 - Set Display Page.
6 - Scroll Screen Up.
7 - Scroll Screen Down.
8 - Read Character and Attributes.
9 - Write Character and Attributes.
10 - Write Character only.

11 - Write Color Palette.
12 - Write Pixel.
13 - Read Pixel.
14 - Write Character in Teletype Emulation mode.
15 - Get Video Parameters.
16 - Set Palette Register Value.
17 - Load Character Generator.
18 - Return EGC State/Set PrtSc Vector.
19 - Write String.

All other registers as required by the function.

Exit:

If selector is greater than 19 then the call is rejected and has no effect.

All flags preserved and registers as specified by the selected function.

In general parameters passed to these routines are not checked and care should be taken when choosing unusual parameters as unexpected results may occur.

IGA Video Int 16 Function 0: Set Video Mode.

CPU registers are used as follows:

Entry:
AH = 0
AL = Mode Selector as follows:

0 - Alpha 25 Rows by 40 Columns B/W.
1 - Alpha 25 Rows by 40 Columns Color.
2 - Alpha 25 Rows by 80 Columns B/W.
3 - Alpha 25 Rows by 80 Columns Color.
4 - Graphics 320 pixels by 200 lines using palette 1.
5 - Graphics 320 pixels by 200 lines using palette 2.
6 - Graphics 640 pixels by 200 lines, 2 colors.
7 - Alpha 25 Rows by 80 Columns in Monochrome.
13 - Graphics 320 pixels by 200 lines, 16 colors.
14 - Graphics 640 pixels by 200 lines, 16 colors.
15 - Graphics 640 pixels by 350 lines, Monochrome.
16 - Graphics 640 pixels by 350 lines, 16/64 colors.

Exit:
All flags and registers preserved.

The major difference between the 'B W' and 'Color' designations in the mode 0 - 3 selections is not the displayable colors but that there is a 'flicker wait' incorporated into the color modes to eliminate a 'snowing' effect in certain devices. In actual fact the IGA BIOS Int 16 can drive a CGA card device when it is the secondary display (refer to 1.22 - Switch Settings).

If the default display mode indicates monochrome a display , then mode 7 is selected regardless of the mode in AL unless AL = 15d in which case monochrome 640x350 graphics is selected.

Modes 0 - 7 are the 'Compatible' modes and modes 13 - 16 are for extended mode support. The gap between 8 and 12 are not user selectable display modes some of which are internal modes used for RAM font loading.

IGA Video Int 16 Function 1: Set Cursor Size.

This function is only relevant in alpha modes as the cursor is not supported in graphics modes. It sets the CRTC cursor start and cursor end values.

CPU registers are used as follows:

Entry:
AH = 1
CH = Starting scan of cursor.
CL = Ending scan of cursor.
Exit:
All flags and registers preserved.

Values greater between 32 and 64 turn the cursor off. Setting cursor start greater than cursor end usually turns the cursor off and other combinations give strange cursors.

IGA Video Int 16 Function 2: Set Cursor Address.

This function sets the current row and column addresses of the cursor in the specified page.

CPU registers are used as follows:

Entry:
AH = 2
BH = Page number.
DH = Cursor Row Address.
DL = Cursor Column Address.
Exit:
All flags and registers preserved.

Refer to table 1.11.2.5 for valid page numbers and page starting address details.

IGA Video Int 16 Function 3: Get Cursor Address.

This function returns the current row and column address of the cursor in the specified page.

CPU registers are used as follows:

Entry:
AH = 3
BH = Page number.
Exit:
DH = Cursor Row Address.
DL = Cursor Column Address.
CH = Starting scan of cursor.
CL = Ending scan of cursor.

All flags and other registers preserved.

Refer to table 1.11.2.5 for valid page numbers and page starting address details.

IGA Video Int 16 Function 4: Get Light Pen Address.

This function returns the address of the light pen.

CPU registers are used as follows:

Entry:
AH = 4.
Exit:
If Light Pen switch set then
AH = 1.
DH = Character Row address (0 to 24).
DL = Character Column address (0 to 79).
CH = Pixel Row address (0 to 199) for modes 0 - 7.
CL corrupt if modes 0-7.
CX = Pixel Row address (0 to 349) for modes 13 - 16.
BX = Pixel Column address (0 to 639).
If Light Pen switch clear then
AH = 1.
BX, CH & DX preserved.
Always
All flags and other registers preserved.

IGA Video Int 16 Function 5: Set Display Page.

This function sets the active display page.

CPU registers are used as follows:

Entry:
AH = 5.
BH = Page number to be selected.
Exit:
All flags and registers preserved.

Pages are numbered from 0 to n-1, where 'n' is the number of pages available. Refer to table 1.11.2.5 for valid page numbers and page addressing details.

IGA Video Int 16 Function 6: Scroll Screen Up.

This function scrolls the active display page, or part of the active display page up a specified number of lines.

CPU registers are used as follows:

Entry:
AH = 6.

DH = Bottom Row of area to scroll.
DL = Right most Column of area to scroll.
CH = Top Row of area to scroll.
CL = Left most Column of area to scroll
BH = Attributes for blank lines scrolled onto the bottom of the scroll area.
AL = Number of lines to roll up.
If AL = 0 then blank the specified area.
If AL not zero then roll specified area up by the number of lines in AL.
If DH = CH then AL must be zero.

Exit:
All registers and flags preserved.

Scrolling always takes effect on the current active display page.

Hardware scrolling is not supported. Scrolling is achieved by copying areas of Video RAM.

In graphics modes blank lines are filled with zeros to display the current background colour.

IGA Video Int 16 Function 7: Scroll Screen down.

This function scrolls the active display page, or part of the active display page down a specified number of lines.

CPU registers are used as follows:

Entry:
AH = 7.
DH = Bottom Row of area to scroll.
DL = Right most Column of area to scroll.
CH = Top Row of area to scroll.
CL = Left most Column of area to scroll
BH = Attributes for blank lines scrolled onto the top of the scroll area.
AL = Number of lines to roll down.
If AL = 0 then blank the specified area.
If AL not zero then roll specified area down by the number of lines in AL.
If DH = CH then AL must be zero.

Exit:
All flags and registers preserved.

Scrolling always takes effect on the current active display page.

Hardware scrolling is not supported. Scrolling is achieved by copying areas of Video Display RAM.

IGA Video Int 16 Function 8: Read Character and Attributes.

This function reads the character and its associated attribute byte at the current cursor address in a specified display page.

In graphics modes, the character pixel data is generated either from an internal character matrix table for characters 0 to 127 or from an external character matrix table for characters 128 to 255, the address of which is held in interrupt vector 31. See 2.3.23 for additional details.

CPU registers are used as follows:

Entry:
AH = 8.
BH = Page to read.
Exit:
AL = Character. (0 if no match found in Graphics Modes).
AH = Attributes byte. (Unchanged in graphics modes).
All flags and registers preserved.

Refer to 1.11.1 and 1.11.2 for the definition of the character attributes bytes.

Refer to table 1.11.2.5 for valid page numbers and page starting address details.

IGA Video Int 16 Function 9: Write Character Attributes.

This function writes a character (or a block of the same character) and its associated attribute byte to the current cursor position in a specified display page.

In 'CGA compatible' graphics modes (3 - 5), the character pixel data is generated either from an internal character matrix table for characters 0 to 127 or from an external character matrix table for characters 128 to 255, the address of which is held in interrupt vector 31. See 2.3.23 for additional details.

For the extended 'EGA compatible' modes (13 - 16) the full 256 character matrix is pointed to by interrupt vector 67 (at address 10Ch). This vector is initialised by the IGA ROM BIOS during Power-On Self Testing and points to a set of ROM character sets in the C2000-C7FFF address range.

CPU registers are used as follows:

Entry:
AH = 9.
AL = Character to write.
CX = Repeat Count.
BH = Page to write.
BL = In alpha modes
Attributes of character.
In graphic modes
Write mode as follows:
Bit 7 = 0 for character overwrite mode.
Bit 7 = 1 for character XOR mode.
Bits 0 to 3 = Character color.
Exit:
All flags and registers preserved.

The repeat count specifies the number of consecutive locations to which the character and attributes are written. In graphics modes all characters must fit on the current line.

In graphics mode if bit 7 of BL is set then the data for the specified character is exclusive ORed with the data already in the display RAM at the cursor address.

The character color specified in BL should be in the range 0 to 3 in 320 x 200 graphics modes, in the range 0 to 1 for mode 640 wide B/W graphics modes, and in the range 0 to 15 for 640 wide 16 color

graphics modes.

IGA Video Int 16 Function 10: Write Character Only.

This function writes a character (or a block of the same character) to the current cursor position in a specified display page. In alpha modes the attribute bytes for all characters written remains unchanged.

See graphics character vector details in function 9 (Write Char/Attribute) above.

CPU registers are used as follows:

Entry:
- AH = 10.
- AL = Character to write.
- CX = Repeat Count.
- BH = Page to write.
- BL = In alpha modes
  - BL is not used.
- In graphic modes
  - Write mode as follows:
    - Bit 7 = 0 for character overwrite mode.
    - Bit 7 = 1 for character XOR mode.
    - Bits 0 to 3 = Character color.

Exit:
- All flags and registers preserved.

The repeat count specifies the number of consecutive locations to which the character is written.

In graphics mode if bit 7 of BL is set then the data for the specified character is exclusive ORed with the data already in the display RAM at the cursor address.

The character color specified in BL should be in the range 0 to 3 in 320 x 200 graphics modes, in the range 0 to 1 for mode 640 wide B/W graphics modes, and in the range 0 to 15 for 640 wide 16 color graphics modes.

IGA Video Int 16 Function 11: Write Color Palette.

This function writes the CGA compatible Color Palette.

CPU registers are used as follows:

Entry:
- AH = 11).
- BH = Function select:
  - Zero - Set background color as specified by BL.
  - Non Zero - Set the Palette (0 or 1) as specified by BL.

Exit:
- All flags and registers preserved.

The palette number setting (BH non-zero) only applies to 320 x 200 graphics mode. Refer to section 1.11.2 for hardware details.

IGA Video Int 16 Function 12: Write a Pixel.

This function writes an individual pixel (only valid in graphics modes).

CPU registers are used as follows:

Entry:
AH = 12.
DX = Pixel Row.
CX = Pixel Column.
AL = Write Mode:
Bit 7 = 0 for character overwrite mode.
Bit 7 = 1 for character XOR mode.
Bits 0 to 3 = Pixel Data.
Exit:
All flags and registers preserved.

The pixel data specified in AL should be in the range 0 to 3 in 320 x 200 graphics modes, in the range 0 to 1 for mode 640 wide B/W graphics modes, and in the range 0 to 15 for 640 wide 16 color graphics modes.

IGA Video Int 16 Function 13: Read a Pixel.

This function is used for reading an individual pixel (only in graphics modes).

CPU registers are used as follows:

Entry:
AH = 13 (0Dh).
DX = Pixel Row.
CX = Pixel Column.
Exit:
AL = Specified pixel's color.
All flags and other registers preserved.

The specified pixel's color will be the same as the pixel data written by function 12 above.

IGA Video Int 16 Function 14: Write in TTY Emulation Mode.

This function writes the specified character in Teletype emulation mode at the current cursor address in the active display page.

CPU registers are used as follows:

Entry:
AH = 14.
AL = Character to write.
BL = In alpha modes
BL is not used.
In graphic modes
Write mode as follows:

Bit 7 = 0 for character overwrite mode.
Bit 7 = 1 for character XOR mode.
Bits 0 to 3 = Character color.

Exit:

All flags and registers preserved.

The character color bits are the same as for the write character (function 10) above.

Upon completion of the write the cursor column is incremented by one. If the column address is greater than the line length then the column address is set to zero and the cursor row address is incremented by one.

If the incremented row address is greater than the last visible line then it is decremented to its original value and the entire page is scrolled up one line. In alpha modes the line added to the bottom of the page is cleared to spaces with the attributes the same as the first character in previous line. In graphic modes the bottom line is cleared to zeroes.

The following display characters are executed rather than displayed symbolically:

BEL (07h)

Sounds a short (bleep) tone on the speaker.

BS (08h)

Decrements the cursor column one character position unless the column is already zero in which case it is ignored.

CR (0Dh)

Sets the cursor column address to zero.

LF (0Ah)

Increments the cursor row address by one and follows the scroll up procedure as detailed in the paragraph above.

All other control characters are displayed.

IGA Video Int 16 Function 15: Get Current Video Parameters.

This function returns the current Video Mode, the current display page and number of visible columns.

CPU registers are used as follows:

Entry:

AH = 15.

Exit:

AH = Number of visible columns (40 or 80).
AL = Current video mode).
BH = Active Display Page Number.

All flags and other registers preserved.

IGA Video Int 16 Function 16: Set Palette Register(s).

This function enables setting individual Palette registers, the Hi-Res Border (overscan) register, all Palette registers at once or setting the Intensity/Blink mode bit.

CPU registers are used as follows:

Entry:
AH = 16.
AL = Sub-Function Selector as follows:
0 - Set Individual Palette Register.
1 - Set Hi-Res Border Register.
2 - Set All Palette Registers and Hi-Res Border Register.
3 - Set Blink/Intensity Mode Bit.
Exit:
All flags and registers preserved.
For sub-function 0:
BL = Palette register number (0 - 15).
BH = Palette Value (0 - 63).
For sub-function 1:
BH = Hi-Res Border Value (0 - 63).
For sub-function 2:
ES:DX = Pointer to a 17 byte table as follows:
0 to 15 = Palette Registers 0 to 15 Value.
16 = Hi-Res Border Value.
For sub-function 3:
BL = 1 - Enable Blink (Disable Intensified Foreground).
0 - Disable Blink (Enable Intensified Foreground).

IGA Video Int 16 Function 17: Load Character Generator.

This function enables setting of extended character sets.

CPU registers are used as follows:

Entry:
AH = 16.
AL = Sub-Function Selector as follows:
0 - Load User Table.
1 - Load ROM 8x14 Font Block.
2 - Load ROM 8x8 Font Block.
3 - Set Character Set Select Register.
16 - Load User Table.
17 - Load ROM 8x14 Font Block.
18 - Load ROM 8x8 Font Block.
32 - Set Video 8x8 Matrix Table (Int Vect 31).
33 - Set User Extended Hi-Res Matrix Vector & Matrix Size.
34 - Set ROM 8x8 Extended Hi-Res Matrix Vector & Matrix Size.
35 - Set ROM 8x14 Extended Hi-Res Matrix Vector & Matrix Size.
48 - Return Character Set Vector.
Exit:
All flags and registers preserved (except function selector 48).

Sub functions 0 to 2 perfomr the same function as sub-functions 16 to 18 except that 0 to 2 initiate a full mode change setup (without clearing the video RAM) and 16 - 18 reprogram the CRTC cursor parameters based on character (points) size. These selections load a selected font table into the font

areas on planes 2 & 3. Refer to Character Set Select Register description in section 1 for the alternate character block layouts and how they are selected using attribute bit 3.

For sub-function 0:
AL = 0. (Load User Font Table).
BL = Font Block to load (0 - 3).
DX = Starting Offset into Font Block (0 - 255 characters).
BH = Points (Bytes per Character) (1 - 31).
CX = Number of Characters to load. (1 - 256).
ES:BP = User Font Table Address Pointer.
For sub-function 1:
AL = 1. (Load ROM 8x14 Font Table).
BL = Font Block to load (0 - 3).
For sub-function 2:
AL = 2. (Load ROM 8x8 Font Table).
BL = Font Block to load (0 - 3).
For sub-function 3:
AL = 3. (Set Character Set Select Register).
BL = 4-bit Character Sel A/B setting.

Refer to the Character Set Select Register description in section 1 for the details concerning character selects A & B.

For sub-function 16:
AL = 16. (Load User Font Table).

Refer to function 0 for register requirements for the user font table. As stated earlier, this sub-function differs from its counterpart only in that the current mode setup is maintained with exception that the CRTC cursor is set up based on the current character height (points) selection. This sub-function and the following two sub-functions should be executed immediately following a mode change and the resultant CRTC reprogramming should be fairly close to the original settings else unexpected results may occur.

For sub-function 17:
AL = 17. (Load ROM 8x14 Font Table).
BL = Font Block to load (0 - 3).
For sub-function 2:
AL = 18. (Load ROM 8x8 Font Table).
BL = Font Block to load (0 - 3).

Refer to section 1.11.3 for the Character Set Select Register layout.

For sub-function 32:
AL = 32. (Set Video 8x8 Matrix Table).
ES:BP = User Graphics Font Address Pointer for characters
128 to 255. See 2.3.22 - Video Matrix Table.
For sub-function 33:
AL = 33. (Set User Hi-Res Graphics Characters Vector).
ES:BP = User Graphics Font Address Pointer for characters
CX = Bytes per Character. (1 - 31)
BL = 0 - Rows is specified by DL.

1 - 14 Rows.
2 - 25 Rows.
3 - 43 Rows.

This font table is used to form the characters written to the screen if the current Video Mode is one of the Extended Hi-Res graphics Modes.

The current value of Rows (-1) is stored in system storage location 0:484 and the bytes per character value (Points) is system storage location 0:485. The int 67 vector pointer is system vector location 0:10C.

For sub-function 48:
AL = 48. (Return Character Set Vector).
BH = 0 - Return vector 31 (User 8x8 upper 128 vector).
1 - Return vector 67 (Extended Hi-Res 256 vector).
2 - Return 8x14 ROM font address.
3 - Return 8x8 ROM font address.
4 - Return upper 128 origin of 8x8 ROM font address.
5 - Return 9x14 ROM font address.
Upon return from sub-function 48:
CX = Bytes Per Character (Points from 0:485).
DL = ROWS - 1 (from 0:484)
ES:BP = Requested Vector.

The 9x14 table is in the form of a 'fixup' table whereby the first byte is the character number requiring the 9x14 fixup followed by 14 bytes of character redefinition. The table is zero terminated.

IGA Video Int 16 Function 18: Return EGC state/Set PrtSc Vector.

This function either returns the current EGC status or selects the IGA BIOS's alternate print screen routine address to be stored in the PrtSc vector (5) location.

CPU registers are used as follows:

Entry:
AH = 18.
BL = 16 - Return EGC Status.
32 - Set PrtSc Vector to the IGA ROM BIOS print screen entry.
Exit:
If return EGC status then registers returned as follows:
BH = 0 - Color Mode Set.
1 - Monochrome Mode Set.
BL = Number of 64Kb Video RAM Blcoks less one j '3' (PC1640 has 256Kb Video RAM installed)
CL = Switch (1 to 4) states (inverted - from 0:488)
CH = Features (0Fh - Features connector not installed on PC1640).

All flags and other registers preserved.

If Set PrtSc then all flags and registers are preserved.

IGA Video Int 16 Function 19: Write String.

This function writes a string to video RAM at a specified cursor position in a specified display page. The supplied string can either consist of characters by themselves (in which case the attribute byte is supplied in BL) or character and attribute byte pairs. Control characters are executed as described in IGA Video Int 16 Function 14 and where they occur they are not associated with an attribute byte within the supplied string. Line wraps occur at the display width specified in the system variable ROWS (at 0:488).

CPU registers are used as follows:

Entry:
AH = 19.
AL = String Function Selector as follows:
0
- String consists of characters only and attribute in BL.
The Cursor stays at the position prior to the string write.
1
- String consists of characters only and attribute in BL.
The Cursor is updated to be the character after the string.
2
- String consists of character and attribute pairs.
The Cursor stays at the position prior to the string write.
3
- String consists of character and attribute pairs.
The Cursor is updated to be the character after the string.

BH = Page to Write.
DH = Cursor Row Address.
DL = Cursor Column Address.
CX = Character Count (not including attribute bytes if supplied with the string).
ES:BP = Address Pointer to the String.
Exit:
All flags and registers preserved.

2.3.9 Interrupt 17: System Configuration.

This software interrupt returns the current system configuration status as defined in RAM locations 00410 and 00411 hex (see section 2.4).

CPU registers are used as follows:

Entry:
No conditions.
Exit:
AX = System Configuration status:

Bit(s)	Function
14 & 15	Number of printers (1-3).
13	Not used.
12	Set if an optional games adapter is fitted.
11	Always zero.

Bit(s)	Function
9 & 10	Number of serial interfaces (1 or 2).
8	Not used.
7	Always zero.
6	Set if second floppy disk drive is fitted.
4 & 5	Default display mode (DDM).
2 & 3	Always set.
1	Set if 8087 NDP is installed.
0	Always set.

All flags and other registers preserved.

Section 1.8.2 (Port A - Status-1 Input) contains the default mode states as defined in the DDM1 and DDM0 bits.

2.3.10 Interrupt 18: Memory Size.

This software interrupt returns the system RAM size as held in system locations 00413 and 00414 hex.

CPU registers are used as follows:

Entry:
No conditions.
Exit:
AX = Number of 1K memory blocks fitted.
All flags and other registers preserved.

2.3.11 Interrupt 19: Disk I/O

This software interrupt provides disk read, write, verify, and format functions for the drives fitted to the standard floppy disk controller.

In actual practice by the time the MS-DOS operating system has been loaded and an applications program is activated, the DOS startup process will have saved the ROS's interrupt vector (from location 0:4Ch) and installed its own entry vector. DOS does this so that it can correct for such conditions as DMA over a 64K segment boundary and it will break this sort of I/O requrest into a number of smaller I/O requests. In addition any installed hard disk will have 'chained' itself into the interrupt 19 vector so that requests with the MSB of the drive number set can be serviced by its ROM based hard disk I/O routines (usually in the C8000h address range). In general the call parameters for the hard disks are the same as those described here for read and write, however there are a number of additional services provided by the hard disk BIOS ROM's. The hard disk error return codes are also somewhat extended to the floppy disk group documented here. The typical 'compatible' hard disk function selector list and errors are documentedd following the ROS's functions.

CPU registers are used as follows:

Entry:
AH = Disk I/O Function selector:
0 - Initialise the disk sub-system.
1 - Return the status of the last operation.

2 - Read a number of consecutive sectors.
3 - Write a number of consecutive sectors.
4 - Verify a number of consecutive sectors.
5 - Format a track.

Exit:

AH = Status Byte:

0 - Operation completed successfully.
1 - Incorrect function (or drive) specifier.
2 - Missing address mark error.
3 - Disk write protected (Write or Format commands only).
4 - Record not found.
8 - DMA overrun error.
9 - Attempted DMA over a 64K segment boundary.
16 - CRC error.
32 - Floppy disk controller error.
64 - Seek error.
128 - Floppy disk controller timeout (Drive Not Ready).

All other registers as specified by the selected function.

For all disk functions the Carry Flag (CY) will be clear if no error else it is set if an error (and AH = error number). All other flags are corrupt.

Disk Function 0: Initalise Disk Sub-System.

This function performs a total initialisation of the disk interface as follows:

1. Reset the FDC (Floppy Disk Controller).
2. Re-configure the FDC parameters to those specified in the disk parameter table (see interrupt 30).

CPU registers are used as follows:

Entry:

AH = 0.

Exit:

AH/Flags = Status as specified above.
All registers preserved.

When an error is returned by any other Diskette I/O function, the Initialise Disk function should be called prior to the next disk I/O operation.

Disk Function 1: Return Last Status.

This function returns the status byte and Carry Bit of the last disk I/O operation.

CPU registers are used as follows:

Entry:

AH = 1.

Exit:

AH/Flags = Status of last disk I/O as specified above.
All other registers preserved.

Disk Function 2: Read Sector.

This function reads a number of consecutive sectors. All sectors to be read must be on the same track.

CPU registers are used as follows:

Entry:
AH = 2.
DH = Head Number (0 or 1).
DL = Drive Number (0 or 1).
CH = Track Number.
CL = Starting Sector Number.
BX = Offset Address of Read Data Buffer.
ES = Segment Address of Read Data Buffer.
AL = Number of Sectors to Read.
Exit:
AH/Flags = Status as specified above.
AL = Number of Sectors successfully read.
(Corrupt if Timeout error.)
All other registers preserved.

Disk Function 3: Write Sector.

This function writes a number of consecutive sectors. All sectors to be written must be on the same track.

CPU registers are used as follows:

Entry:
AH = 3.
DH = Head Number (0 or 1).
DL = Drive Number (0 or 1).
CH = Track Number.
CL = Starting Sector Number.
BX = Offset Address of Write Data Buffer.
ES = Segment Address of Write Data Buffer.
AL = Number of Sectors to Write.
Exit:
AH/Flags = Status as specified above.
AL = Number of Sectors successfully written.
(Corrupt if Timeout error.)
All other registers preserved.

Disk Function 4: Verify Sector.

This function verifies a number of consecutive sectors. All sectors to be verified must be on the same track.

CPU registers are used as follows:

Entry:
AH = 4.
DH = Head Number (0 or 1).
DL = Drive Number (0 or 1).
CH = Track Number.
CL = Starting Sector Number.
AL = Number of Sectors to Verify.
Exit:
AH/Flags = Status as specified above.
AL = Number of Sectors successfully verified.
(Corrupt if Timeout error.)
All other registers preserved.

Since the verification process is halted upon the first occurrence of an error, AL represents the number of sectors successfully verified prior to the occurrence of an error or total sectors verified if no error.

Disk Function 5: Format Track.

This function formats an entire track.

CPU registers are used as follows:

Entry:
AH = 5.
DH = Head Number (0 or 1).
DL = Drive Number (0 or 1).
CH = Track Number.
BX = Offset Address of Format Buffer.
ES = Segment Address of Format Buffer.
Exit:
AH/Flags = Status as specified above.
All other registers preserved.

The format buffer contains four bytes of information for each sector on the track:

1. Track Number.
2. Side Number.
3. Sector Number.
4. Sector Size Code:
   0 - 128 bytes / Sector.
   1 - 256 bytes / Sector.
   2 - 512 bytes / Sector.
   3 - 1024 bytes / Sector.

The gap length, filler byte and sectors per track required by the FDC Format command are obtained from the DPT (See Disk Parameter Table - Section 2.3.22).

2.3.11.1 Hard Disk Call parameters and registers.

As explained in the beginning of the ROS's floppy disk Int 19 calls, the 'compatible' hard disk expansion slot ROM supports a register interface similar to the floppy disk I/O but with extended functions required for the hard disk environment. This section gives the setup registers for this 'compatible' hard disk I/O call.

CPU registers are used as follows:

Entry:

AH = Hard Disk I/O Function selector:

- 0 - Reset Controller.
- 1 - Return the status of the last operation.
- 2 - Read a number of consecutive sectors.
- 3 - Write a number of consecutive sectors.
- 4 - Verify a number of consecutive sectors.
- 5 - Format a track.
- 6 - Format a track with bad sector markers.
- 7 - Format the entire drive.
- 8 - Return drive parameters.
- 9 - Set drive parameters.
- 10 - Read Long.
- 11 - Write Long.
- 12 - Seek to Track.
- 13 - Reset Controller (Alternate Entry).
- 14 - Read Sector Buffer.
- 15 - Write Sector Buffer.
- 16 - Test Drive ready.
- 17 - Recalibrate Seek.
- 18 - Ram Diagnostic.
- 19 - Drive Diagnostic.
- 20 - Controller Diagnostic.
- 22 - Park Heads over the landing zone.

Exit:

AH = Status Byte:

- 0 - Operation completed successfully.
- 1 - Incorrect function (or drive) specifier.
- 2 - Missing address mark error.
- 4 - Record not found.
- 7 - Drive initialization failure.
- 8 - DMA overrun error.
- 9 - Attempted DMA over a 64K segment boundary.
- 11 - Bad Track marker detected.
- 16 - ECC error.
- 17 - ECC data corrected.
- 32 - Hard disk controller error.
- 64 - Seek error.
- 128 - Disk controller timeout (Drive Not Ready).
- 187 - Undefined error occurred.
- 255 - Read Status Failure.

The register call parameters for the function selections are very similar to the equivalent floppy disk I/O calls and are typically as follows:

Function Selector
    AH: 0 - 20 & 22 (as per the table above)
Sector Count
    AL: 1 - n. (17 Sectors / Track - typical)
Interleave Factor
    AL: 1 - 16 Typically 3 to 7 (for the formatting calls)
Drive Number
    DL: 80h - FFh (Drive C: is 80h, D: is 81h ... etc).
Sector Number
    CL: 1 - 17 (Lower 6 bits of CL).
Head Number
    DH: 0 - n (Typically 3 for 20MB drives with 4 heads).
Cylinder Number
    CH: 0 - 1023 (8 LS Bits and 2 additional MS bits in the upper 2 MS bits of CL).
Buffer Address
    ES:BX --> Segment:Offset value.

Exit:
    AH: Return status as listed in the above table.
    CF: Carry Flag Clear - No errors. (AH = 0)
    Carry Flag Set - Error ccode in AH.

When Drive parameters are requested (Function 8) the return registers are as follows:

Hard disk drive count
    DL. (1 to n)
Maximum Head Number
    DH. (0 to n-1)
Sectors per Track
    CL. (bits 0 - 5)
Maximum Cylinder Number
    CH. (10 bits: 8 LSBs in CH and 2 MS bits in the 2 upper bits of CL).

Cylinders and heads are numbered starting from zero, therefore the max numbers are 'n-1'. Sectors are numbered starting at 1 and go to n which is typically 17 for the current 'MFM' technology. Newer 'RLL' technology drives are beginning to appear with 26 sectors per track. The maximum track number is actually one higher than the number reported but the highest track (termed the maintenance cylinder) is reserved for diagnostic software maintenance tests so that applications cannot use this track for storage.

Setting bad track markers is the method used to force DOS to set its 'Bad Sector Markers' during the logical formatting process.

Hard disk formatting is a confusing subject because there are two formatting passes necessary before the disk is ready for usage. The first formatting required is the low level (hard) formatting process which writes the actual sector ID fields on the media. When a hard disk is being prepared for usage by MS-DOS it must then be partitioned (using the FDISK utility) and finally 'Logically Formatted' the using the MS-DOS FORMAT utility. MS-DOS formatting is merely a quick once over verify process to find any areas which are not readable and for which it will mark the unreadable blocks in the DOS File Allocation Table (FAT). MS-DOS then installs its file directory and FATs on the disk and reports disk size and bad sector counts. The initial hard formatting process is a factory process which must take place in a controlled

environment (temperature and etc.) and the manufacturer's media defect list is entered and bad tracks marked. User attempts to format hard disks are fraught with difficulties usually because of the media defect problem whereby areas which may appear readable to MS-DOS's formatter but which fail to retain certain data patterns contained in actual data written later and these will cause 'Unrecoverable Read Error' reports to appear and frustrate all concerned. It is generally a good idea to leave the debugger 'g=C800:5' ROM formatter entry point to the experts.

2.3.12 Interrupt 20: Serial I/O.

This software interrupt provides functions for character I/O to one of the two serial channels and functions for configuring the serial parameters.

Two channels are supported, logical serial device 0 (COM1:) which is always configured and logical serial device 1 (COM2:) which is optional. Power-up initialisation determines whether serial device 1 is installed.

CPU registers are used as follows:

Entry:
AH = Serial I/O function selector:
0 - Initialise Serial Port.
1 - Write Character to Serial Port.
2 - Read Character from Serial Port.
3 - Return status of Serial Port.
DX = Logical Channel Number (0 or 1).
All other registers as required by the specified function.
Exit:
AX = Returned Status/Character as defined by the function.
All flags and other registers preserved.

If logical channel number is out of range (greater than 1) or is not fitted then the function is abandoned and the timeout error status (bit 7 of AH) is returned and all other bits in AX are undefined.

The Logical Serial Device Timeout Count RAM variables (locations 0047C & 0047D) specify the time out delay (in half seconds) used for channel timeout. See section 2.4.

Serial Function 0: Initalise Port.

This function performs a complete reinitialisation of a serial channel. Setting the Baud Rate, Data Bits, Stop Bits and Parity.

CPU registers are used as follows:

Entry:
AH = 0.
DX = Logical Channel Number (0 or 1).
AL = Hardware configuration:

Bit(s)	Function
5 - 7	Baud Rate Code (0 - 7).
4	Set for Even Parity / Clear for Odd Parity.

Bit(s)	Function
3	Set Parity Enable.
2	Set for 2 Stop Bits / Clear for 1 Stop Bit.
1	Always set.
0	Set for 8 Data Bits/Clear for 7 Data Bits.

Exit:

AH = 8250 Line Status register (See section 3.4).
AL = 8250 Modem Status register.
All flags and other registers preserved.

The Baud Rate code (bits 5 thru 7) is one of the following:

0 - 110 Baud.
1 - 150 Baud.
2 - 300 Baud.
3 - 600 Baud.
4 - 1200 Baud.
5 - 2400 Baud.
6 - 4800 Baud.
7 - 9600 Baud.

If the hardware flow control bit in the NVR default display mode byte is set then RTS is raised true and DTR is set false. Otherwise the current state of the control lines is preserved.

Serial Function 1: Send Character.

This function performs a character out sequence to the selected port. The character is output when CTS and the 8250 Tx Holding Register Empty status is also true. If the character cannot be sent within the time specified in the logical serial device timeout count RAM variable then the command is abandoned and AH is returned with bit 7 set.

CPU registers are used as follows:

Entry:

AH = 1.
AL = Character to be sent.
DX = Logical Channel Number (0 or 1).

Exit:

AH = 8250 Line Status register bits 0 to 6. Bit 7 is set if the channel timed out else bit 7 is clear and the character was sent.

All flags and other registers preserved.

When this function is called, RTS is raised true.

Upon exit both the RTS and DTR control lines are left in their current state.

The Logical Serial Device Timeout Count RAM variables (locations 0:47C and 0:47D) specify the time out delay (in half seconds) used for channel timeout.

Serial Sub-Function 2: Read Character.

This function attempts to read a character from the specified serial port. The character is not read until both Data Ready (DR) and Data Set Ready (DSR) status bits are both true. If a character is not received within the time specified by the logical device timeout count then the command is abandoned and timeout status is flagged.

CPU registers are used as follows:

Entry:
AH = 2.
DX = Logical Channel Number (0 or 1).
Exit:
If character received from 8250 then
AL = Character received.
AH = Character status:

Bit(s)	Meaning
7 - 5	Always '0'.
4	Break status.
3	Set if framing error.
2	Set if parity error.
1	Set if overrun error.
0	Always '0'.

If logical channel timed out then
AH = 080h (bit 7 = 1).
Always
All flags and other registers preserved.

If the character is received with no errors then AH = 0 on exit.

Upon entry, if no character is available at the serial port DTR is set in the Modem Control Register.

If logical channel number is out of range or is not fitted then the function is abandoned and the timeout error status (bit 7 of AH) is returned and all other bits in AX are undefined.

The Logical Serial Device Timeout Count RAM variables (locations 0:47C and 0:47D) specify the time out delay (in half seconds) used for channel timeout.

Serial Function 3: Get Channel Status.

This function returns the status of the specified logical channel.

CPU registers are used as follows:

Entry:
AH = 3.
DX = Logical Channel Number (0 or 1).
Exit:
AH = 8250 Line Status register (See section 3.4).

AL = 8250 Modem Status register.
All flags and other registers preserved.

All flags and other registers preserved.

If logical channel number is out of range or is not fitted then the function is abandoned and the timeout error status (bit 7 of AH) is returned and all other bits in AX are undefined.

2.3.13 Interrupt 21: Enhanced Function Interrupt.

This software interrupt provides access to the enhanced hardware features of the AMSTRAD PC1640.

CPU registers are used as follows:

Entry:
AH = Enhanced function Selector:
0 - Read and Reset Mouse
1 - Write NVR Location.
2 - Read NVR Location.
6 - Return ROS Version Number.
All other registers as required by specified sub-function.

Functions 3 - 5 are for PC1512 video hardware and should not be used on the PC1640.

Exit:
If sub-function number out of range then
Carry True.
All other flags corrupt.
All registers preserved.
If sub-function within range then
All flags and registers as specified by the sub-function.

Enhanced Function 0: Read/Reset Mouse X/Y Counts.

Read and reset the mouse X and Y count registers. Each register is read twice. If the data from two consecutive reads differs then the process is repeated until two consecutive reads produce the same data. Upon completion of the read procedure the registers are cleared to zero.

CPU registers are used as follows:

Entry:
AH = 0.
Exit:
CX = Signed X count.
DX = Signed Y count.
Carry False.
Other flags corrupt.
All other registers preserved.

Enhanced Function 1: Write NVR Location.

This sub-function writes a specified location in the Real Time Clock Non-Volatile RAM (NVR), re-computes and stores the new checksum value. The location written is then read back and compared with the new value and if different an error code is returned.

CPU registers are used as follows:

Entry:
    AH = 1.
    AL = NVR Address to be written (0 to 63).
    BL = NVR Data to be written.
Exit:
    AH = Return Code:
        0 - NVR written successfully.
        1 - NVR Address out of range.
        2 - NVR Data write error.
    Carry false.
    All other flags corrupt.
    All other registers preserved.

Although locations 0-13 may be accessed using this function, they are used by the RTC hardware and should not be modified with this function.

Section 2.5 (Non Volatile RAM) contains the NVR information layout.

Enhanced Function 2: Read NVR Location.

This sub-function reads a specified location in the Real Time Clock Non-Volatile RAM (NVR). The checksum is computed and compared with the actual value and if the NVR checksum is incorrect an error code is returned.

CPU registers are used as follows:

Entry:
    AH = 2.
    AL = NVR Address to be read (0 to 63).
Exit:
    AH = Return Code:
        0 - NVR read successfully.
        1 - NVR Address out of range.
        2 - NVR checksum error.
    AL = Byte read from NVR.
    Carry false.
    All other flags corrupt.
    All other registers preserved.

Section 2.5 (Non Volatile RAM) contains the NVR information layout.

Enhanced Function 6: Return ROS Version Number.

This sub-function returns the two part ROS version number.

CPU registers are used as follows:

Entry:
    AH = 6.
Exit:
    BH = Release number.
    BL = Issue Number.

    Carry false.
    All other flags corrupt.
    All other registers preserved.

The Release Number is incremented only when the interface to the ROS is changed. The Issue Number is incremented for each version of a particular release. A new release always starts with issue number zero.

Note that this function call can be used to detect whether a program is running on an Amstrad PC. Prior to entry clear the carry flag and set BX to zero. Upon return if carry is set or BX is zero then the program is not running in an Amstrad PC. An additional hardware check (in the printer control port - Section Section 1.10.2) can be used to sort out whether the Amstrad PC is a PC1512 or a PC1640.

2.3.14 Interrupt 22: Keyboard I/O.

This software interrupt provides access to the keyboard buffer and the current toggle status.

CPU registers are used as follows:

Entry:
    AH = Keyboard I/O function selector.
        0 - Get key token from the keyboard buffer.
        1 - Return keyboard buffer status.
        2 - Return current key toggle and key States.
Exit:
    If function selector out of range then
        AH = AH - 2.
    If function within range then
        All flags and registers as specified by function.

Keyboard I/O Function 0: Get Key Token.

Return the next token from the key token buffer. If no key token is available then wait until a key token is available.

CPU registers are used as follows:

Entry:
    AH = 0.
Exit:
    AX = Key Token.
    All flags and other registers preserved.

Keyboard I/O Function 1: Return Keyboard Buffer Status.

Test whether the key token buffer is empty. If it is not empty return the next key token to be taken out of the buffer without removing it from the buffer.

CPU registers are used as follows:

Entry:
    AH = 1.
Exit:
    If key token buffer is empty then
        Zero flag true.
        AX corrupt.
    If one or more tokens in buffer then
        Zero flag false.
        AX = next key token to be removed from buffer.
    Always
        Interrupts enabled.
        All other flags corrupt.
        All other registers preserved.

Keyboard I/O Function 2: Return Shift States.

Return the current value of the shift states (from 00417h).

CPU registers are used as follows:

Entry:
    AH = 2.
Exit:
    AL = Current shift states:

Bit(s)	Function (Set if key active)
7	INS
6	CAPS LOCK
5	NUM LOCK
4	SCROLL LOCK
3	ALT
2	CTRL
1	LEFT SHIFT
0	RIGHT SHIFT

All flags and other registers preserved.

2.3.15 Interrupt 23: Printer I/O.

This software interrupt provides access to the three printer channels.

CPU registers are used as follows:

Entry:

AH = Printer I/O Function selector:

0 - Send character to printer port.

1 - Initialise printer port.

2 - Return printer port status.

DX = Logical Channel Number (0 - 2).

Other registers as specified by function.

Exit:

AH = Printer Port Status (Bits 1 - 7):

Bit(s)	Function (Bit Set True)
7	Printer Idle.
6	Printer Acknowledge
5	Paper Out.
4	Printer Selected.
3	I/O Error.
1 & 2	Always Zero.
0	Zero if I/O successful or set if Timeout. (see Printer functions).

All flags and other registers preserved.

Three logical channels are supported. Logical printer device 0 is the system port and is standard to all machines. The power-up initialisation sequence determines if additional external printer ports are present. When both additional printer interfaces are present, device 1 is the external printer port and device 2 is the printer port on the external monochrome VDU controller. If only one additional printer interface is present it is always logical device 1.

Locations 0478h - 047Ah contain the Logical Printer Device timeout counts (see section 2.4).

Printer Function 0: Print Character.

This function attempts to output a character to the specified printer port. If the character cannot be sent within the time specified by the logical printer timout count RAM variable then the command is abandoned and AH is returned with bit 0 set.

CPU registers are used as follows:

Entry:

AH = 0.

AL = Character to be printed.

DX = Logical Channel Number (0 - 2).

Exit:

AH = Printer Port status (as given above) or Timeout (Bit 0) set.

All flags and other registers preserved.

Printer Function 1: Initialise Printer Channel.

This function performs a complete reinitialisation of a specified printer channel (if present). The printer INIT signal is held low for approximately 4 milliseconds. Printer interrupts and auto linefeed are disabled.

CPU registers are used as follows:

Entry:
AH = 1.
DX = Logical Channel Number (0 - 2).
Exit:
AH = Printer Port status (as given above) or Invalid Channel (Bit 0) set.
All flags and other registers preserved.

Printer Function 2: Return Channel Status.

This function returns the status register of the specified logical printer channel (if present).

CPU registers are used as follows:

Entry:
AH = 2.
DX = Logical Channel Number (0 - 2).
Exit:
AH = Printer Port status (as given above) or Invalid Channel (Bit 0) set.
All flags and other registers preserved.

2.3.16 Interrupt 24: System Restart.

If a Hard Disk ROM BIOS was initialized during Power-Up Initialization then it will have installed its entry point into this vector and saved the ROS's vector in its private storage area. Its bootstrap process resembles the process described below (under Int 25) and if successful the system from the active partition of the hard disk will be loaded if not then it executes the ROS's Int 24 whose vector it has saved.

This software interrupt is intended to provide an orderly system restart capability. A message is displayed on the active VDU requesting that the user 1Insert a SYSTEM disk into Drive A1 and 1Then press any key.1 When the keypress is received, the Disk Bootstrap process (Interrupt 25) is invoked.

CPU registers are used as follows:

Entry:
No conditions.
Exit:
Disk Bootstrap.

2.3.17 Interrupt 25: Disk Bootstrap.

This software interrupt to provide access to the disk bootstrap process which is normally executed after power-up initialisation tests.

The ROS attempts to load the bootstrap sector (from drive A, side 0, track 0, sector 1) into memory at 07C00h. If the bootstrap sector is loaded successfully it is given control (far jump to segment 0000 offset 7C00). If the bootstrap sector cannot be loaded after 10 retries, the ROS will display a message prompting the user to "Insert a SYSTEM disk into drive A" and "Then press any key." The ROS then waits for the keypress and repeats the system restart procedure (Int 24).

Useage of this interrupt under MS-DOS 3.2 may fail if there has been any process installed in memory which has connected itself to any active interrupt chains such as the System Clock (ticker) interrupt. The ROS performs the bootstrap without initialising the interrupt vectors and if a process has become resident under MS-DOS it will be removed during the bootstrap process, however the next occurrence of an interrupt will give control to the address of the formerly resident process and as a result the system hangs because the processor is executing random code. Programs such as MOUSE.COM are of this category of resident timer-chained process.

CPU registers are used as follows:

Entry:
No conditions.
Exit:
To program loaded by Disk Bootstrap.

2.3.18 Interrupt 26: System Clock & Real Time Clock.

This software interrupt routine provides access to both the system (software maintained) clock location as well the Real Time Clock (RTC) hardware.

CPU registers are used as follows:

Entry:
AH = Clock Function specifier:
0 - Get System Clock.
1 - Set System Clock.
2 - Get RTC time.
3 - Set RTC time.
4 - Get RTC date.
5 - Set RTC date.
6 - Set RTC alarm.
7 - Reset RTC alarm.
All other registers as required by function.
Exit:
All registers as specified by function.

Clock Function 0: Get System Clock.

This function returns the current value of the 32 bit system clock value.

CPU registers are used as follows:

Entry:
AH = 0.
Exit:
DX = Least Significant Word of the clock count.
CX = Most Significant Word of the clock count.
AL = 24 Hour Flag:
0 if not past 24 hours.
1 if past 24 hours.

All flags and other registers preserved.

The 32 but system clock is incremented every 54.9 milliseconds by the ticker hardware interrupt routine. When the count reaches the 24 hour value (0001800B0h) the 24 Hour flag is set and the system clock count is reset to zero. This 24 hour count is based on the system clock 1.19318 MHz divided by the maximum divisor, 65536. This gives an interrupt rate of 54.92549323 Ms which when divided into the number of seconds in 24 hours gives this 24 Hour time value above.

Note that the 24 hour flag is reset to zero after it has been read.

Clock Function 1: Set System Clock.

This function sets the current value of the 32 bit system clock value.

CPU registers are used as follows:

Entry:
AH = 1.
DX = Least Significant Word of the clock count.
CX = Most Significant Word of the clock count.
Exit:
All flags and other registers preserved.

Clock Function 2: Get RTC Time.

This function gets the current time from the Real Time Clock.

CPU registers are used as follows:

Entry:
AH = 2.
Exit:
If RTC not operating then
Carry True.
CX DX preserved.
If RTC operating then
Carry False.
CH = Hour (BCD).
CL = Minute (BCD).
DH = Second (BCD).
Always
All other flags corrupt.
All other registers preserved.

Clock Sub-Function 3: Set RTC Time.

This function sets the Real Time Clock time.

CPU registers are used as follows:

Entry:

AH = 3.
CH = Hour (BCD).
CL = Minute (BCD).
DH = Second (BCD).
DL = 1 to enable daylight savings option (otherwise 0).

Exit:

If RTC not operating then
Carry True.
If RTC operating then
Carry False.
Always
All other flags corrupt.
All other registers preserved.

When the daylight savings option is set it enables two special updates of the current time. On the last Sunday in April, the time increments from 1:59:59 AM to 3:00:00 AM. Also on the last Sunday in October the time increments from 1:59:59 AM to 1:00:00 AM.

Note that this option also disables the alarm function.

Clock Function 4: Get RTC Date.

This function gets the current date from the Real Time Clock.

CPU registers are used as follows:

Entry:

AH = 4.

Exit:

If RTC not operating then
Carry True.
CX DX preserved.
If RTC operating then
Carry False.
CH = Century (BCD).
CL = Year (BCD).
DH = Month (BCD).
DL = Day Of Month (BCD).
Always
All other flags corrupt.
All other registers preserved.

The century byte is set to 19 (BCD) if the year is 80 (BCD) or above otherwise it is set to 20 (BCD).

Clock Function 5: Set RTC Date.

This function sets the Real Time Clock time.

CPU registers are used as follows:

Entry:

AH = 5.
CH = Century (BCD) [Ignored].
CL = Year (BCD).
DH = Month (BCD).
DL = Day of Month (BCD).

Exit:

If RTC not operating then
    Carry True.
If RTC operating then
    Carry False.
Always
    All other flags corrupt.
    All other registers preserved.

Century is ignored and is computed as described in Clock Function 4.

Clock Function 6: Set RTC Alarm.

This function sets the alarm time and arms the Real Time Clock alarm interrupt. The alarm interrupt will occur then the current time matches the alarm time. An application program which uses this function must first write the address of its alarm interrupt routine into interrupt vector 10.

CPU registers are used as follows:

Entry:

AH = 6.
CH = Hour (BCD).
CL = Minute (BCD).
DH = Second (BCD).

Exit:

If RTC alarm already set then
    Carry True.
If RTC alarm not already set then
    Carry False.
Always
    All other flags corrupt.
    All other registers preserved.

Clock Function 7: Kill RTC Alarm.

This function disarms the Real Time Clock alarm function.

CPU registers are used as follows:

Entry:

AH = 7.

Exit:

All flags and registers preserved.

2.3.19 Interrupt 27: Keyboard Break Interrupt.

This software interrupt is invoked by the keyboard hardware interrupt routine when a keyboard break ([CTRL] + [NUM LOCK]) is detected.

The power-up initialisation process loads the address of a dummy break handler routine which does an interrupt return (IRET) instruction.

Application programs which supply a keyboard break interrupt must conform to the following register conventions:

Entry:
DS = 0040h (Spanning the ROS data).
Exit:
All registers must be preserved except AX, BX, CX, DX, DS and Flags which may be corrupt.

The supplied interrupt routine must not invoke any other ROS interrupts from within itself but may modify any of the system RAM locations used by the ROS.

2.3.20 Interrupt 28: External Ticker Interrupt.

This software interrupt is called from within the System Clock hardware interrupt routine. It is initialised by power-up with a dummy handler which returns from interrupt by doing an IRET instruction. It can be used by application programs which require a process to be run at a regular interval.

Application programs which supply an external ticker interrupt must conform to the following register conventions:

Entry:
DS = 0040h (Spanning the ROS data).
Exit:
All registers must be preserved except AX, DX, DS and Flags which may be corrupt.

The supplied interrupt routine must not invoke any other ROS interrupts from within itself but may modify any of the system RAM locations used by the ROS.

2.3.21 Interrupt 29: VDU Parameter Table.

This interrupt vector location contains the 20-bit address of the Video Parameter table used in setting up the 6845 CRTC when changing video mode. Upon power-up or after a reset, the system ROS initialisation process loads the ROM table address into this vector location (0074-0077 hex).

The Video Parameter table consists of four consecutive 16 byte entries. Each entry contains an initialisation quantity for each of the MC6845 CRTC registers (See section 1.11.5). When a new video mode is selected the table entry used for initialisation as follows:

Table Entry	VDU Mode
0	0 - Alpha 25 by 40 Chars. 1 - Alpha 25 by 40 Chars.
1	2 - Alpha 25 by 80 Chars. 3 - Alpha 25 by 80 Chars.
2	4 - Graphics 320 by 200 Pixels, palettes 0 or 1. 5 - Graphics 320 by 200 Pixels, palette 2.

Table Entry	VDU Mode 6 - Graphics 640 by 200 Pixels.
3	7 - Alpha 25 by 80 chars using monochrome adapter.

The table contains the following initialisation data:

Register Number	Function	Entry 0	Entry 1	Entry 2	Entry 3
R0	Horizontal Total	56	113	56	97
R1	Horizontal Displayed	40	80	40	80
R2	Horiz. Sync Position	45	90	45	82
R3	Horiz. Sync Width	10	10	10	15
R4	Vertical Total	31	31	127	25
R5	Vertical Total Adjust	06	06	06	06
R6	Vertical Displayed	25	25	100	25
R7	Vertical Sync Position	28	28	112	25
R8	Interlace	02	02	02	02
R9	Max. Raster Address	07	07	01	13
R10	Cursor Start Raster	06	06	06	11
R11	Cursor End Raster	07	07	07	12
R12	Start Address High	00	00	00	00
R13	Start Address Low	00	00	00	00
R14	Cursor Location High	00	00	00	00
R15	Cursor Location Low	00	00	00	00

2.3.22 Interrupt 30: Disk Parameter Table.

This interrupt vector location contains the 32-bit address of the parameter table of configuration parameters for the disk interface. Upon power-up or after a reset, the initialisation process loads the ROM table address into this vector location (0078 - 007B hex).

The Disk Parameter Table consists of 11 bytes as follows:

Byte	Function	Value
0	2nd byte of the disk controller specify command. (6 Ms Step Rate, Head Unload delay = 15)	DFh
1	3rd byte of the disk controller specify command. (Head Load delay = 1 & FDC DMA Mode = 0)	02h
2	Motor off timeout (approx 5.4 seconds).	64h
3	Sector size selector (512 bytes)	02h
4	End of Track (sector 9)	09h
5	Gap length for Read/Write commands.	2Ah
6	DTL - Data Length	FFh
7	Gap Length for format command.	50h
8	Filler byte for format command.	F6h
9	Head Settling Delay (15 Ms)	0Fh
10	Motor on Delay (500 Ms)	04h

2.3.23 Interrupt 31: VDU Matrix Table.

This interrupt vector location contains the 32-bit address of the VDU matrix table used in compatible graphics modes for generating pixel data for characters 128 to 255.

Upon power-up or after a reset, the initialisation process loads this vector (007C-007F) with all zeros to indicate that no external VDU matrix table is loaded. Programs such as GRAFTABL.EXE load a resident upper 128 character matrix which can be used in the 640x200 resolution graphics modes. In addition the IGA ROM BIOS uses another vector, 67 (vector location 0:10Ch) which it initialises to point to its full 256 character ROM table. IGA Video Int 16, function 17, sub-function 48, BH=4, can be used to obtain the upper 128 origin of its 8x8 ROM font address and this resultant value can be used to initialise interrupt vector 31.

When the IGA extended font ROM is enabled for foreign character sets (sw9 ON) then a vector table is available at C7FE8 (C000:7FE8) which contains 12 word size entries. The first four words are for Danish fonts, the next four words are for Portugese and the final four words are for the Greek Fonts. The first pointer in each of these groups is for the 8x8 high 128 character set for the national variant. By using the offset value plus a segment value of C000h, national variant upper 128 character sets can easily be obtained without using any disk based tables.

Each of the 128 character table entries consists of eight bytes, one for each character scan. The first byte is the top scan value and the last byte is the button scan value. The MSB, bit 7, is the left most pixel and the LSB, bit 0, is the right most pixel of the scan. A set bit displays the foreground colour and a reset bit displays the background colour.

2.4 System RAM Variables.

The System RAM address space from 00300 to 00500 is used by the ROS for variable storage. The following table lists the variables and their usage. They are either classified as Byte (8-bit), Word (16-bit), Long Word (32-bit) or Buffer (greater than 32-bit) storage locations. Depending on the CPU's segment register setting, the variables at 004xx can be said to be referenced at 40:XXX or 0:4XX.

Location(s)	Usage
00300-003FF	Initialisation Stack (Buffer). Used as stack area only during initialisation.
00400	Logical Serial Device 0 Base I/O Address (Word). Contains the base address of logical serial device 0. Initally the System Asynchronous Serial port address.
00402	Logical Serial Device 1 Base I/O Address (Word). Contains the base address of logical serial device 1. Initally the external asynchronous serial port or zero if it is not present at initialisation.
0404 - 0407	Reserved.
00408	Logical Printer Device 0 Base I/O Address (Word). The base address of logical printer device 0. Initally the System Parallel Printer port.
0040A	Logical Printer Device 1 Base I/O Address (Word). The base address of logical printer device 0. Initially the external parallel printer port if it is present else it points to the external monochrome VDU controller if it is present. If neither is present it is initialised to zero.
0040C	Logical Printer Device 2 Base I/O Address (Word). Initially points to the external monochrome VDU controller if both the external parallel

Location(s)	Usage
	printer port and the external monochrome VDU controller are present. If either is not installed initialised to zero.
0040E	Reserved (Word).
00410	System Configuration Status (Word). Contains the System Configuration as follows:

Bit(s)	Function
14 & 15	Number of printers (1-3).
13	Not used.
12	Set if an optional games adapter is fitted.
11	Always zero.
9 & 10	Number of serial interfaces (1 or 2).
8	Not used.
7	Always zero.
6	Set if second floppy disk drive is fitted.
4 & 5	Default VDU mode.
2 & 3	Always set.
1	Set if 8087 NDP is installed.
0	Always set.

00412	Reserved (Byte).
00413	Total RAM Size (Word). Initially set to the number of 1K User (System) RAM Blocks installed. (640 for PC1640).
00415	Extra RAM Size (Word). Initially set to the number of 1K User (System) RAM Blocks installed minus 64. (576 for PC1640).
00417	Key Toggles and Key States (Byte). This byte is used to record the state of the Key Toggles (bits 4-7) and Key States (bits 0-3) as follows:

Bit	Key (Bit set if active)
7	INS
6	CAPS LOCK
5	NUM LOCK
4	SCROLL LOCK
3	ALT
2	CTRL
1	LEFT SHIFT
0	RIGHT SHIFT

00418	Keys down (Byte). This byte is used to record the state of the toggle keys so that they do not repeat when the key is held down.

Bit	Key (Set if down)
7	INS
6	CAPS LOCK
5	NUM LOCK
4	SCROLL LOCK

<table>
<tr><td>Location(s)</td><td>Usage</td></tr>
<tr><td></td><td>Bit 3 is set when Pause state ([CTRL][NUM LOCK] is active. Bits 0 - 2 are unused and initially zeroed.</td></tr>
<tr><td>00419</td><td>Absolute Key Token Number (Byte).
When an absolute key token numbered is entered via ALT and the numeric key pad, this variable holds the current state of the token.</td></tr>
<tr><td>0041A</td><td>Key Token Buffer Out Pointer (Word).
This variable holds the absolute offset to the next key token to be removed from the key token buffer.
Note that the ROS assumes that the buffer has a segment paragraph address of 0040h.</td></tr>
<tr><td>0041C</td><td>Key Token Buffer In Pointer (Word).
This variable holds the absolute offset to the next empty position in the key token buffer. The buffer is empty when this location is the same as the Out Pointer.</td></tr>
<tr><td>0041E</td><td>Key Token Buffer (Buffer).
The Key Token Buffer is a 16 word circular buffer used to store up to 16 key tokens.</td></tr>
<tr><td>0043E</td><td>Drive Restore Flag (Byte).
Each floppy disk drive has a restore flag associated with it (bit 0 for drive 0 and bit 1 for drive 1).
If the restore flag for the specified drive is reset prior to any disk access (read/write/verify /format), then the restore command is issued to the FDC for that drive. If successful then the associated flag bit is set. When the initialise sub-function of the disk interrupt is called the restore flag is cleared.
Bit 7 is used for handling FDC hardware interrupts.</td></tr>
<tr><td>0043F</td><td>Drive Motor Flag (Byte).
When a disk drive motor is running then either bit 0 or bit 1 will be set to which drive (0 or 1 respectively) is selected.</td></tr>
<tr><td>00440</td><td>Drive Motor Timeout Counter (Byte).
After each disk operation the the motor off timeout count is copied from the Disk Parameter table (See interrupt 30) into this variable. Each time the system clock interrupt is executed, the count is decremented. When it reaches zero the Drive Motor Flag is reset.</td></tr>
<tr><td>00441</td><td>Disk Status (Byte).
This byte holds the status returned by the last disk operation. (See section 2.3.11 Disk I/O Interrupt - Function 1.)</td></tr>
<tr><td>00442</td><td>FDC Results/HD Parameter Buffer (Buffer).
This seven byte buffer is used for storage of the FDC status information returned upon the completion of a disk I/O operation. It is also used by the Hard Disk BIOS ROM for call parameter storage.</td></tr>
<tr><td>00449</td><td>Current Video Mode (Byte).
The current Video mode from the last Int 16 setmode call is stored here.</td></tr>
<tr><td>0044A</td><td>Visible Display Columns (Word).
The number of visible character columns currently being displayed is stored here.</td></tr>
<tr><td>0044C</td><td>Video Display Page Size (Word).
This word holds the amount of Video RAM used by the ROM BIOS to display one page as defined below:
<table>
<tr><td>Mode(s)</td><td>Size</td></tr>
<tr><td>0 & 1</td><td>2048</td></tr>
<tr><td>2 & 3</td><td>4096</td></tr>
</table>
</td></tr>
</table>

Location(s)	Usage
	4 - 6 16384 7 4096 13 8192 14 16384 15 - 16 32768
0044E	Display Page Start Offset (Word). Contains the origin of the currently active VDU display page.
00450	Cursor Address Buffer (Buffer) This 16 byte buffer contains the row and column addresses for up to eight display pages. This is the limiting factor is the number of pages which can be supported by the video ROM BIOS routines.
00460	Cursor End Scan (Byte). This byte contains the current end scan number that was programmed into the CRT controller.
00461	Cursor Start Scan (Byte). This byte contains the current start scan number that was programmed into the CRT controller.
00462	Active Display Page (Byte). This byte contains the selected display page number.
00463	CRTC I/O Address (Word). This word contains the I/O address of the CRTC interface currently in use. (3B4 - Mono / 3D4 - Colour)
00465	Current Video Mode Control Register (Byte). This byte contains the current contents of the Video Mode Control Register.
00466	Current CGA Colour Select Register (Byte). This byte contains a copy of the data loaded into CGA colour select register.
0467-046B	Reserved
0046C	System Clock (Long Word). The 32 bit system clock count
00470	24 Hour Flag (Byte). When the system clock reaches 0001800B0h then it is cleared and this flag byte is set to 0FFh. Note that reading the clock via interrupt 26 clears this flag.
00471	Break (Byte). This byte is initially set to zero. Each time Break ([CTRL]+[NUM LOCK]) is detected, bit 7 is set. An application program using this bit to detect break must reset bit 7 when it detects the break event.
00472	System Reset Flag (Word). When soft reset, [CTRL]+[ALT]+[DEL], is detected this location is set to 01234h prior to issuing a system reset. The power-up self test routine then recognizes this pattern and does not repeat the RAM tests. Setting 1235h prevents full hardware reset.
0474	Last Hard Disk I/O Completion Code. This location is used by the Hard Disk BIOS ROM to store the last I/O operation's completion code.
0475	Hard Disk Drive Count. This location is used by the Hard Disk BIOS ROM to store the number of drives on line.

<table>
<tr><td>Location(s)</td><td>Usage</td></tr>
<tr><td>0476-0477</td><td>Reserved for Hard Disk BIOS ROM.</td></tr>
<tr><td>0478 - 47A</td><td>Logical Printer Device 0 - 2 Timeout Count (Buffer).
These timeout counts specify how long the ROS should wait in half second multiples, while trying to output a character to a logical printer channel. The are initially set to 20 (10 Second timeout).</td></tr>
<tr><td>0047B</td><td>Reserved.</td></tr>
<tr><td>047C - 047D</td><td>Logical Serial Device 0 - 1 Timeout Count (Buffer).
These timeout counts specify the length of the wait time half second intervals for character I/O to a particular logical serial channel. All counts are set to 1 (for a) second timeout).</td></tr>
<tr><td>0047E</td><td>Reserved.</td></tr>
<tr><td>00480</td><td>Key Token Buffer Start Address (Word).
Offset pointer to the start of the key token buffer.
Note that the assumed buffer segment paragraph address is 0040h.</td></tr>
<tr><td>00482</td><td>Key Token Buffer End Address (Word).
Offset pointer to the start of the key token buffer.</td></tr>
<tr><td>00484</td><td>Display Rows (Byte).
This location contains the number of character rows (less one) on the display screen.</td></tr>
<tr><td>00485</td><td>Character Points (Byte).
This location contains the current character matrix length in bytes.</td></tr>
<tr><td>00487</td><td>EGC Status (Byte).
This location is used by the IGA ROM BIOS to hold its current status information (Colour/Mono, Primary/Sec).</td></tr>
<tr><td>00488</td><td>Switches (Byte).
This location contains the current switch settings for the EGC control switches 1-4 in bits 0-3 (inverted). The upper four bits are ones.</td></tr>
<tr><td>00500</td><td>Print Screen Status (Byte).
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>0</td><td>Print Screen completed OK.</td></tr>
<tr><td>1</td><td>Print Screen in progress.</td></tr>
<tr><td>255</td><td>Print Screen abandoned due to timeout.</td></tr>
</table></td></tr>
</table>

2.5 Non-Volatile RAM (NVR)

The first 40 bytes of the battery backed RAM within the RTC hardware are for system parameter storage as follows:

Byte(s)	Usage	Default
0-9	Time and Date parameters.	--
1	RTC Control Register A.	070
11	RTC Control Register B.	002
12	RTC Control Register C.	--
13	RTC Control Register D.	--
14 - 19	Time and Date when machine last used.	--
20	User RAM Checksum.	

Byte(s)	Usage	Default
21 - 22	Enter Key translation token.	1C0D
23 - 24	Forward Delete Key translation token.	2207
25 - 26	Joystick Fire Button 1 translation token.	FFFF
27 - 28	Joystick Fire Button 2 translation token.	FFFF
29 - 30	Mouse Button 1 translation token.	FFFF
31 - 32	Mouse Button 2 translation token.	FFFF
33	Mouse X direction scaling factor.	00A
34	Mouse Y direction scaling factor.	00A
35	Initial VDU mode and drive count	020
36	Initial VDU Character attributes.	007
37	Size of RAM disk in 2K blocks.	000
38	Initial system UART hardware setup byte.	0E3
39	Initial external UART hardware setup byte.	0E3
40-63	Unused	--

Locations 0 to 13 are RTC hardware registers. Refer to 3.8 for an explanation of their usage and setup values.

After power-up or upon system reset the NVR is checksummed as part of initialisation. If the lower byte of the sum is not 0AAh or if the battery voltage low bit is set in the RTC status register, then the values in the default column are loaded into their respective locations and a warning message is displayed on the VDU. Those locations without defaults (marked with '--') are not changed.

The default key token value in bytes 25 to 32 is a special value (FFFF) which signals the keyboard hardware interrupt to ignore the key press rather than to insert the key token into the buffer.

The initial VDU mode (byte 35) is used to set up by the ROS on Power-Up Initialisation based on switch settings. It is then used to setup the system status-1 channel. (DDM - bits 4 & 5. See 1.8.2 for the valid combinations.) Bits 4 and 5 of byte 35 are set up correspondingly. Bit 6 is set if two drives are fitted else it is cleared. The default version of byte 35 has bit 6 set (two drives) and bits 4 & 5 set to 1 & 0 (Colour, alpha, 80 x 25 chars).

Bit 7 of byte 35 is used to enable or disable the serial I/O flow control option. Refer to section 2.3.12 for serial I/O and flow control details.

The initial VDU character attribute (byte 36) is written to all the attribute bytes of the VDU buffer when one of the alpha modes is selected. The default value selects a white foreground on a black background.

The RAM disk size (byte 37) is used by the MS-DOS operating system to specify RAM disk setup size.

The initial UART parameters (bytes 38 and 39) specifies 9600 baud, 8 data bits, 1 stop bit and no parity. These values are loaded to their respective serial channel by the Serial I/O Initialise sub-function (See Section 2.3.12: Interrupt 20).

2.6 ROS Messages

The ROS outputs a number of messages during Power-Up Self Test initialisation as detailed below. The language in which these messages are displayed is dependent of the three option links connected

to the three least significant bits of the system printer port status. (See Table 3.1 for the interpretation of the three link bits.)

2.6.1 Non-Fatal ROS Messages

The following messages are displayed on the primary display screen (in the default display mode as specified by the NVR) in the situations as described. The initialisation process is allowed to complete even though some of them may represent self test failures.

Please wait
: This message is displayed on the top line of the screen after Power-Up or after a Soft Reset ([Ctrl]+[Alt]+[Del]) from the keyboard. A dot is displayed after it for each major hardware self test segment completed successfully.

Amstrad PC nnnK (Vv.i) Last used at hh:mm on dd mn yy
: This message is displayed after the successful completion of all self tests, where:
  nnn = the RAM size in kilobytes.
  v.i = the ROS Version (v) and Issue (i) number.
  hh:mm = the hours (hh) and the minutes (mm) of last on time.
  dd mn yy = the day (dd), the calendar month (mn) and the year (yy) of the last date used.

Please fit new batteries
: This message is displayed below the AMSTRAD PC message when it is noted that the RTC battery voltage low bit is set (indicating that there is either no battery installed or that the battery is very near to failing).

Check keyboard and mouse
: This message is displayed when the keyboard self test firmware does not respond with the test pass (0AAh).

Insert a SYSTEM disk into drive A
Then press any key
: This message set is displayed when the floppy disk bootstrap is unable to successfully read the bootstrap sector from drive A after 10 retries.

Error: External ROM checksum incorrect: ROM address = nnnnnh
: This message is displayed when the checksum on an external ROM is not zero (See section 2.1 - 16). The physical address of the ROM is displayed in five (nnnnn) hexadecimal digits.

2.6.2 Fatal ROS Messages

The following messages indicate that a self test segment has failed and that initialisation cannot continue. In this situation the machine must be switched off and on again in order to reinitiate operations. The display is switched to 80 column alpha mode and cleared prior to displaying any of these messages.

Error: Faulty SYSTEM RAM
Error: Faulty VDU RAM
Error: Faulty interrupt controller
Error: Faulty direct memory access controller
Error: Faulty floppy disk controller or disk drive
Error: Faulty interval timer
Error: Faulty system status register
Error: Faulty real time clock
Error: Faulty VDU controller

Error: Faulty system printer port
Error: Faulty system serial port
Error: Faulty ROS ROM checksum
Error: Faulty memory (parity error)

When one of these failures occurs, no other testing is run since further testing may require use of the failing component. For this reason the system is placed in a non-interruptible loop. Failures of this sort are not expected to occur even intermittently. When any self test failure does occur it should be referred to a qualified AMSTRAD service facility for further diagnostic testing.

3 Reference Information

The following tables document a number of the hardware and software features of the AMSTRAD PC1640 some of which may have been already mentioned in earlier sections but are repeated here for easy reference.

3.1 Language Links.

The lower three bits of the Printer Status Channel (I/O address 379) are wired to reflect the (one's complement) state of a set of option links (LK1 - LK3) located on the left side of the main board about 2 inches below the printer connector. They are used by the ROS firmware to define the language option or diagnostic mode option as detailed os follows:

LK1	LK2	LK3	ROS Usage
OFF	OFF	OFF	English Language.
OFF	OFF	ON	German Language.
OFF	ON	OFF	French Language.
OFF	ON	ON	Spanish Language.
ON	OFF	OFF	Danish Language.
ON	OFF	ON	Swedish Language.
ON	ON	OFF	Italian Language.
ON	ON	ON	Diagnostic Mode.

The ROS messages are displayed in the selected languge. In diagnostic mode, the messages revert to English, and the normal testing is skipped. Any self test failures are reported but are ignored and upon completion a disk bootstrap is attempted. This enables loading of an extended set of diagnostic software. The actual value observed by reading I/O address 379h is the one's complement value so that the values range (when masked with 07h) from 7 for English, 6 for German, and etc. down to 0.

3.2 Processor Memory Usage.

The following is a repeat of the processor's physical memory layout in tabular form with interrupts and ROS areas included.

Location(s)	Usage
00000 - 003FF	Processor interrupt vectors 0 to 255. To derive an individual interrupt vector's starting address multiply the vector number by four.
00400 - 00500	ROS Variables. (See section 2.4)
00501 - 9FFFF	System (or User) RAM artea. The 640K byte area (inclusive of the previous entries) is the maximum amount of system memory.
A0000 - BFFFF	128K byte area reserved for IGA video RAM and other display adapters (See 1.11 for buffers)

Location(s)	Usage
C0000 - C3FFF	IGA ROM BIOS area with executable code in the first 2000h with mode setup tables and fonts in the second 2000h area. This area of ROM (as well as the following C4000-C7FFF fonts area) can be switched off by sw10 set to ON.
C4000 - C7FFF	Amstrad Foreign Fonts for Danish, Portugese, and Greek national variants. This area can be switched off by sw9 in the OFF position.
C8000 - C9FFF	Base Hard Disk Controller ROM BIOS area.
CA000 - EFFFF	This area used by additional HD controllers and peripheral cards (such as LANs) which require support via a BIOS ROM. The Amstrad Diagnostic Pack ROM resides from E0000 to E7FFF when it is installed.
F0000 - FFFFF	64K byte area reserved for System ROM. The ROS resides in the 16K byte area from FC000 to FFFFF. The remaining 48K bytes is reserved for future expansion. The 16K byte ROS ROM repeats four times in this address range.

3.3 Keyboard and Key Codes.

Key Code	Hex	(UK) Key Cap
1	01	ESC
2	02	1 and !
3	03	2 and "
4	04	3 and £
5	05	4 and $
6	06	5 and %
7	07	6 and ^
8	08	7 and &
9	09	8 and *
10	0A	9 and (
11	0B	0 and)
12	0C	- and _
13	0D	= and +
14	0E	<-DEL
15	0F	TAB
16	10	Q
17	11	W
18	12	E
19	13	R
20	14	T
21	15	Y
22	16	U
23	17	I
24	18	O
25	19	P
26	1A	[and {
27	1B	] and }
28	1C	CR
29	1D	CTRL

Key Code	Hex	(UK) Key Cap
30	1E	A
31	1F	S
32	20	D
33	21	F
34	22	G
35	23	H
36	24	J
37	25	K
38	26	L
39	27	; and :
40	28	' and @
41	29	# and ~
42	2A	LEFT SHIFT
43	2B	\ and |
44	2C	Z
45	2D	X
46	2E	C
47	2F	V
48	30	B
49	31	N
50	32	M
51	33	, and <
52	34	. and >
53	35	/ and ?
54	36	RIGHT SHIFT
55	37	* and PRTSC
56	38	ALT
57	39	SPACE
58	3A	CAPS LOCK
59	3B	F1
60	3C	F2
61	3D	F3
62	3E	F4
63	3F	F5
64	40	F6
65	41	F7
66	42	F8
67	43	F9
68	44	F10
69	45	NUM LOCK
70	46	SCROLL LOCK
71	47	KEY PAD 7
72	48	KEY PAD 8
73	49	KEY PAD 9

Key Code	Hex	(UK) Key Cap
74	4A	KEY PAD -
75	4B	KEY PAD 4
76	4C	KEY PAD 5
77	4D	KEY PAD 6
78	4E	KEY PAD +
79	4F	KEY PAD 1
80	50	KEY PAD 2
81	51	KEY PAD 3
82	52	KEY PAD 0
83	53	KEY PAD .
84 - 111	54 - 6F	UNDEFINED
112	70	DEL ->
113 - 115	71 - 73	UNDEFINED
116	74	ENTER
117 - 118	75 - 76	UNDEFINED
119	77	JOY FIRE2
120	78	JOY FIRE1
121	79	JOY RIGHT
122	7A	JOY LEFT
123	7B	JOY DOWN
124	7C	JOY UP
125	7D	MOUSE M2
126	7E	MOUSE M1
127	7F	UNDEFINED

3.4 Asynchronous Communications Element (8250) Registers.

For serious design purposes, it is recommended that the designer obtain the standard INS8250 data sheets. The following excerpt are the major software accessible registers.

Modem Status Register (MSR) [R6] - I/O Address 3FEh.

Bit(s)	Function
7	Data Carrier Detect (DCD).
6	Ring Indicator (RI).
5	Data Set Ready (DSR).
4	Clear To Send (CTS).
3	Delta Data Carrier Detect (DDCD).
2	Trailing Edge Ring Indicator (TREI).
1	Delta Data Set Ready (DDSR).
0	Delta Clear To Send (DCTS).

Line Status Register (LSR) [R5] - I/O Address 3FDh.

Bit(s)	Function

Bit(s)	Function
7	Always Clear (0).
6	Transmitter Shift Register Empty (TSRE).
5	Transmitter Holding Register Empty (THRE).
4	Break Interrupt (BI).
3	Framing Error (FE).
2	Parity Error (PE).
1	Overrun Error (OE).
0	Data Ready (DR).

Modem Control Register (MCR) [R4] - I/O Address 3FCh.

Bit(s)	Function
7	Always Clear (0).
6	Always Clear (0).
5	Always Clear (0).
4	Loop (Diagnostic Mode).
3	Out2 (Looped to RI).
2	Out1 (Looped to DCD).
1	Request to Send (RTS) (Looped to DSR).
0	Data Terminal Ready (DTR) (Looped to CTS).

Line Control Register (MCR) [R3] - I/O Address 3FBh.

Bit(s)	Function
7	Divisor Latch Access (DLAB) (Selects Regs 0 & 1).
6	Set Break.
5	Stick Parity (Holds parity as EPS not if PEN set).
4	Even parity Select (EPS).
3	Parity Enable (PEN).
2	Number of Stop Bits (STB) (0=1 Stop Bit, 1= >1).
1	Word Length Select Bit 1 (WLS1). (0-3 = 5-8 Bits)
0	Word Length Select Bit 0 (WLS0).

Interrupt Identification Register (IIR) [R2] - I/O Address 3FAh.

Bit(s)	Function
7	Always Clear (0).
6	Always Clear (0).
5	Always Clear (0).
4	Always Clear (0).
3	Always Clear (0).
2	Interrupt ID Bit 1 (IID1).
1	Interrupt ID Bit 0 (IID0).
0	Not Interrupt Pending.

IID	Int Type
3	Rx Line Status
2	Rx Data Avail.
1	Tx H.Reg Empty.

Interrupt Enable Register (IER) [DLAB = 0:R1] - I/O Address 3F9h.

When the Divisor Access Latch Bit (Line Control Register bit 7: DLAB) is clear, inputting I/O address 3F9 reads the IER.

Bit(s)	Function
7	Always Clear (0).
6	Always Clear (0).
5	Always Clear (0).
4	Always Clear (0).
3	Modem Status (EDSSI).
2	Receiver Line Status (ELSI).
1	Transmitter Holding Register Empty (ETBEI).
0	Received Data Available (ERBAI).

Receive Buffer Register (RBR)
Transmit Holding Register (THR) [DLAB = 0:R0] - I/O Address 3F8h.

When the Divisor Access Latch Bit (Line Control Register bit 7: DLAB) is clear, reading and writing I/O location 3F8 accesses the RBR/THR registers. An input from I/O address 3F8 reads the Receiver buffer Register (bits 0 to 7). Outputting to I/O address 3F8 writes the Transmitter holding Register.

Divisor Latches MS & LS (DLL & DLM) [R0 & R1 when DLAB Set].

When the Divisor Access Latch Bit (Line Control Register bit 7: DLAB) is set, then registers 0 & 1 are the (16-bit) Divisor Register. The least significant bits are written to by outputting to address 3F8 and the most significant bits are written to by an output to location 3F9. The divisors and their respective baud rates are as follows.

Baud Rate	Divisor	R1 & R0 (hex)
75	1536	06 - 00
300	384	01 - 80
600	192	00 - C0
1200	96	00 - 60
2400	48	00 - 30
4800	24	00 - 18
9600	12	00 - 0C

3.5 High Performance Programmable DMA Controller (8237A-4) Registers.

The following are the major software accessible 8237A registers.

Command Register - Write I/O Address 008.

Bit(s)	Function (Action ... { 1 / 0 })
7	DACK sense active { hi / lo }.
6	DREQ sense active { hi / lo }.
5	{Extended/Late} write selection.
4	{Rotating/Fixed} priority.

Bit(s)	Function (Action ... { 1 / 0 })
3	{Compressed/Normal} timing.
2	{Disable/Enable} Controller.
1	{Enable/Disable} Channel 0 address hold.
0	{Enable/Disable} Memory-to-memory (not supported).

Status Register - Read I/O Address 008.

Bit(s)	Function
7	Channel 3 Request.
6	Channel 2 Request.
5	Channel 1 Request.
4	Channel 0 Request.
3	Channel 3 has reached TC.
2	Channel 2 has reached TC.
1	Channel 1 has reached TC.
0	Channel 0 has reached TC.

Mode Register - I/O Address 00B [WO].

Bit(s)	Function
7	Mode Select Bit 1. (Modes: 0 = Demand, 1 = Single,
6	Mode Select Bit 0. 2 = Block, 3 = Cascade)
5	Address {decrement/increment} select.
4	Autoinitialisation {enable/disable}.
3	Transfer Type Bit 1. (Modes: 0 = Verify, 1 = Write,
2	Transfer Type Bit 0. 2 = Read, 3 = Illegal)
1	Channel Select Bit 1. (Channels: 0-3 respectively)
0	Channel Select Bit 0.

Request Register - I/O Address 009h [WO].

Bit(s)	Function
7	Don't Care.
6	Don't Care.
5	Don't Care.
4	Don't Care.
3	Don't Care.
2	Request Bit {Set/Reset}.
1	Channel Select Bit 1. (Channels: 0-3 respectively)
0	Channel Select Bit 0.

Mask Set/Reset Register - I/O Address 00Ah [WO].

Bit(s)	Function
7	Don't Care.
6	Don't Care.

Bit(s)	Function
5	Don't Care.
4	Don't Care.
3	Don't Care.
2	{Set/Reset} Mask Bit.
1	Channel Select Bit 1. (Channels: 0-3 respectively)
0	Channel Select Bit 0.

Mask Write Register - I/O Address 00F [WO].

Bit(s)	Function
7	Don't Care.
6	Don't Care.
5	Don't Care.
4	Don't Care.
3	{Set/Clear} Channel 3 Mask Bit.
2	{Set/Clear} Channel 2 Mask Bit.
1	{Set/Clear} Channel 1 Mask Bit.
0	{Set/Clear} Channel 0 Mask Bit.

3.6 Programmable Interrupt Controller (8259A-2) Command Words.

The Initialisation Command Word (ICW) sequence is as follows:

Initialisation Command Word 1 (ICW1) - Write I/O Address 020h.

Bit(s)	Function (Action ... { 1/0 })
7	N/A.
6	N/A.
5	N/A.
4	Always Set (1)
3	{Level/Edge} Trigger Mode.
2	Call Address Interval of {4/8}.
1	{Single/Cascade} Mode (Need ICW3 if Single Mode).
0	ICW4 {Needed/Not Needed}.

Initialisation Command Word 2 (ICW2) - Write I/O Address 021h.

Bit(s)	Function (Action ... { 1/0 })
7	Interrupt Type Bit 7 (T7).
6	Interrupt Type Bit 6 (T6).
5	Interrupt Type Bit 5 (T5).
4	Interrupt Type Bit 4 (T4).
3	Interrupt Type Bit 3 (T3).
2	Not used.

Bit(s)	Function (Action ... { 1/0 })
1	Not used.
0	Not used.

This byte selects one of the interrupt service vector locations (in absolute locations 0 through 3FF) to be used when interrupting. Type bits 3 - 7 (asserted on the data bus during the INTA cycle) map to address bits 5 - 9 for interrupt vector selection. The lower three type bits are derived from the interrupt level.

Initialisation Command Word 3 (ICW3) - Write I/O Address 021h.

This command word is not used since Single (ICW1 bit 1) is always true in the PC1640. When used, this command word specifies which IR has a slave in Master mode, or it a slave then bits 0 through 3 specify the slave ID number (0 to 7).

Bit(s)	Function (Action ... { 1/0 })
7	Always clear (0).
6	Always clear (0).
5	Always clear (0).
4	{Enable/Disable} Special Fully Nested Mode.
3	Buffered Mode {On/Off}.
2	{Master/Slave} Mode (Only valid in Buffered Mode).
1	{Auto/Normal} EOI.
0	Always set (1) - (8086/8088 Mode).

Operation Control Words

The operation control words select various 8259A modes of operation.

Operation Control Word 1 (OCW1) - Write I/O Address 021.

Bit(s)	Function (Action ... { 1/0 })
7	Interrupt Mask 7 {Set/Reset}.
6	Interrupt Mask 6 {Set/Reset}.
5	Interrupt Mask 5 {Set/Reset}.
4	Interrupt Mask 4 {Set/Reset}.
3	Interrupt Mask 3 {Set/Reset}.
2	Interrupt Mask 2 {Set/Reset}.
1	Interrupt Mask 1 {Set/Reset}.
0	Interrupt Mask 0 {Set/Reset}.

The eight mask bits either mask (i.e. inhibit when M=1) or enable their respective channels.

Operation Control Word 2 (OCW2) - Write I/O Address 020h.

Bit(s)	Function
7	Rotate (R) Bit.
6	Specific (SL) Bit.
5	End of Interrupt (EOI) bit.
4	Always zero.

Bit(s)	Function
3	Always zero.
2	Level bit 2 (L2).
1	Level bit 1 (L1).
0	Level bit 0 (L0).

The level bits are required when specific (SL) is set.

Operation Control Word 2 (OCW2) - Write I/O Address 020h.

Bit(s)	Function (Action ... { 1/0 })
7	Always zero.
6	Enable Special Mask Mode (ESMM) bit.
5	Special Mask Mode (SMM) {Set/Reset}.
4	Always zero.
3	Always set.
2	{Enable/Disable} Poll Command.
1	Read Register (RR) enable bit.
0	Read {IS/IR} register on next -RD pulse (RIS).

The ESMM bit must be set for the SMM bit to have any effect. Similarly the RR bit must be set for the RIS bit to have an effect.

3.7 Programmable Interval Timer (8253) Registers.

The 8253 PIT has four addressable elements, the three counters (0 - 2) which are read or written 8 bits at a time (on I/O addresses 40h - 42h) and the Control Word register (write I/O address 043h).

Bit(s)	Function (Action ... { 1/0 })
7	Select Counter bit 1 (SC1).
6	Select Counter bit 0 (SC0).
5	Read/Load bit 1 (RL1).
4	Read/Load bit 0 (RL0).
3	Mode bit 2 (M2).
2	Mode bit 1 (M1).
1	Mode bit 0 (M0).
0	{Enable/Disable} Binary Coded Decimal (BCD) counter.

The SC bits select counters 0-2 and the 3 (both bits set) state is illegal.

The RL bits enable the counter's Read/Load operation as follows:

0:
    Counter Latching - Snapshot current counter (to a holding register) for next read operation.
1:
    Read/Load MS byte only.
2:
    Read/Load LS byte only.

3:
Read/Load LS byte first then the MS byte.

The Mode bits select one of five valid modes (six & seven wrap around to modes two and three). The modes are as follows:

0:
Interrupt on Terminal count.
1:
Programmable One-Shot.
2:
Rate Generator.
3:
Square Wave generator..
4:
Software Triggered Strobe.
5:
Hardware triggered Strobe.

3.8 Real Time Clock (HD146818) Registers.

The HD146818 is a CMOS peripheral device which combines three unique features: a complete time-of-day clock with an alarm and one hundred year calendar, a programmable periodic interrupt and square-wave generator, and 50 bytes of low-power static RAM.

The figure below shows the address map of the HD146818. The memory consists of 50 bytes of general purpose RAM, 10 RAM bytes which normally contain the time, calendar, and alarm data, and four control and status bytes. All bytes are directly readable and writable by the processor except Registers C and D which are read only. Bit 7 of Register A and the seconds byte are also read only.

0	Seconds	00
1	Sec Alarm	01
2	Minutes	02
3	Min Alarm	03
4	Hours	04
5	Hr Alarm	05
6	Day of Wk	06
7	Day of Mo	07
8	Month	08
9	Year	09
10	Register A	0A
11	Register B	0B
12	Register C	0C
13	Register D	0D
14	50 Bytes User RAM	0E

3.8.1 Time, Calendar and Alarm Locations

The processor obtains time and calendar information by reading the appropriate locations. The program may initialise the time, calendar and alarm by writing these locations. The contents of the 10 time, calendar and alarm bytes may either be binary or binary-coded decimal (BCD).

Before initialising the internal registers the SET bit in register B should be set to a "1" to prevent time/calendar updates from occurring. The program initialises the 10 locations in the selected format (binary or BCD), then indicates the format in the data mode (DM) bit of register B. All 10 locations must use the fame data mode, either binary or BCD. The SET bit may now be cleared to allow updates. Once initialised the real-time clock makes all updates in the selected data mode. The data mode cannot be changed without reinitialising the 10 data bytes.

The table below shows the binary and BCD formats of the time, calendar and alarm locations.

Address	Function	Range	Binary Data Mode	BCD Data Mode
0	Seconds	0-59	00h-3Bh	00h-59h
1	Sec Alarm	0-59	00h-3Bh	00h-59h
2	Minutes	0-59	00h-3Bh	00h-59h
3	Min Alarm	0-59	00h-3Bh	00h-59h
4	Hours		01h-0Ch (AM)	01h-12h (AM)
	12-Hr Mode	1-12	81h-8Ch (PM)	81h-92h (PM)
	24-Hr Mode	0-23	00h-17h	00h-23h
5	Hrs Alarm		01h-0Ch (AM)	01h-12h (AM)
	12-Hr Mode	1-12	81h-8Ch (PM)	81h-92h (PM)
	24-Hr Mode	0-23	00h-17h	00h-23h
6	Day of Wk	1-7	01h-07h	01h-07h
7	Day of Mon	1-31	01h-1Fh	01h-31h
8	Month	1-12	01h-0Ch	01h-12h
9	Year	0-99	00h-63h	00h-99h

For the Day of the Week, Sunday = 1.

The 24/12 bit in register B establishes whether the hour locations represent 1-to-12 or 0-to-23. The 24/12 bit cannot be changed without reinitialising the hour locations. When the 12-hour format is selected the high-order bit of the hours represents PM when it is a "1". The time, calendar and alarm bytes are not always accessible by the processor. Once per second the 10 bytes are switched to the update logic to be advanced by one second and to check for an alarm condition. If any of the 10 locations are read at this time, the data outputs are undefined. The update-in-progress (UIP) bit in Register A may be used to determine if the update cycle is in progress or not. The UIP bit goes high once a second and the update cycle begins 244 uS later. Therefore, if a "0" is read on the UIP bit, the user has at least 244 uS before the time/calendar data will be changed.

3.8.2 RTC Register Locations

The HD 146818 has four registers which are accessible to the processor. The four registers are fully accessible during the update cycle.

The bit assignments for Register A (address 0Ah) are as follows:

Bit	Assignment
7	Update In Progress (UIP)
6	Divider Bit 2 (DV2)
5	Divider Bit 1 (DV1)
4	Divider Bit 0 (DV0)
3	Rate Selection Bit 3 (RS3)
2	Rate Selection Bit 2 (RS2)
1	Rate Selection Bit 1 (RS1)
0	Rate Selection Bit 0 (RS0)

The UIP bit indicates whether the 10 time, calendar and alarm bytes are being updated or not as explained above.

The three Divider bits (DV2-DV0) are used to identify which of the three time base frequencies is in use or to reset the divider chain.

The four rate selection bits (RS3-RS0) select one of 15 taps on the 22-stage divider chain, or disable the divider output. The tap selected may be used to generate an output on the square (SQW) pin and/or a periodic interrupt.

The bit assignments for Register B (address 0Bh) are as follows:

Bit	Assignment
7	SET Bit
6	Periodic Interrupt Enable (PIE) Bit
5	Alarm Interrupt Enable (AIE) Bit
4	Update-ended Interrupt Enable (UIE) Bit
3	Square-Wave Enable (SQWE) Bit
2	Data Mode (DM) Bit
1	12/12 hour format Bit
0	Daylight Savings Enable (DSE) Bit

When the SET bit is a "0" the update cycle functions normally by advancing the counts once per second. When the SET bit is written to a "1", any update cycle in progress is aborted and the processor may initialise the time and calendar locations without updates occurring. SET is a read/write bit which is not modified by -RES or internal functions of the HD146818.

The PIE bit is a read/write bit which allows the periodic-interrupt (PF) bit to cause the -IRQ pin to be driven low. The program writes a "1" to the PIE bit in order to receive periodic interrupts at the rate specified by the RS3 RS0 bits in Register A. A "0" in PIE blocks 'IRQ from being generated, bu the periodic flag (PF) bit still goes high at the periodic rate.

The AIE bit is a read/write bit which when set to "1" permits the alarm flag (AF) to assert -IRQ. An alarm interrupt occurs for each second that the three time bytes equal the three alarm bytes. When AIE is a "0" the AF bit does not initiate an -IRQ. The -RES pin clears AIE to "0". The internal functions do not affect the AIE bit.

The UIE bit is a read/write bit which enables the update-end flag (UF) bit to assert -IRQ. The -RES pin

going low or the SET bit going high clears the UIE bit.

When the SQWE bit is set to a "1" by the processor, a square-wave signal at the frequency specified by the rate selection bits (RS3 to RS0) appears on the SQW pin. When the SQWE bit is set to "0" the SQW pin is held low. The SQWE bit is cleared by the -RES pin. SQWE is a read/write bit.

The DM bit indicates whether time and calendar updates are to use binary or BCD format. DM is a read/write bit and is not modified by -RES or internal functions of the HD146818. A "1" in DM signifies binary data and a "0" specifies BCD data mode.

The 24/12 control bit specifies the format of the hour bytes. A "1" specifies 24-hour mode and a "0" specifies 12-hour mode. It is a read/write bit and is not affected by -RES or any HD146818 internal functions.

The DSE bit is a read/write bit which when set to "1" enables daylight savings mode. When enabled, two special updates take place. On the last sunday in April the time increments from 1:59:59 to 3:00:00 AM. On the last sunday in October when the time first reaches 1:59:59 AM is decremented to 1:00:00 AM. DSE is not changed by -RES or any internal operations.

The bit assignments for Register C (address 0Ch) are as follows:

Bit	Assignment
7	Interrupt Request Flag (IRQF) Bit
6	Periodic Interrupt Flag (PF) Bit
5	Alarm Interrupt Flag (AF) Bit
4	Update-Ended Interrupt Flag (UF) Bit
3-0	0

The C register is a read-only register and a program write has no effect any of the bits.

The IRQF bit is set by the logical equation: IRQF = PF·PIE + AF·AIE + UF·UIE. Any time the IRQF bit is a "1", the $\overline{\text{IRQ}}$ pin is driven low. All flag bits in the C register are cleared after a program read or when the -RES pin is low.

The PF bit is set to "1" when a particular edge is detected in the selected tap of the divider chain as selected by the RS3 to RS0 bits. The PF bit is set to a "1" independent of the state the PIE bit.

The AF bit is set to a "1" when the current time matches the alarm time.

The UF bit is set after each update cycle.

The remaining bits (3 to 0) are always low.

The bit assignments for Register D (address 0Dh) are as follows:

Bit	Assignment
7	Valid RAM Time (VRT) Bit
6 - 0	0

The VRT bit indicates that the contents of the RAM and time are valid. A "0" appears in the VRT bit when the power sense (PS) pin is low. The processor can set the VRT bit when the time and calendar are

initialised to indicate that they are valid. The VRT bit is a read-only bit and is not modified by the -RES pin. The VRT bit can only be set by reading the D register.

Bits 6 to 0 are unused and are always read as zeroes.

3.9 Floppy Disk Controller (uPD765A).

The uPD765A Floppy Disk Controller (FDC) contains two registers which are accessible to the CPU; the Main Status Register (at I/O address 03F4h) and the Data Register (at I/O address 03F5h) both of which are 8 bits wide. The Status register contains the status of the FDC and may be accessed at any time. The Data Register is actually made up of several registers in a stack and stores data, commands and Floppy Disk Drive (FDD) status information. Data is written into the data register in order to program a particular command. The data address is read in order to obtain the result after an operation. The Main Status register (I/O address 3F4h) may only be read and is used to facilitate the transfer of data between the CPU and the uPD765A FDC.

There are 15 separate commands which the uPD765A FDC can execute. Each of these commands require multiple bytes to fully specify the operation. The result after execution of the command may also be a multi-byte transfer back to the processor. Because of this multi-byte interchange of information between the processor and the FDC, it is convenient to consider each command as consisting of three phases:

Command Phase:
: The FDC accepts all information to perform a particular operation from the CPU.

Execution Phase:
: The FDC performs the operation.

Result Phase:
: After completion of the operation, status and housekeeping information are made available to the CPU.

The uPD765A contains five status registers. The main status register mentioned earlier which may be read at any time and four result phase status registers (ST0, ST1, ST2 and ST3) which are only made available during the Result Phase after completion of a command. The particular command which has been executed determines which status registers will be returned.

The Command bytes which are sent to the uPD765A during the Command Phase must occur in the order shown in the command table. That is, the command code must be sent first followed by the other bytes in the prescribed sequence. No foreshortening of the Command Phase or the Result Phase is allowed. After the last byte of data in the Command Phase is sent the Execution Phase automatically starts. In a similar fashion, when the last byte of data is read out in the Result Phase, the command is automatically ended and the uPD765A is ready for a new command.

It is important to note that during the Result phase all bytes shown in the Command table must be read. The Read Data command, for example has seven bytes listed in the result phase. All seven bytes must be read out else a new command will not be accepted.

The status registers as follows:

Main Status Register

Bit(s)	Function

Bit(s)	Function
7	Request for Master (RQM).
6	Data Input/Output (DIO).
5	Execution Mode (EXM).
4	FDC Busy (CB).
3	FDD 3 Busy (D3B).
2	FDD 2 Busy (D2B).
1	FDD 1 Busy (D1B).
0	FDD 0 Busy (D0B).

Status Register 0 (ST0)

Bit(s)	Function
7	Interrupt Code bit 1 (IC1).
6	Interrupt Code bit 2 (IC2).
5	Seek End (SE).
4	Equipment Check (EC).
3	Not Ready (NR).
2	Head Address (HD).
1	Unit Select 1 (US1).
0	Unit Select 2 (US2).

Status Register 1 (ST1)

Bit(s)	Function
7	End of Cylinder (EN).
6	Always zero.
5	Data Error (DE).
4	Over Run (OR).
3	Always zero.
2	No Data (ND).
1	Not Writable (NW).
0	Missing Address Mark (MA).

Status Register 2 (ST2)

Bit(s)	Function
7	Always zero.
6	Control Mark (CM).
5	Data Error in Data Field (DD).
4	Wrong Cylinder (WC).
3	Scan Equal Hit (SH).
2	Scan Not Satisfied (SN).
1	Bad Cylinder (BC).
0	Missing Address Mark in Data Field (MD).

Status Register 3 (ST3)

Bit(s)	Function
7	Fault (FT).
6	Write Protect (WP).
5	Ready (RY).
4	Track 0 (T0).
3	Two Side (TS).
2	Head Address (HD).
1	Unit Select 1 (US1).
0	Unit Select 0 (US0).

The Commands are as follows:

Read Data

Command Phase: 9 bytes.

Byte 1: Command Code.

Bit(s)	Function (Action ... { 1 / 0 })
7	(MT) Multi-Track {Enable/Disable}.
6	(FM) Select {MFM/FM} (Single/Dobule density) Mode.
5	(SK) Enable Skip deleted data address mark.
4	0.
3	0.
2	1.
1	1.
0	0.

Byte 2: Head and Unit select.

Bit(s)	Function
7-3	Don't Care.
2	HD - Head Select (0 or 1).
1	US1.
0	US0 - Unit Select (0 or 1).

Byte 3: Cylinder Number (0-76).

Byte 4: Head Number (as specified in the ID field).

Byte 5: Sector to be read.

Byte 6: Number of bytes per sector.

Byte 7: EOT - Final sector number on track.

Byte 8: GPL - Gap 3 Length.

Byte 9: DTL - Data Length to be read.

During execution data is transferred between the FDD and the CPU memory.

The Result phase returns 7 bytes:

Byte 1: ST0 - Status register 0 (See ST0 table).

Byte 2: ST1 - Status register 1 (See ST1 table).

Byte 3: ST2 - Status register 2 (See ST2 table).

Byte 4: Final Cylinder number.

Byte 5: Final head read.

Byte 6: Final Sector read.

Byte 7: Number of bytes read.

Read Track

Command Phase: 9 bytes.

Byte 1: Command Code.

Bit(s)	Function (Action ... { 1 / 0 })
7	0.
6	(FM) Select {MFM/FM} (Single/Dobule density) Mode.
5	(SK) Enable Skip deleted data address mark.
4	0.
3	0.
2	0.
1	1.
0	0.

Byte 2: Head and Unit select.

Bit(s)	Function
7-3	Don't Care.
2	HD - Head Select (0 or 1).
1	US1.
0	US0 - Unit Select (0 or 1).

Byte 3: Cylinder Number (0-76).

Byte 4: Head Number (as specified in the ID field).

Byte 5: Sector to be read.

Byte 6: Number of bytes per sector.

Byte 7: EOT - Final sector number on track.

Byte 8: GPL - Gap 3 Length.

Byte 9: DTL - Data Length to be read.

During execution data is transferred between the FDD and the CPU memory. The FDC reads all data fields from index hole to EOT.

The Result phase returns 7 bytes:

Byte 1: ST0 - Status register 0 (See ST0 table).

Byte 2: ST1 - Status register 1 (See ST1 table).

Byte 3: ST2 - Status register 2 (See ST2 table).

Byte 4: Final Cylinder number.

Byte 5: Final head read.

Byte 6: Final Sector read.

Byte 7: Number of bytes read.

Read Deleted Data

Command Phase: 9 bytes.

Byte 1: Command Code.

Bit(s)	Function (Action ... { 1 / 0 })
7	(MT) Multi-Track {Enable/Disable}.
6	(FM) Select {MFM/FM} (Single/Dobule density) Mode.
5	(SK) Enable Skip deleted data address mark.
4	0.
3	1.
2	1.
1	0.
0	0.

Byte 2: Head and Unit select.

Bit(s)	Function
7-3	Don't Care.
2	HD - Head Select (0 or 1).
1	US1.
0	US0 - Unit Select (0 or 1).

Byte 3: Cylinder Number (0-76).

Byte 4: Head Number (as specified in the ID field).

Byte 5: Sector to be read.

Byte 6: Number of bytes per sector.

Byte 7: EOT - Final sector number on track.

Byte 8: GPL - Gap 3 Length.

Byte 9: DTL - Data Length to be read.

During execution data is transferred between the FDD and the CPU memory.

The Result phase returns 7 bytes:

Byte 1: ST0 - Status register 0 (See ST0 table).

Byte 2: ST1 - Status register 1 (See ST1 table).

Byte 3: ST2 - Status register 2 (See ST2 table).

Byte 4: Final Cylinder number.

Byte 5: Final head read.

Byte 6: Final Sector read.

Byte 7: Number of bytes read.

Read ID

Command Phase: 9 bytes.

Byte 1: Command Code.

Bit(s)	Function (Action ... { 1 / 0 })
7	0.
6	(FM) Select {MFM/FM} (Single/Dobule density) Mode.
5	0.
4	0.
3	1.
2	0.
1	1.
0	0.

Byte 2: Head and Unit select.

Bit(s)	Function
7-3	Don't Care.
2	HD - Head Select (0 or 1).
1	US1.

Bit(s)	Function
0	US0 - Unit Select (0 or 1).

During execution the first correct ID information on the cylinder is stored in the Data Register.

The Result phase returns 7 bytes:

Byte 1: ST0 - Status register 0 (See ST0 table).

Byte 2: ST1 - Status register 1 (See ST1 table).

Byte 3: ST2 - Status register 2 (See ST2 table).

Byte 4: Cylinder.

Byte 5: head.

Byte 6: Sector.

Byte 7: Number of bytes per sector.

Write Data

Command Phase: 9 bytes.

Byte 1: Command Code.

Bit(s)	Function (Action ... { 1 / 0 })
7	(MT) Multi-Track {Enable/Disable}.
6	(FM) Select {MFM/FM} (Single/Dobule density) Mode.
5	0.
4	0.
3	0.
2	1.
1	0.
0	1.

Byte 2: Head and Unit select.

Bit(s)	Function
7-3	Don't Care.
2	HD - Head Select (0 or 1).
1	US1.
0	US0 - Unit Select (0 or 1).

During execution data is transferred between the cpu memory and the FDD.

The Result phase returns 7 bytes:

Byte 1: ST0 - Status register 0 (See ST0 table).

Byte 2: ST1 - Status register 1 (See ST1 table).

Byte 3: ST2 - Status register 2 (See ST2 table).

Byte 4: Final Cylinder number.

Byte 5: Final head written.

Byte 6: Final Sector written.

Byte 7: Number of bytes written.

Write Deleted Data

Command Phase: 9 bytes.

Byte 1: Command Code.

Bit(s)	Function (Action ... { 1 / 0 })
7	(MT) Multi-Track {Enable/Disable}.
6	(FM) Select {MFM/FM} (Single/Dobule density) Mode.
5	0.
4	0.
3	1.
2	0.
1	0.
0	1.

Byte 2: Head and Unit select.

Bit(s)	Function
7-3	Don't Care.
2	HD - Head Select (0 or 1).
1	US1.
0	US0 - Unit Select (0 or 1).

Byte 3: Cylinder Number (0-76).

Byte 4: Head Number (as specified in the ID field).

Byte 5: Sector.

Byte 6: Number of bytes per sector.

Byte 7: EOT - Final sector number on track.

Byte 8: GPL - Gap 3 Length.

Byte 9: DTL - Data Length to be written.

During execution data is transferred between the CPU memory and the FDD.

The Result phase returns 7 bytes:

Byte 1: ST0 - Status register 0 (See ST0 table).

Byte 2: ST1 - Status register 1 (See ST1 table).

Byte 3: ST2 - Status register 2 (See ST2 table).

Byte 4: Final Cylinder number.

Byte 5: Final head written.

Byte 6: Final Sector written.

Byte 7: Number of bytes written.

Format Track

Command Phase: 6 bytes.

Byte 1: Command Code.

Bit(s)	Function (Action ... { 1 / 0 })
7	0.
6	(FM) Select {MFM/FM} (Single/Dobule density) Mode.
5	0.
4	0.
3	1.
2	1.
1	0.
0	1.

Byte 2: Head and Unit select.

Bit(s)	Function
7-3	Don't Care.
2	HD - Head Select (0 or 1).
1	US1.
0	US0 - Unit Select (0 or 1).

Byte 3: Number of bytes per sector.

Byte 4: Number of sectors per track.

Byte 5: GPL - Gap 3 Length.

Byte 6: D - Filler Byte.

During execution the FDC writes address headers to the entire track.

The Result phase returns 7 bytes:

Byte 1: ST0 - Status register 0 (See ST0 table).

Byte 2: ST1 - Status register 1 (See ST1 table).

Byte 3: ST2 - Status register 2 (See ST2 table).

Byte 4: Cylinder number.

Byte 5: Head.

Byte 6: Sector.

Byte 7: Number of bytes per sector.

Scan Equal

Command Phase: 9 bytes.

Byte 1: Command Code.

Bit(s)	Function (Action ... { 1 / 0 })
7	(MT) Multi-Track {Enable/Disable}.
6	(FM) Select {MFM/FM} (Single/Dobule density) Mode.
5	(SK) Enable Skip deleted data address mark.
4	1.
3	0.
2	0.
1	0.
0	1.

Byte 2: Head and Unit select.

Bit(s)	Function
7-3	Don't Care.
2	HD - Head Select (0 or 1).
1	US1.
0	US0 - Unit Select (0 or 1).

Byte 3: Cylinder Number (0-76).

Byte 4: Head Number (as specified in the ID field).

Byte 5: Sector.

Byte 6: Number of bytes per sector.

Byte 7: EOT - Final sector number on track.

Byte 8: GPL - Gap 3 Length.

Byte 9: STP - Step Factor: 1 = Contiguous: 2 = Alternate Sectors.

During execution data is transferred from the CPU memory and compared with data from the FDD.

The Result phase returns 7 bytes:

Byte 1: ST0 - Status register 0 (See ST0 table).

Byte 2: ST1 - Status register 1 (See ST1 table).

Byte 3: ST2 - Status register 2 (See ST2 table).

Byte 4: Final Cylinder number.

Byte 5: Final head compared.

Byte 6: Final Sector compared.

Byte 7: Number of bytes compared.

Scan Low or Equal

Command Phase: 9 bytes.

Byte 1: Command Code.

Bit(s)	Function (Action ... { 1 / 0 })
7	(MT) Multi-Track {Enable/Disable}.
6	(FM) Select {MFM/FM} (Single/Dobule density) Mode.
5	(SK) Enable Skip deleted data address mark.
4	1.
3	1.
2	0.
1	0.
0	1.

Byte 2: Head and Unit select.

Bit(s)	Function
7-3	Don't Care.
2	HD - Head Select (0 or 1).
1	US1.
0	US0 - Unit Select (0 or 1).

Byte 3: Cylinder Number (0-76).

Byte 4: Head Number (as specified in the ID field).

Byte 5: Sector to be compared.

Byte 6: Number of bytes per sector.

Byte 7: EOT - Final sector number on track.

Byte 8: GPL - Length of Gap 3.

Byte 9: STP - Step Factor: 1 = Contiguous: 2 = Alternate Sectors.

During execution data from the CPU memory is compared with data from the FDD.

The Result phase returns 7 bytes:

Byte 1: ST0 - Status register 0 (See ST0 table).

Byte 2: ST1 - Status register 1 (See ST1 table).

Byte 3: ST2 - Status register 2 (See ST2 table).

Byte 4: Final Cylinder number.

Byte 5: Final head compared.

Byte 6: Final Sector compared.

Byte 7: Number of bytes compared.

Scan High or Equal

Command Phase: 9 bytes.

Byte 1: Command Code.

Bit(s)	Function (Action ... { 1 / 0 })
7	(MT) Multi-Track {Enable/Disable}.
6	(FM) Select {MFM/FM} (Single/Dobule density) Mode.
5	(SK) Enable Skip deleted data address mark.
4	1.
3	1.
2	1.
1	0.
0	1.

Byte 2: Head and Unit select.

Bit(s)	Function
7-3	Don't Care.
2	HD - Head Select (0 or 1).
1	US1.
0	US0 - Unit Select (0 or 1).

Byte 3: Cylinder Number (0-76).

Byte 4: Head Number (as specified in the ID field).

Byte 5: Sector to be compared.

Byte 6: Number of bytes per sector.Byte 7: EOT - Final sector number on track.

Byte 8: GPL - Length of Gap 3.

Byte 9: STP - Step Factor: 1 = Contiguous: 2 = Alternate Sectors.

During execution data from the CPU memory is compared with data from the FDD.

The Result phase returns 7 bytes:

Byte 1: ST0 - Status register 0 (See ST0 table).

Byte 2: ST1 - Status register 1 (See ST1 table).

Byte 3: ST2 - Status register 2 (See ST2 table).

Byte 4: Final Cylinder number.

Byte 5: Final head compared.

Byte 6: Final Sector compared.

Byte 7: Number of bytes compared.

Recalibrate

Command Phase: 2 bytes.

Byte 1: Command Code.

Bit(s)	Function (Action ... { 1 / 0 })
7	0.
6	0.
5	0.
4	0.
3	0.
2	1.
1	1.
0	1.

Byte 2: Head and Unit select.

Bit(s)	Function
7-3	Don't Care.
2	0.
1	US1.
0	US0 - Unit Select (0 or 1).

During execution phase, the Head is retracted to Track zero.

No status information is returned during the result phase.

Sense Interrupt Status

Command Phase: 1 byte.

Byte 1: Command Code = 08h.

The Result phase returns two bytes:

Byte 1: ST0 - Status Register 0.

Byte 2: PCN - Present Cylinder Number.

Specify

Command Phase: 3 bytes.

Byte 1: Command Code = 03h.

Byte 2: SRT/HUT - Step Rate Time (4 MS bits - in 1 ms increments)/Head Unload Time (4 LS bits - in 16 ms increments).

Byte 3: HLT/ND - Head Load Time (Bits 1 to 7 - in 2 ms increments)/Non-DMA Mode (Bit 0).

Seek

Command Phase: 3 bytes.

Byte 1: Command Code = 0Fh.

Byte 2: Head and Unit select.

Bit(s)	Function
7-3	Don't Care.
2	HD - Head Number.
1	US1.
0	US0 - Unit Select (0 or 1).

Byte 3: New Cylinder Number.

During execution phase, the Head is positioned to the specified Cylinder.

No status information is returned during the result phase.

Sense Drive Status

Command Phase: 2 bytes.

Byte 1: Command Code = 04h.

Byte 2: Head and Unit select.

Bit(s)	Function

Bit(s)	Function
7-3	Don't Care.
2	0.
1	US1.
0	US0 - Unit Select (0 or 1).

Result phase: 1 byte.

Byte 1: ST3 - Status Register 3.

Invalid Opcodes

All command codes not listed above are considered invalid. When an invalid code is encountered the FDC returns the ST0 register with the MS bit (Invalid Opcode bit) set.

Section 2 Index Appendices

Appendix 1: Mouse Software Interfaces

The operating system supplied with your PC1640, MS-DOS 3.2, supports a set of Mouse functions which allow a program to access the mouse and control the cursor. These functions can be called from within your application programs by using the software interfaces described below.

The mouse functions described in this appendix are as follows.

Number	Function
0	Mouse Initialisation
1	Show Cursor
2	Hide Cursor
3	Get Mouse Position and Button Status
4	Set Mouse Cursor Position
5	Get Button Press Information
6	Get Button Release Information
7	Set Minimum and Maximum X-Cursor Position
8	Set Minimum and Maximum Y-Cursor Position
9	Set Graphics Cursor Block
10	Set Text Cursor
11	Read Mouse Motion Counters
12	Set User Defined Subroutine Input Mask
13	Light Pen Emulation Mode On
14	Light Pen Emulation Mode Off
15	Set Mickey/Pixel Ratio
16	Conditional Off
19	Set Double Speed Threshold

The program MOUSE.COM must be loaded by either typing "MOUSE" at the keyboard or by having a "MOUSE" line in your AUTOEXEC.BAT.

When MOUSE.COM is loaded, it performs an initialisation process and installs the mouse driver software into the system. Once installed, the mouse driver remains permanently resident until the next time you bootstrap your computer. After successfully completing initialisation, the following message is output to the display:

"--- Installing Mouse Device Driver V5.00 ---"

If the mouse driver fails to load you will get one of the two following messages:

1. "MOUSE: Mouse Driver already installed."
   Because either you've previously installed MOUSE.COM or there's some other resident utility which has chained itself onto the vector needed by MOUSE.COM's resident driver.
2. "MOUSE: Amstrad Mouse not found."
   Because either there's a hardware fault or your hardware isn't an Amstrad PC.

During initialisation, the following actions take place:

1. The hardware ticker routine residing at software interrupt 8 which is invoked every 54ms is replaced by a Mouse Ticker routine that is invoked every 18ms.
2. Counter 0 of the 8253 interrupt controller is re-programmed so that it produces an interrupt every 18ms rather than every 54ms. A word of caution here, programs which chain themselves into the ticker interrupt may get 1overly excited1 by the unusually high interrupt rate.
3. The Mouse Buttons Interrupt routine is inserted into the Mouse Buttons Interrupt vector (software interrupt 6).
4. The Amstrad PC Mouse X and Y movement registers are initialised to zero. The Mouse Buttons are both marked as being released.
5. The Non Volatile RAM is read to determine the X & Y Scaling factors, which are to be used during cursor key generation for mouse movement in text mode.
6. The Mouse driver is initialised to be in Text Mode.

The general procedure for making an assembly language program call to the mouse function driver program is:

1. Load the specified register parameters.
2. Execute software interrupt 51 (033h).

The cursor coordinates required for the various function calls are in the form of X-Cursor (horizontal) and Y-Cursor (vertical) values. The range of the X-Cursor is always the full 0 to 639 points of the high resolution graphics screen and the Y-Cursor ranges from 0 to 199. This coordinate system defines the "virtual" screen and when in modes with less resolution than 640 points then the least significant bits of the X-Cursor are

ignored. In 4-Colour (320 x 200) graphics only even values are significant while in 80 column text mode only every eighth position is valid and in 40 column text modes only every 16th position is valid. Supplied values are rounded to the nearest values permitted for the current screen mode.

The standard unit of mouse motion is called the "mickey" and is equal to approximately 1/200 of an inch. See Mouse function 15 which sets the mickey to pixel ratios.

In Text Mode, mouse movement will cause cursor key tokens to be inserted into the keyboard buffer. The scaling factors read from the NVR during initialisation are used to determine how many units of mouse movement are to be sensed before a single cursor key token is inserted into the keyboard buffer. Invoking any Mouse Function except Function 0 or Function 2 will disable this extra mode (i.e. cursor movement tokens are not generated). Invoking Function 0 (Initialisation) enables this extra mode (i.e. cursor tokens are generated) and invoking Function 2 (Hide Cursor) does not change the current mode.

In Text Mode, the mouse buttons interrupt routine (interrupt 6) translates the Left and Right mouse buttons into the appropriate scan codes which are held in NVR bytes 29-30 (for Left) and 31-32 (for Right). The default NVR value for these scan codes is the ignore code (all F's). The NVRPATCH programs can be used to set the mouse button codes to handy values such as CR and ESC.

In Graphics Mode, the mouse buttons interrupt routine translates the Right mouse button to the shift key token, and the Left button is passed through as a mouse event to the user defined subroutine. (See mouse function 12.)

Function 0: Mouse Initialisation.

This function initialises the mouse driver and returns the current status of the mouse hardware and software.

CPU registers are used as follows:

Entry:
AX = 0
Exit:
AX = Mouse Status
BX = Number of Buttons.

All Flags and other registers preserved.

Since the mouse hardware is verified by power-up testing the driver always returns a mouse status of true (-1). If the mouse driver is not resident then AX is returned as false (0).

The mouse driver parameters are reset to the following values:

Parameter	Value
Cursor Flag	Hidden (-1)
Cursor Position	Center Screen
Graphics Cursor	Arrow
Hot Spot	-1, -1
Text Cursor	Inverting box
User Defined Call Mask	Zeros
Light Pen Emulation Mode	Enabled
Mickey to X-Pixel Ratio	8
Mickey to Y-Pixel Ratio	16
Min/Max X-Cursor Position	0/639
Min/Max Y-Cursor Position	0/199

The mouse X and Y hardware counters are reset and a number of internal software counters are zeroed.

The mouse driver is initialised to be in Text Mode (and cursor tokens are generated in response to mouse motion).

Function 1: Show Cursor.

This function increments the Cursor Flag and, if the flag is zero, the cursor display is enabled.

CPU registers are used as follows:

Entry:
AX = 1.
Exit:
All flags and registers preserved.

Function 2: Hide Cursor.

This function decrements the Cursor Flag.

CPU registers are used as follows:

Entry:
AX = 2.
Exit:
All flags and registers preserved.

Function 3: Get Mouse Position and Button Status.

This function returns the state of the Left and Right buttons and the current cursor position.

CPU registers are used as follows:

Entry:
AX = 3.
Exit:
BX = Button status.
CX = X-Cursor Position.
DX = Y-Cursor Position.
All flags and other registers preserved.

The Button Status word returned in BX is a single integer value. Bits 0 and 1 represent the Left and Right buttons, respectively. A bit is set if a button is down and clear if it is up.

Function 4: Set Mouse Cursor.

This function sets the cursor to the specified X-Cursor and Y-Cursor positions. The values must in range of the virtual screen. If the screen is not in high resolution mode, the values are rounded to the nearest values permitted for the current screen mode.

CPU registers are used as follows:

Entry:
AX = 4.
CX = X-Cursor position.
DX = Y-Cursor position.
Exit:
All flags and registers preserved.

Function 5: Get Button Press Information.

This function returns the current button status, a count of button presses since last call to this function, and the X-Cursor and Y-Cursor positions at the last button press.

CPU registers are used as follows:

Entry:
AX = 5
BX = Button Number (0=Left/1=Right).
Exit:
AX = Button status.
BX = Count of Presses since last call (0-32k).
CX = X-Cursor at last press.
DX = Y-Cursor at last press.
All flags and other registers preserved.

Function 6: Get Button Release Information.

This function returns the current button status, a count of button releases since last call to this function, and the X-Cursor and Y-Cursor positions at the last button release.

CPU registers are used as follows:

Entry:
AX = 6.
BX = Button Number (0=Left/1=Right).
Exit:
AX = Button status.

BX = Count of releases since last call (0-32k).
CX = X-Cursor at last release.
DX = Y-Cursor at last release.
All flags and other registers preserved.

Function 7: Set Minimum and Maximum X-Cursor Position.

This function sets the minimum and maximum X-Cursor position. Subsequent cursor motion is restricted to the specified range.

CPU registers are used as follows:

Entry:
AX = 7.
CX = Minimum X-Cursor Position.
DX = Maximum X-Cursor Position.
Exit:
All flags and registers preserved.

If the cursor is outside the area when the call is made, it is set to just inside the area. If Minimum is greater than Maximum, the two values are exchanged.

Function 8: Set Minimum and Maximum Y-Cursor Position.

This function sets the minimum and maximum Y-Cursor position. Subsequent cursor motion is restricted to the specified range.

CPU registers are used as follows:

Entry:
AX = 8.
CX = Minimum Y-Cursor Position.
DX = Maximum Y-Cursor Position.
Exit:
All flags and registers preserved.

If the cursor is outside the area when the call is made, it is set to just inside the area. if Minimum is greater than Maximum, the two values are exchanged.

Function 9: Set Graphics Cursor Block.

This function defines the shape, colour, and center of the cursor for graphics mode.

CPU registers are used as follows:

Entry:
AX = 9.
BX = X-Cursor Hot Spot.
CX = Y-Cursor Hot Spot.
DX = Pointer to Screen and Cursor Masks.
ES = Segment Address of Screen and Cursor Masks.
Exit:
All flags and registers preserved.

The Hot Spot is a point relative to the upper left corner of the cursor block used to determine the cursor coordinates. Both coordinates must be in the range of -16 to +16.

The values in the screen mask and cursor mask are used to build the cursor shape and colour. The ES register contains the segment address of the screen and cursor mask array and DX is the offset to be applied to the ES register.

The screen and cursor masks are two 16- by 16-bit arrays arranged contiguously in memory. The screen mask determines whether the cursor pixel is part of the shape or part of the background. The cursor mask determines how the pixel under the cursor contributes to the colour of the cursor. To create the cursor, the mouse software first logically ANDs the screen mask with the 256 bits of data that define the pixels under the cursor. Then, it logically XORs the cursor mask with the result of the AND operation. The following truth table shows the relationship between the screen mask, the cursor mask, and the resultant screen memory.

The values in the screen mask and cursor mask are used to build the cursor shape and colour. The ES register contains the segment address of the screen and cursor mask array and DX is the offset to be applied to the ES register.

The screen and cursor masks are two 16- by 16-bit arrays arranged contiguously in memory. The screen mask determines whether the cursor pixel is part of the shape or part of the background. The cursor mask determines how the pixel under the cursor contributes to the colour of the cursor. To create the cursor, the mouse software first logically ANDs the screen mask with the 256 bits of data that define the pixels under the

cursor. Then, it logically XORs the cursor mask with the result of the AND operation. The following truth table shows the relationship between the screen mask, the cursor mask, and the resultant screen memory.

The values in the screen mask and cursor mask are used to build the cursor shape and colour. The ES register contains the segment address of the screen and cursor mask array and DX is the offset to be applied to the ES register.

The screen and cursor masks are two 16- by 16-bit arrays arranged contiguously in memory. The screen mask determines whether the cursor pixel is part of the shape or part of the background. The cursor mask determines how the pixel under the cursor contributes to the colour of the cursor. To create the cursor, the mouse software first logically ANDs the screen mask with the 256 bits of data that define the pixels under the cursor. Then, it logically XORs the cursor mask with the result of the AND operation. The following truth table shows the relationship between the screen mask, the cursor mask, and the resultant screen memory.

The values in the screen mask and cursor mask are used to build the cursor shape and colour. The ES register contains the segment address of the screen and cursor mask array and DX is the offset to be applied to the ES register.

The screen and cursor masks are two 16- by 16-bit arrays arranged contiguously in memory. The screen mask determines whether the cursor pixel is part of the shape or part of the background. The cursor mask determines how the pixel under the cursor contributes to the colour of the cursor. To create the cursor, the mouse software first logically ANDs the screen mask with the 256 bits of data that define the pixels under the cursor. Then, it logically XORs the cursor mask with the result of the AND operation. The following truth table shows the relationship between the screen mask, the cursor mask, and the resultant screen memory.

Screen Mask Bit	Cursor Mask Bit	Resultant Screen Bit
0	0	0
0	1	1
1	0	Unchanged
1	1	Inverted

In 640x200 or 640x350 resolution graphics modes each bit in the screen and cursor masks logically maps to one bit on the screen. In (320x200) graphics modes each pair of bits correspond to one pixel.

Function 10: Set Text Cursor.

This function selects the software or hardware text cursor and defines the attributes of the selected cursor.

CPU registers are used as follows:

Entry:
AX = 10.
BX = Cursor Type (0=Software/1=Hardware).
CX = Screen Mask/Scan Line Start.
DX = Cursor Mask/Scan Line Stop.
Exit:
All flags and registers preserved.

If the software text cursor is selected CX & DX contain Screen and Cursor masks. The 16-bit masks are ANDed and XORed in the same manner as the graphics cursor operation and operate upon the character and attributes bytes of the character position of the cursor.

In both 40-column and 80-column text modes the 16-bits of screen data for each character take the following form:

Bits:	15	14..12	11	10..8	7.....0
Contents:	B	Backgnd	I	Foregnd	Character

Where:

B
Blink Bit.
Backgnd
Three bit (RGB) pattern specifying background colour.
I
Intensity bit.
Foregnd
Three bit (RGB) pattern specifying foreground colour.

Refer to section 1.11 for the attribute byte details.

The screen and cursor masks are divided into the same fields as shown above so that the value of the masks defines the new attributes of the character when the cursor is over it. For example a screen mask of 077FFh and a cursor mask of 07700h would invert the foreground and background colours.

If the hardware cursor is selected CX & DX define the first and last scan line in the cursor shown on the screen.

Function 11: Read Mouse Motion Counters.

This function returns the horizontal and vertical mickey count since the last call to this function. CPU registers are used as follows:

Entry:
AX = 11.
Exit:
CX = X-Count.
DX = Y-Count.
All flags and other registers preserved.

The "mickey" is the standard unit mouse motion equal to approximately 1/200 of an inch. See Mouse function 15 which sets the mickey to pixel ratios.

Function 12: Set User Defined Subroutine Input Mask.

This function sets the call mask and subroutine address for the mouse interrupts. CPU registers are used as follows:

Entry:
AX = 12.
CX = Call Event Mask.
DX = Address Offset to Subroutine.
ES = Segment address of Subroutine.

All flags and other registers preserved.

The mouse driver uses the ticker interrupt to poll the mouse hardware (at approx. 55 times per second) and when one of the events specified in the call event mask is noted your subroutine is called. The layout of the call event mask is:

Bit	Event (1=Enabled)
15-5	Unused.
4	Right Button Released. (will never occur).
3	Right Button Pressed. (will never occur).
2	Left Button Released.
1	Left Button Pressed.
0	Cursor Position Changed.

Note that calling mouse function zero (Initialisation) disables all events so that function 12 must be called again.

When calling your subroutine the CPU registers are loaded as follows:

AX = Event bit (as per the above table).
BX = Button State (BL = Left Button / BH = Right Button - will never occur).
CX = X-Cursor.
DX = Y-Cursor.

Function 13: Light Pen Emulation Mode On.

This function enables the light pen emulation by the mouse. CPU registers are used as follows:

Entry:
AX = 13.
Exit:
All flags and registers preserved.

With the Light Pen Emulation on, the VDU I/O software interrupt (Int 16) returns mouse identification vice the normal light pen address information (See VDU Sub-Function 4).

Function 14: Light Pen Emulation Mode Off.

This function disables the light pen emulation by the mouse. CPU registers are used as follows:

Entry:
AX = 14.
Exit:
All flags and registers preserved.

Function 15: Set Mickey/Pixel Ratio.

This function sets the mickey to pixel ratio for mouse motion. CPU registers are used as follows:

Entry:
AX = 15.
CX = X-Mickey/Pixel Ratio.
DX = Y-Mickey/Pixel Ratio.
Exit:
All flags and registers preserved.

The X- and Y- ratios specify a number of mickeys per 8 pixels. The values must be in the range of 1 to 32767.

With a setting of 16 mickeys per 8 pixels horizontally it takes about 6.4 inches of mouse movement to move the cursor across the screen (640 pixels). With the same 16 mickeys per 8 pixels vertically it takes about 2 inches of travel to move the cursor the full vertical deflection (200 pixels).

Function 16: Conditional Off.

This function defines a region on the screen for updating. CPU registers are used as follows:

Entry:
AX = 16.
CX = Upper X-Screen Coordinate.
DX = Upper Y-Screen Coordinate.
SI = Lower X-Screen Coordinate.
DI = Lower Y-Screen Coordinate.
Exit:
All flags and registers preserved.

The mouse cursor is hidden while the screen is being updated and a call to function 1 is needed to show the cursor again.

Function 16 is similar to function 2 (Hide Cursor) bit is for advanced applications which require quicker screen updates.

Function 19: Set Double Speed Threshold.

This function defines the threshold speed for doubling the cursor's motion on the screen. CPU registers are used as follows:

Entry:
AX = 19.
DX = Threshold Speed in Mickeys/Second.

This function makes it easier to point at images widely separate on the screen.

A threshold value of zero sets a value of 64 mickets.second. Setting a large value (such as 32767) disables the double speed threshold.

Appendix 2: MS-DOS System Configuration

The MS-DOS operating system allows for a number of installation specific configuration options during the system startup progress through the use of a file called CONFIG.SYS when it is found in the root directory of the startup disk. These configuration options include the following commands:

Command	Description
BREAK	Extended BREAK checking (Ctrl-C).
BUFFERS	Number of sector buffers.
COUNTRY	Country Specific parameter selection.
DEVICE	Device driver installations.
DRIVPARM	Override the drive parameters for a logical drive.
FCBS	Number of files open by file control blocks.
FILES	Maximum number of files open concurrently.
LASTDRIVE	Maximum drive letter allowable.
SHELL	Top level command processor specification.
STACKS	Override the default DOS stack resources.

The CONFIG.SYS can be created with any text editor and the simple screen editor RPED is ideal for this purpose.

2.1 BREAK Command

This command enables the MS-DOS extended break checking to be either set or reset. Normally, MS-DOS checks to see if CTRL-C has been typed while it is reading from the keyboard, writing to the screen or a printer. Setting Break to 'on' allows CTRL-C checking to be extended to other functions such as disk reads or writes. The syntax of the BREAK command is:

BREAK=[ON]
or
BREAK=[OFF]

If no field is specified then OFF is assumed (as the default value).

2.2 BUFFERS Command

This command allows you to specify the number of buffers that MS-DOS allocates when it starts up. A disk buffer is a block of memory where MS-DOS holds data being read from or written to a disk when the amount of data is not an exact multiple of sector size.

The syntax of the BUFFERS command is:

BUFFERS=n

Where 'n' is a number between 2 and 255. If the BUFFERS command is not used then MS-DOS defaults to 2 buffers. The number of buffers remains in effect after bootstrap until the machine is switched off or bootstrapped again. For best performance for standard applications environments (word processors, spreadsheets, etc.) a buffers allocation between 10 and 20 is recommended. If you tend to use many subdirectories then an allocation upwards to 30 may be better. But since buffers use the system available memory, there may have to be a compromise between memory usage and performance. Buffers allocated beyond 40 serves no useful purpose. Refer to the Users's Manuals for you applications if in douby about required buffers for particular applications programs.

2.3 Country Command

The country command is used to select the country dependent information as shown in appendix 3.

The syntax of the country command is:

COUNTRY=nnn

Where 'nnn' is the 3-digit country (Num) code from Appendix 3. Note that only the information in table is effected and other country dependent factors such as the language links, N-Utility setup, KEYBxx, and other national variant disks effect the total country dependent environment.

2.4 DEVICE Command

This command installs the device driver in the specified pathname to the system list.

The syntax of the DEVICE command is:

DEVICE=[drive:]<pathname>

The file specified is loaded and given control. The driver may then perform the necessary steps to configure itself and the system for its operation. See the MS-DOS Technical Reference Manual for information on how to create your own device driver.

Your MS-DOS disk (Disk 1) contains two installable device drivers, DRIVER.SYS, and RAMDRIVE.SYS which can be used for variable device configurations.

If you plan to use the ANSI escape sequences described in the PC1640 Users Manual, you would need to include the following command in your CONFIG.SYS file:

DEVICE=ANSI.SYS

This command causes MS-DOS to replace all keyboard input and screen output support with the ANSI escape sequences. Refer to the PC1640 User Instructions (Book 1) Appendix IV for ANSI escape sequence reference information.

2.4.1 DRIVER.SYS>

DRIVER.SYS is an installable device driver that supports external drives. To install DRIVER.SYS, include the following command in your CONFIG.SYS file:

DEVICE=DRIVER.SYS /D:dd [/C] [/F:ff] [/H:hh] [/N] [/S:ss] [/T:tt]

Where:

/D:dd is drive number (0-127: Floppy drives, 128-255 Hard drives)

and optionally:

/C indicates changeline (doorlock) support required.
/F:ff indicates the form factor where:
0 = 5.25 inch floppy diskette, 320/360 K bytes.
1 = 5.25 inch floppy diskette, 1.2 M bytes.
2 = 3.5 inch floppy diskette, 720 K bytes.
3 = 8 inch floppy diskette, Single Density.
4 = 8 inch floppy diskette, Double Density.
5 = Hard Disk.
6 = Tape Drive.
7 = 3.5 inch floppy diskette, 1.44 M bytes.
/H:hh is the maximum head number (1-99).
/N indicates non-removable block device.
/S:ss is the number of sectors per track (1-99).
/T:tt is the number of tracks per side (1-999).

2.4.2 RAMDRIVE.SYS

RAMDRIVE.SYS is an installable device driver which enables the usage of a portion of the computer's memory as though it were a disk drive. This area of memory is referred to as a RAM disk or a virtual disk.

If you have extended memory installed starting at the 1MB boundary or if you have an extended memory which meets the LIM [Lotus(R)/Intel(R) /Microsoft(R)] Expanded Memory Specification, you can use this memory for one or more RAM disks. Otherwise RAMDRIVE.SYS locates RAM drives in low memory.

To install RAMDRIVE.SYS, include the following command in your CONFIG.SYS file:

DEVICE=RAMDRIVE.SYS [bbbb] [ssss] [dddd] [/A]

Where:

bbbb is disk size in kilobytes. Default is 64 and minimum is 16, (or NVR to use NVR value).
ssss is the sector size. The values 128, 256, 512, and 1024 are allowed. Default is 128.
dddd is the number of root directory entries. The default is 64, the minimum value is 4 and the maximum is 1024.
/A indicates that an extended memory board which meets the LIM Expanded Memory Specification for a RAM drive is in use. If this switch is used, the /E switch cannot be used.

There is an additional parameter for this driver which applies to 80286 style CPU architecture with memory above the 1M byte range. This parameter is as follows:

/E indicates that extended memory (above 1MB) is to be used. If this switch is used, the /A switch cannot be used.

2.5 DRIVPARM Command

The DRIVPARM command allows overriding of the device parameters for a specific logical drive.

The syntax is:

DRIVPARM= /D:dd [/F:ff /T:tt /S:ss /N /C /H:hh]

Where:

/D:dd is drive number (0-255). (0=A, 1=B, 2=C ...)

and optionally:

/T:tt is the number of tracks per side (1-999).
/S:ss is the number of sectors per track (1-99).
/H:hh is the maximum head number (1-99).
/C indicates changeline (doorlock) support required.
/N indicates non-removable block device.
/F:ff indicates the form factor where:
0 = 5.25 inch floppy diskette, 320/360 K bytes.
1 = 5.25 inch floppy diskette, 1.2 M bytes.
2 = 3.5 inch floppy diskette, 720 K bytes.
3 = 8 inch floppy diskette, Single Density.
4 = 8 inch floppy diskette, Double Density.
5 = Hard Disk.

6 = Tape Drive.
7 = 3.5 inch floppy diskette, 1.44 M bytes.

This command allows the overriding of default system parameters for a particular logical drive. This information would be used by the commands which create new diskettes (such as FORMAT and COPY) when writing out the directory and FAT (File Allocation Table) information. For any physical device which is read the information in the FAT ID is used when determining device characteristics for floppy disks, hard disks and tape drives.

If no form factor (/F:) is specified then a value of 2 is assumed (720K, 3.5 inch diskette).

2.6 FCBS Command

The FCBS command allows you to specify the number of file control blocks available to the system and consequently the number of files which can be opened at any one time.

The syntax of the FCBs is:

FCBS=<x>,<y>

Where <x> is the number of FCBs (in the range of 1 to 255) to allocate and <y> is the number of FCBs protected from closure when a program tries to open more than <x> files. The first <y> files opened will be protected. MS-DOS selects the least recently used (non-protected) FCB when it must automatically close a file.

If the FCBS command is not used MS-DOS defaults <x> and <y> to 4 and 0 respectively. It is an error to set <y> greater than <x>

2.7 FILES Command

The FILES command specifies the maximum number of file handles that can concurrently be opened. When a program opens a file or a device it is assigned an identifier or "handle" which can be used by that program in referring to the file.

The syntax of the FILES command is:

FILES=n

Where 'n' is the number of handles in the range of 8 to 255. When no FILES command is used MS-DOS assumes a default value of 8. Any value higher than 20 serves no useful function.

2.8 LASTDRIVE Command

The LASTDRIVE command is used to set the maximum drive letter which MS-DOS will accept.

The syntax of the LASTDRIVE command is:

LASTDRIVE=d

Where 'd' is any letter from A to Z (and is case insensitive). When the drive letter is lower than the actual physical drives then MS-DOS ignores the LASTDRIVE specification and uses the default value which is the letter 'E'.

2.9 SHELL Command

The SHELL command is used to specify an alternate top-level command processor in place of the standard COMMAND.COM file.

The syntax of the SHELL command is:

SHELL=[drive:]pathname [param1 [parm2 ..[paramn]]]

This command is used in conjunction with major software packages which furnish their own command processors. The MS-DOS technical manual contains information on developing command processors.

2.10 STACKS Command

The Stacks command allows you to override the default DOS stack resource parameters. For each hardware interrupt which occurs, MS-DOS allocates a stack to it from the pool of available stacks. When the interrupt process is completed, MS-DOS returns the stack to the available stack pool.

The syntax of the STACKS command is:

STACKS=<number of stacks>,<stack size in bytes>

If there is no STACKS= command in your CONFIG.SYS file then MS-DOS allocates default stack resources equivalent to the command STACKS=9,128. This however may not be sufficient if you are using multiple interrupting devices (such as LANs, 8087 NDPs, or Hard Disks) and under these circumstances you may experience a number of stack related messages such as "Internal Stack Failure" (most

predominantly) or even "Divide Overflow". When messages such as this occur it is advisable to try increasing the stacks, bearing in mind that stacks do use up available system memory in the same way that buffers and FCBs do. The number of stacks allowable is from 8 to 64 and the stack size parameter may vary from 32 to 512 bytes.

Appendix 3: Country Dependent Information for MS-DOS 3.2

Country	Num	DtF	DtS	TmS	TmF	CSm	CFt	CSd	ThS	DeS	DlS
Australia	061	1	-	:	1	$	0	2	,	.	,
Belgium	032	1	/	:	1	F	3	2		,	;
Canada	002	2	-	:	1	$	3	2		,	;
Denmark	045	1	/	.	1	DKR	3	2	.	,	;
Finland	358	1	-	:	1	MK	3	2		,	;
France	033	1	/	:	1	F	3	2		,	;
Germany	049	1	.	.	1	DM	0	2	.	,	;
Italy	039	1	/	:	1	Lit	1	0	.	,	;
Israel	972	1	/	:	1	ö	2	2	,	.	,
Middle East	785	1	/	:	1	$	3	3	.	,	;
Netherlands	031	1	-	:	1	*ƒ*	0	2	.	,	;
Norway	047	1	/	.	1	KR	2	2	.	,	;
Portugal	351	1	/	:	1	$	4	2	.	,	;
Spain	034	1	/	:	1	Pt	3	2	.	,	;
Sweden	046	2	-	.	1	SEK	2	2	.	,	;
Switzerland	041	1	.	.	1	Fr	2	2	,	.	,
United Kingdom	044	1	-	:	1	£	0	2	,	.	,
United States	001	0	-	:	0	$	0	2	,	.	,

Table Columns:

Num = Country Number Code.
DtF = Date Format. (0 = U.S. M/D/Y, 1=EURO D/M/Y, 2 = JAPAN Y/M/D)
DtS = Date Separator.
TmS = Time Separator.
TmF = Time Format. (0=12-hour clock, 1=24-hour clock)
CSm = Currency Symbol.
CFt = Currency Format. (Bit 0: 0 = Currency symbol Precedes/1=Follows Field, Bits 1 & 2: Number of spaces between Value & Symbol)
CSd = Number of significant decimal digits in currency.
ThS = Thousands Separator.
DeS = Decimal Separator.
DlS = Data List Separator.

Appendix 4: RS232C Connections

For a complete understanding of the connections required between the RS232C and the outside world, it is important to realize that all devices with a serial interface can be classified as either a modem or as a terminal. Modems are merely a way of extending the length of the connection (often via a terminal wire) between two terminals. Fig 1 (below) shows a simplified, idealised terminal to terminal connection through modems.

IDEALISED TERMINAL TO TERMINAL CONNECTION

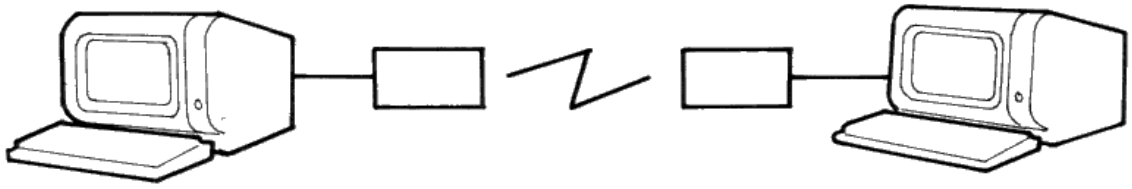


Fig 1

The standard connector used for serial interfaces has 25 pins although only up to seven are required in most cases. When connecting to a modem a 'one-to-one' cable is used, i.e. pin 1 to pin 1, pin 2 to pin 2, ... pin 25 to pin 25. Assuming such cables are in use, data is transferred

as follows:

Following the signal path from left to right (in Fig 2), characters from the keyboard are sent as serial data patterns out of pin 2 of the left-hand terminal, to pin 2 of the modem (the connection marked 'transmit data'). The left-hand modem sends the characters via the telephone line, to the right hand modem. The characters are received pin 3 of the right hand modem (the connection marked 'receive data') which sends them to pin 3 of the right-hand terminal. On receipt of the characters, the right-hand terminal displays them on the screen.

Notice how the names of the connections 'transmit data' and 'receive data' are expressed from the view point of the terminals and not the modems.

The data path from left to right just described, is exactly matched by a data path from right to left which uses the same numbered connections, i.e. pin 2 from the right-hand terminal to its modem (transmitting), and then to pin 3 of the left-hand (receiving) modem to the terminal. This arrangement is perfectly symmetrical, and there is no confusion over who is using which pin number and for what direction of data transfer.

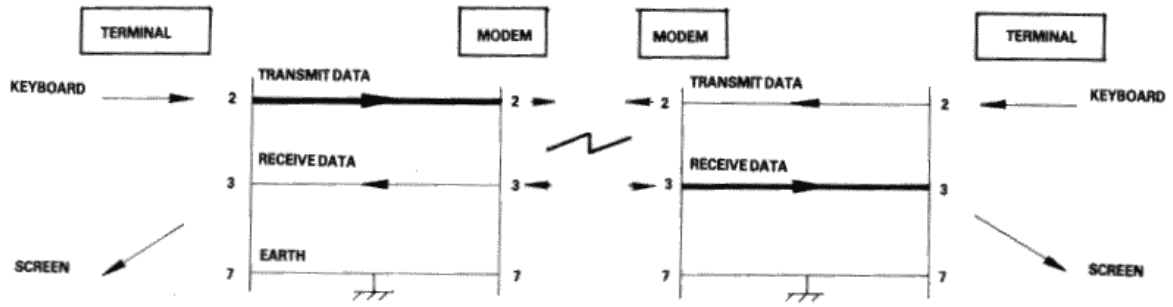


Fig 2

Problems of definition arise, howeve, when we wish to connect two terminals together locally, without the intervening pair of modems. We cannot connect pin 2 to pin 2 because both keyboard will be transmitting head-on and neither screen is connected to anyone who is sending. The obvious solution is to cross over pins 2 and 3 so that the transmit pin of each terminal is connected to the receive pin of the other. A cable containing such a cross-over connection is known as a 'Null-modem' cable because of the way in which it replaces the pair of back to back modems.

The earth pin (pin 7) is still common to both terminals using this arrangement.

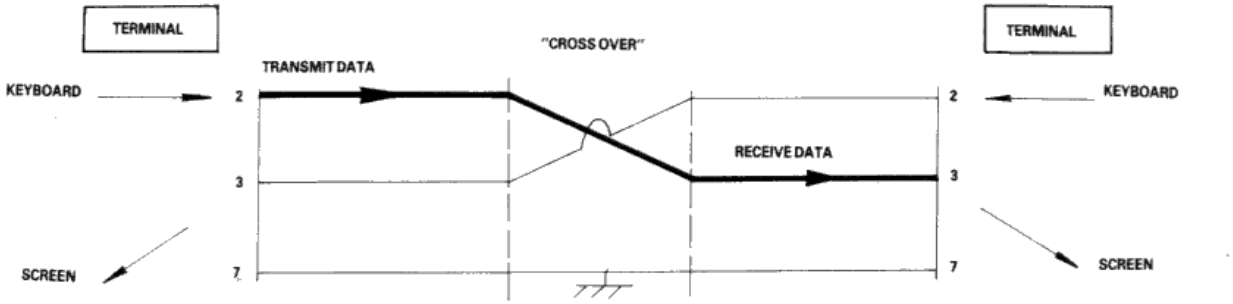


Fig 3

Naturally, the Amstrad PC1640 with its RS232C interface is considered a terminal, and therefore to connect to a modem (for example, to dial-up a database) requires a simple one-to-one cable.

The Null-modem cable is required for connecting to other terminals. The sort of equipment we mean by terminals is: a second Amstrad computer plus RS232C, a conventional Visual Display Unit (VDU), a printer with a serial interface, or any other serial interface device.

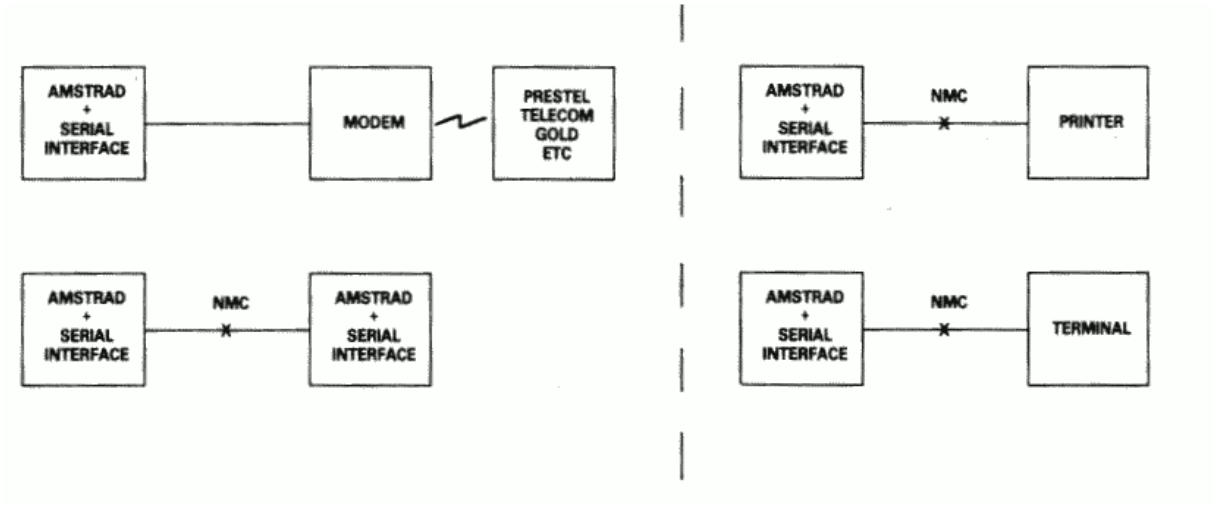


Fig 4

There is a point to be noted here: many manufacturers of devices such as desk-top computers wire up their serial interface (for VDU or a Printer) as if it were a modem, not a terminal. This is in the belief that life will therefore be simpler because VDU's and printers can be connected to that computer with one-to-one cables.

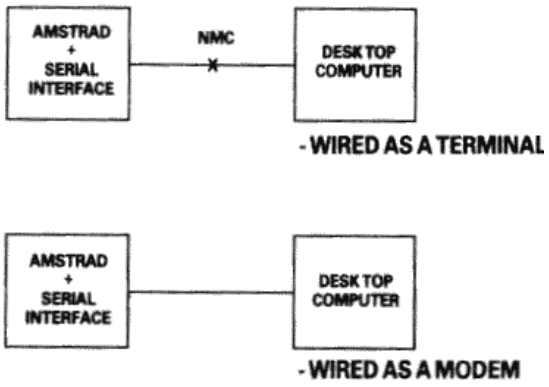


Fig 5

In a perfect world, it would be possible to identify which serial devices behave like modems and which ones behave like terminals by examining the 'sex' of the 25-way connector - terminals should have a 'male' connector, and modems a 'female' connector. This is not, unfortunately, as reliable a guide as it should be, as many manufacturers of terminals and printers equip them with 'female' connectors, mostly for reasons of electrical safety.

If in doubt, the ultimate test is to examine the user manual and determine the function of PIN 2 - if the description includes the word 'TRANSMIT' then the equipment is wired as a terminal, and if it includes the word 'RECEIVE' then the equipment is wired as a modem.

Hardware Flow Control

The simplified connection described so far does not allow any control of the data flow. In practice, we often with the receiving device to have control over the transmitting device, thus preventing the receiving device from being overwhelmed (where it is slower in using the input than the rate at which the input is arriving). In addition, if the transmitting device has reason to mistrust the data which it is sending, there should be some provision for it to disable the receiving device.

In the case of modem to terminal connection; when the terminal is able to transmit it activates pin 4 - the RTS pin (Request To Send). When the modem is ready to receive input it activates pin 5 - the CTS pin (Clear To Send). The terminal will only send when CTS is activated. Thus the modem can control the flow rate using CTS.

When the modem considers that the data which it is about to send is suitable, it activates pin 8 - the DCD pin (Data Carrier Detect). When the terminal is ready to receive input it activates pin 20 - the DTR pin (Data Terminal Ready). The modem will only transmit when DTR is activated. Thus the terminal can control flow rate using DTR.

There are two further signals which must be introduced here. One is on pin 22 - the Ring indicator, which simply allows the modem to tell the terminal that the phone is ringing (at which point software in the terminal might be expected to wake up). The other signal is on pin 6 - DSR (Data Set Ready). This signal is ignored by the receiving side of the RS232C; the modem will activate this signal at much the same time that it activates DCD, and therefore no functionality is lost by ignoring DSR.

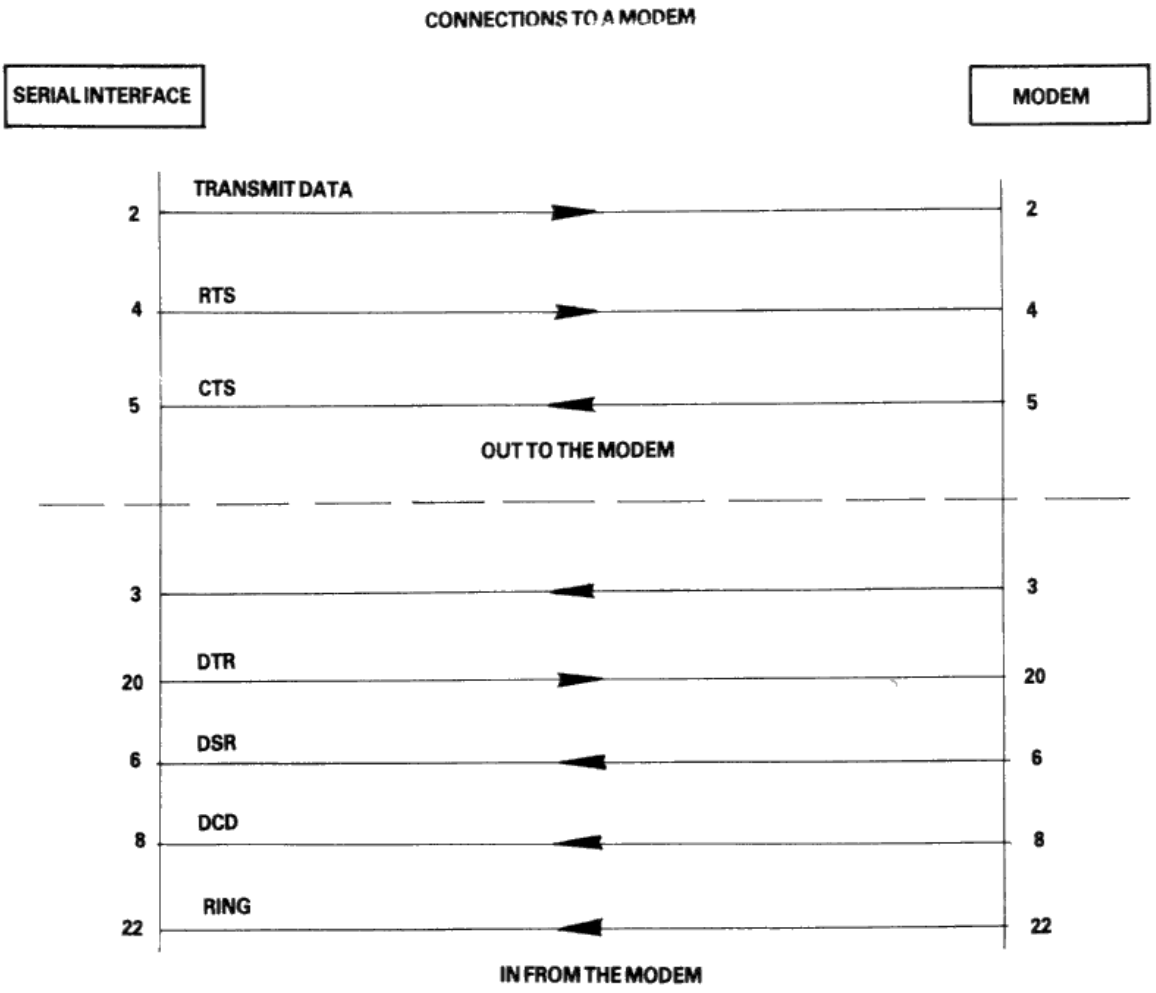


Fig 6

In the case of terminal-to-terminal conenctions, the Null-modem cable must be used with the additional connections to pins 2, 3, and 7 as already discussed. The full Null-modem cable swaps pins 4 and 8 - the RTS/DCD 'I am happy to send' signals, and pins 20 and 5 - the DTR/CTS 'busy' signals. To be on the safe side, pin 6 (DSR) is connected to pin 8 (DCD) in case that end of the cable is ever connected to a terminal which is fussy and requires DSR as well as DCD.

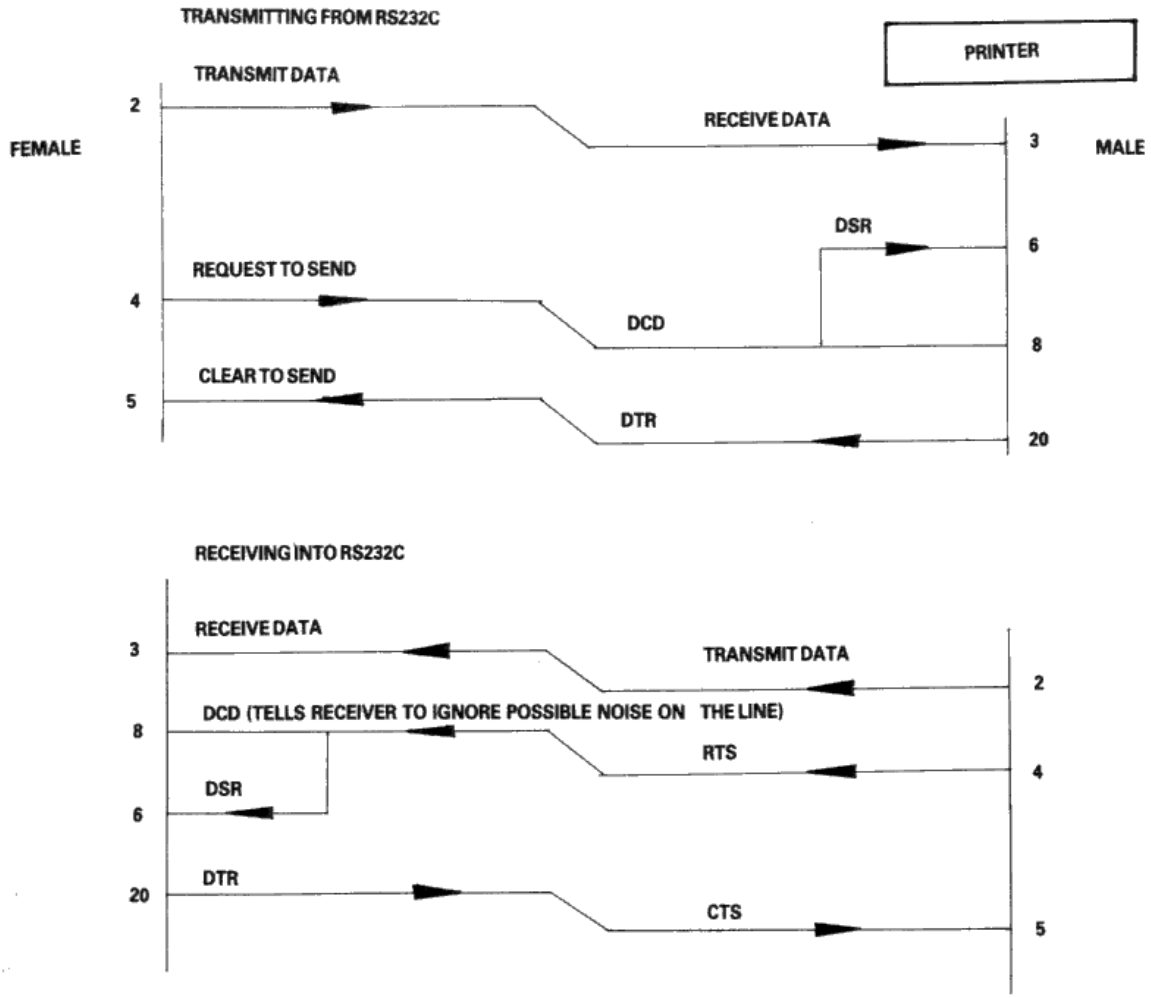


Fig 7

There is a school of thought which says that a Null-modem cable, unlike the pair of modems it replaces, is ALWAYS 'happy to send'. Therefore it is quite in order to generate DCD (and DSR) permanently. This is achieved by connecting them to the RTS at the same end of the cable, rather than to the RTS at the other end of the cable.

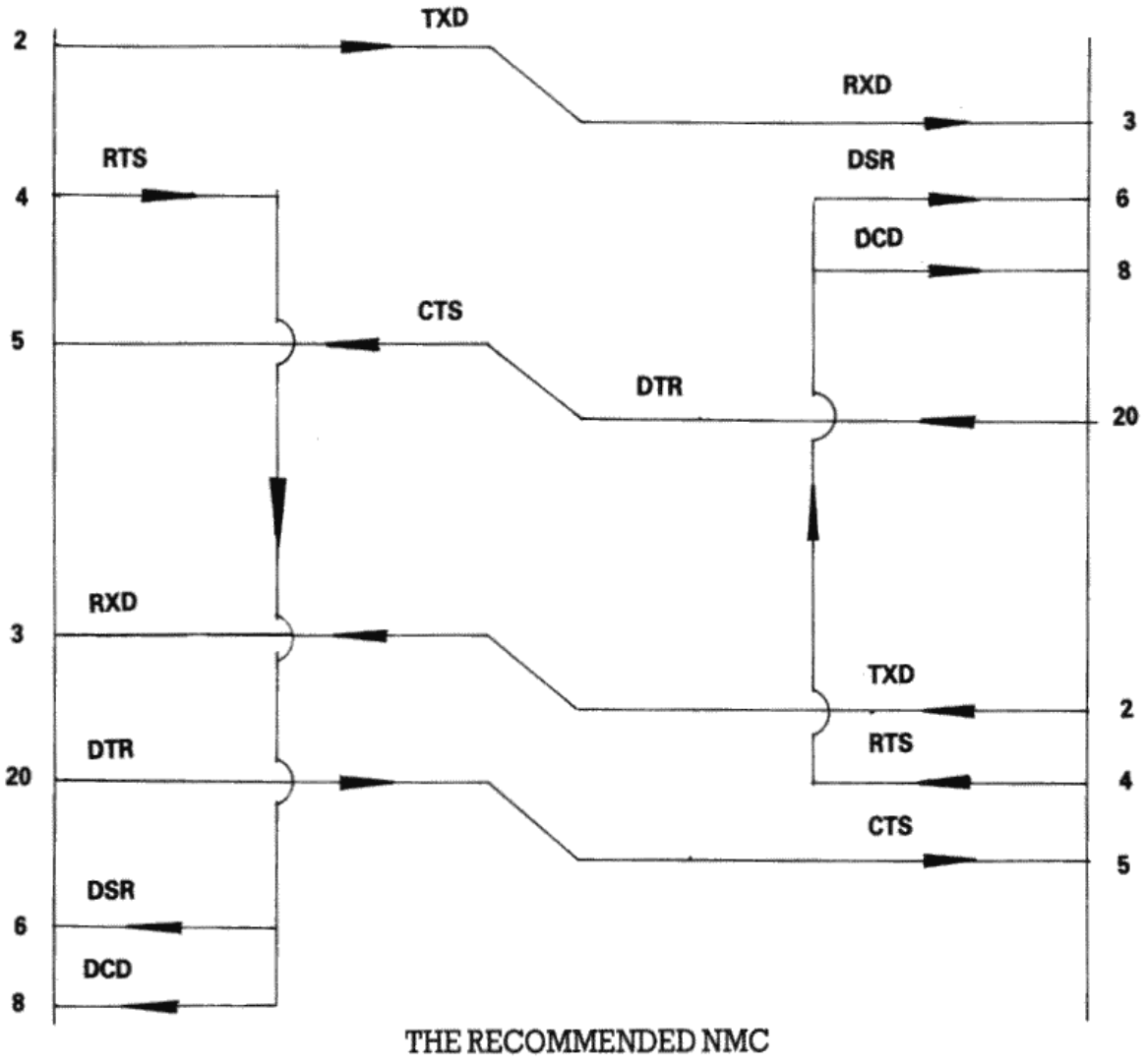


Fig 8

Finally, if the transmission rate from one of the two terminals is known to be unstoppable (e.g. a person typing at the keyboard), or is so slow and infrequent (e.g. the software handshake characters 'XON, XOFF' sent by the printer) that there is no danger of over-running the receiving end, then it is permissible to permanently enable the transmission by linking pin 5 (CTS) to pin 4 (RTS), i.e. to always send if ready (at the transmitting end of the cable). It may well be facilitated in any case, for the transmitting terminals to ignore the state of CTS under these circumstances.

THE RECOMMEND TERMINAL CABLE

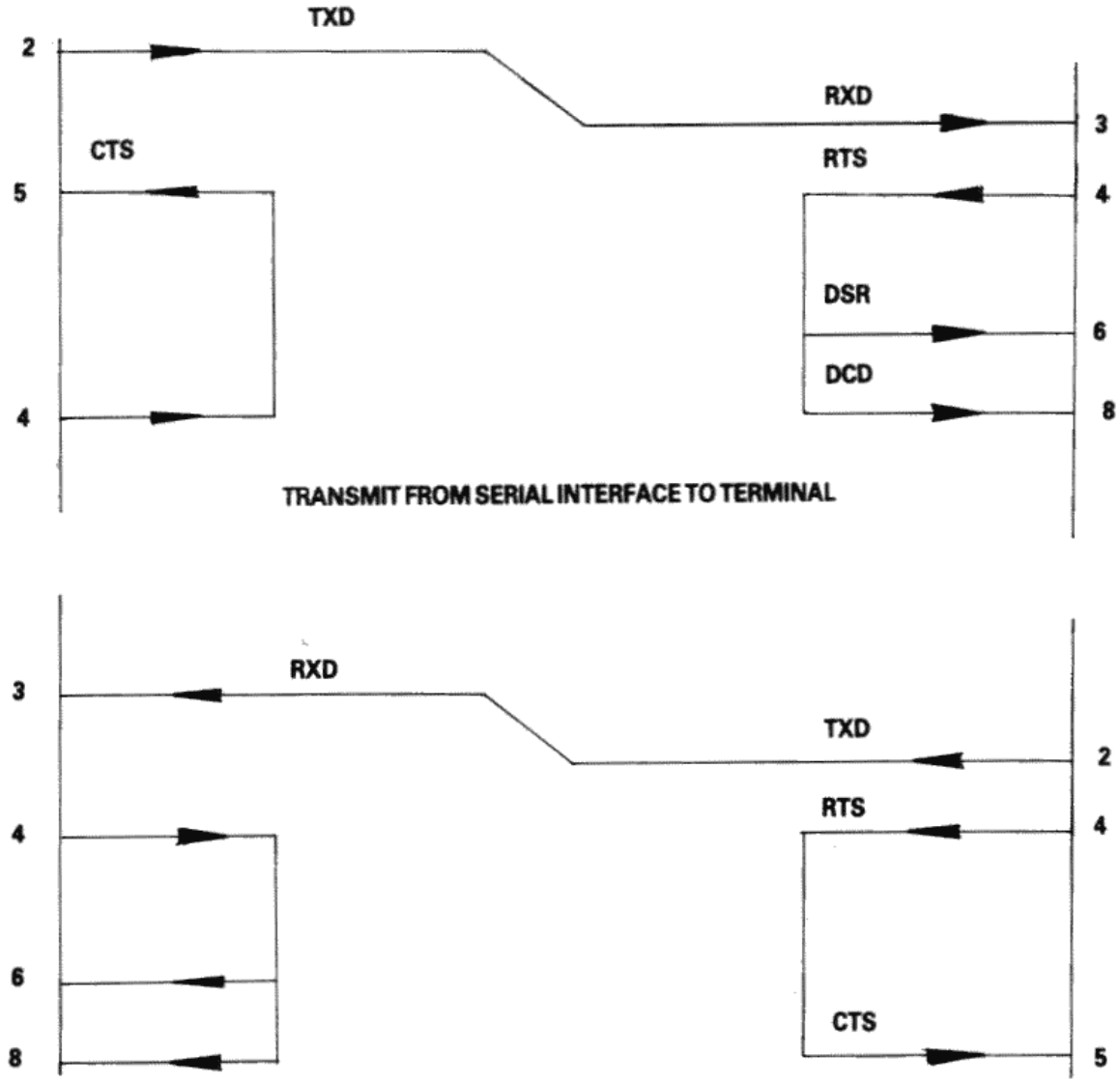


Fig 9

Appendix 5: Printer Lead (PL-2) Wiring Specification

Connectors.

1. Computer Centronics Parallel Interface connector is a 25-way, D-plug.
2. Printer Input connector is a 36-way, IEEE-488 plug.

Cable Wiring.

Line Name	Computer Connector	Printer Connector
-Strobe	1	1
Data Bit 0	2	2
Data Bit 1	3	3
Data Bit 2	4	4
Data Bit 3	5	5
Data Bit 4	6	6
Data Bit 5	7	7
Data Bit 6	8	8
Data Bit 7	9	9
-Ack	10	10
Busy	11	11
PO	12	12

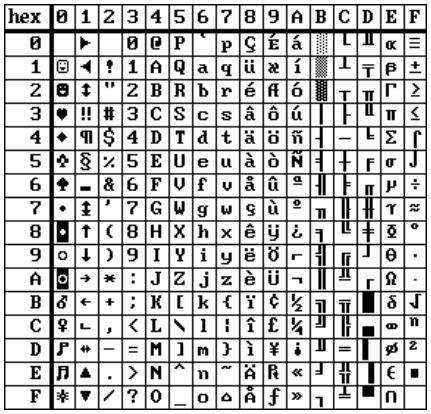
Line Name	Computer Connector	Printer Connector
Select Out	13	13
-AutoFd	14	14
-Error	15	32
-Reset	16	31
-Select In	17	36
GND	18	19
GND	19	20
GND	20	21
GND	21	22
GND	22	23
GND	23	24
GND	24	25
GND	25	26
GND	-	27
GND	-	28
GND	-	29
GND	-	30
GND	-	33
GND	-	15
GND	-	16
GND	-	17
GND	-	18
GND	-	34
GND	-	35

Appendix 6: PC1640 Power Usage

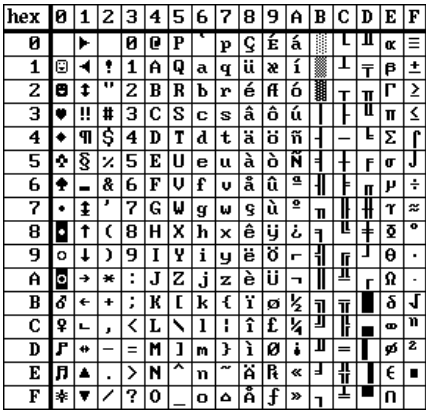
The following is a summary of the power supply requirements for the Amstrad PC1640.

Major Component	+5 V	+ 12V	-5 V	- 12V
Main Board and keyboard	2.10a	0.15a	0.00a	0.10a
Floppy Disk Drive	0.40a	1.00a	0.00a	0.00a
Hard Disk	1.10a	0.80a	0.00a	0.00a
Expansion Slots	2.10a	0.15a	0.10a	0.15a
Total	5.70a	2.10a	0.10a	0.25a

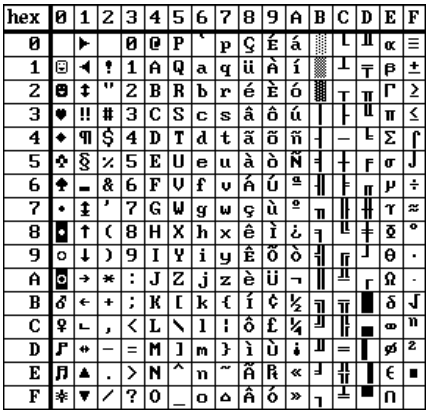
Appendix 7: ROM Character Set



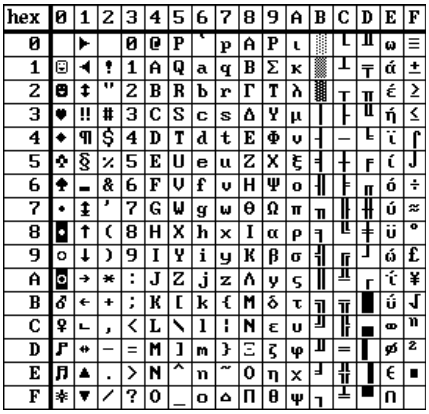
ENGLISH character set



DANISH character set

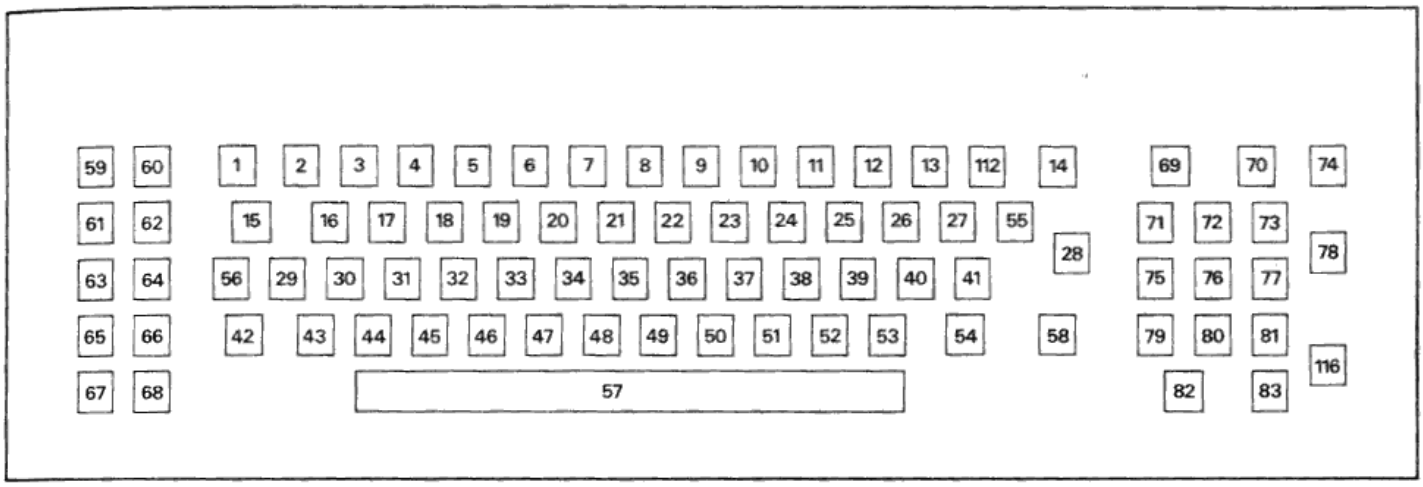


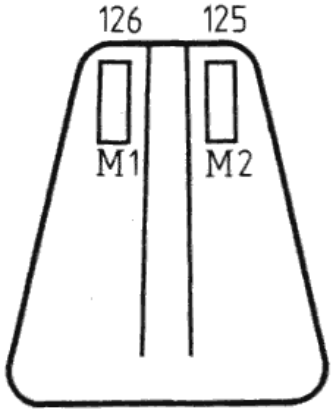
PORTUGESE character set



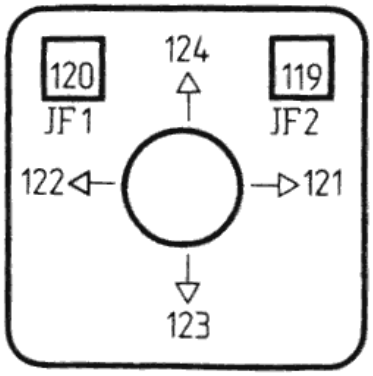
GREEK character set

Appendix 8: Keyboard Keycodes





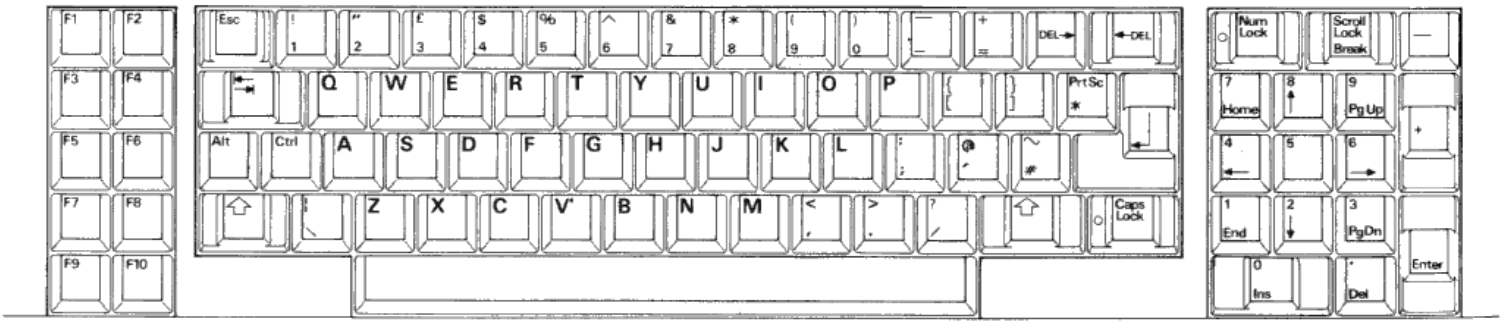
Mouse



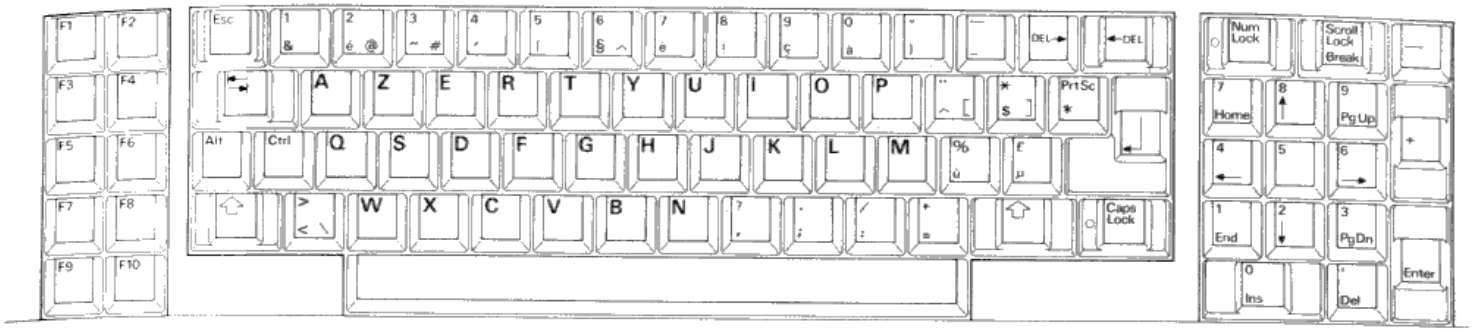
Joystick

Appendix 9: Keyboard Layouts

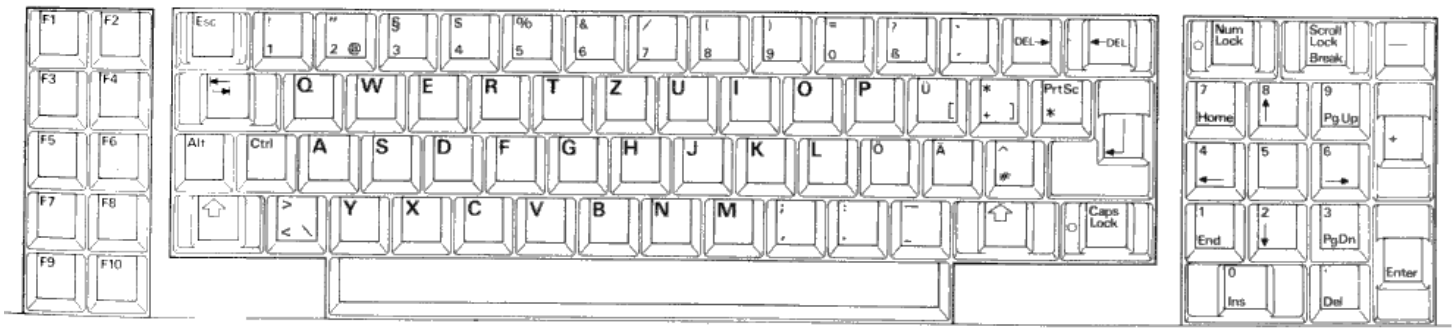
USA keyboard



UK keyboard

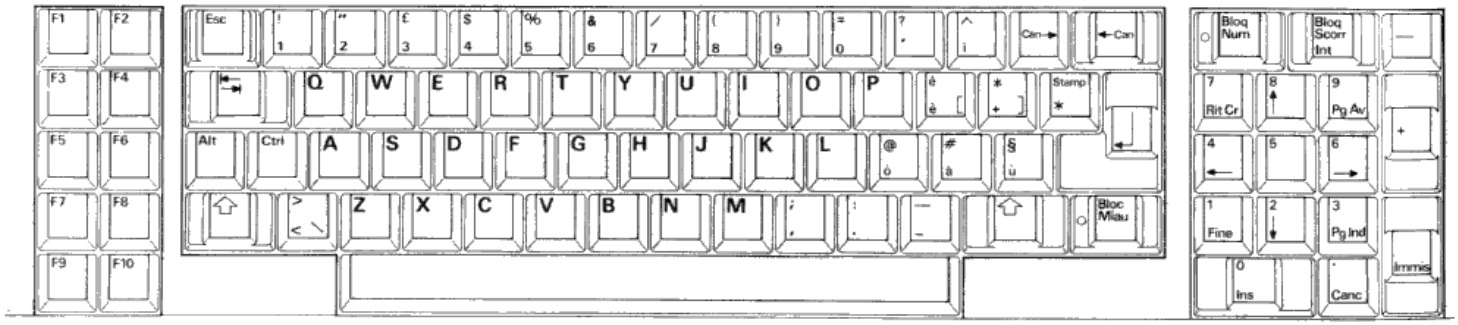


FRENCH keyboard

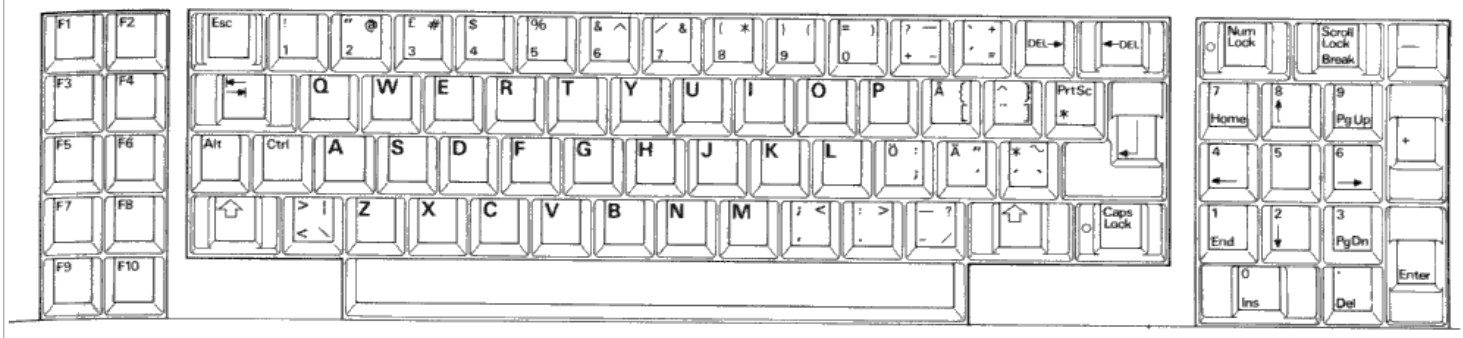


GERMAN keyboard

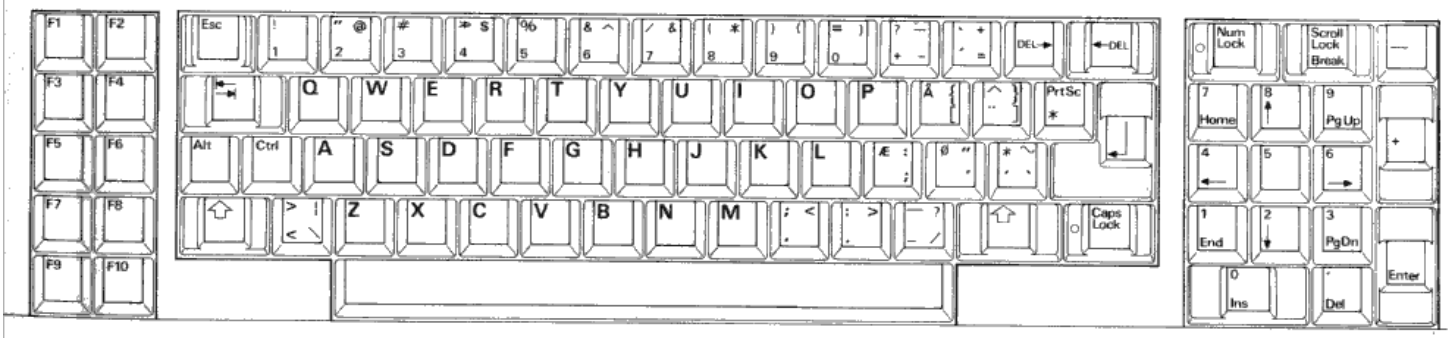
SPANISH keyboard



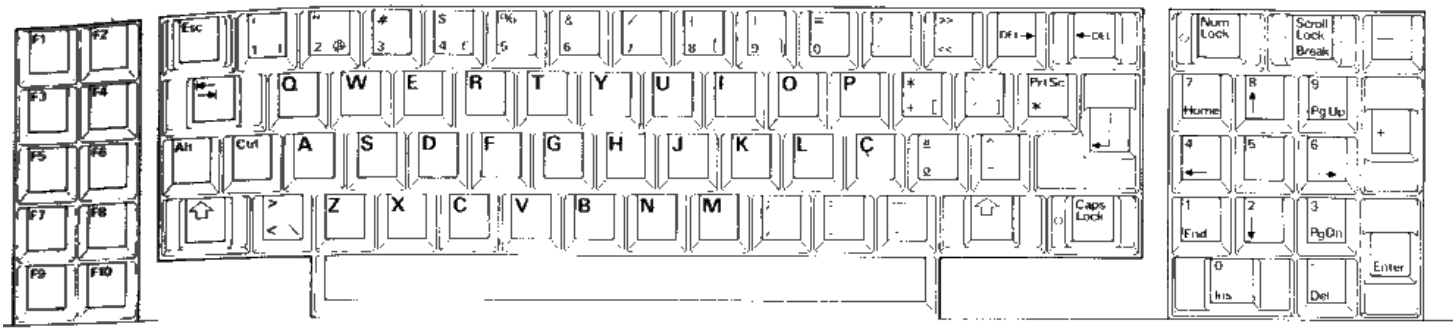
ITALIAN keyboard



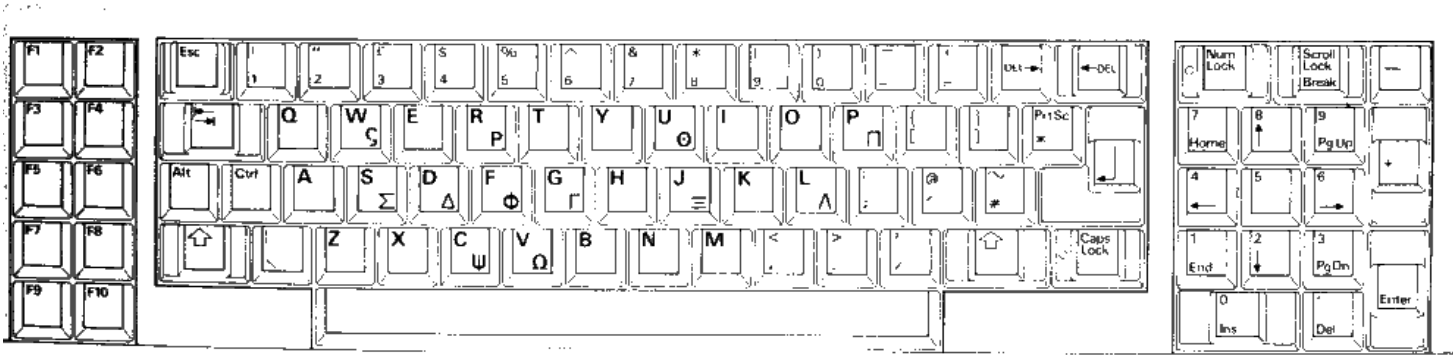
SWEDISH keyboard



DANISH keyboard



PORTUGESE keyboard



GREEK keyboard

NORWEGIAN keyboard

Appendix 10: THE LINKER PROGRAM (MS-LINK)

10.1 INTRODUCTION

In this appendix you will learn about MS-LINK. It is recommended that you read the entire appendix before you use MS-LINK.

The MS-DOS linker (called MS-LINK) is a program that:

1. Combines separately produced object modules into one relocatable load module - an executable (.EXE) program file which you can run.
2. Searches library files for definitions of unresolved external references.
3. Resolves external cross-references.
4. Produces a listing that shows both the resolution of external references and error messages.

10.2 Overview of MS-LINK

When you write a program, you write it in source code. This source code is passed through a compiler which produces object modules. The object modules must be passed through the link process to produce machine language that the computer can understand directly. This machine language is in the form required for running programs.

You may wish to link (combine) several programs and run them together. Each of your programs may refer to a symbol that is defined in another object module. This reference is called an external reference.

MS-LINK combines several object modules into one relocatable load module, or Run file (called an .EXE or Executable file). As it combines modules, MS-LINK can search several library files for definitions of any external references that are not defined in the object modules.

MS-LINK also produces a List file that shows external references resolved, and it also displays any error messages.

MS-LINK uses available memory as much as possible. When available memory is exhausted, MS-LINK creates a temporary disk file named VM.TMP.

Figure 1 illustrates the various parts of the MS-LINK operation:

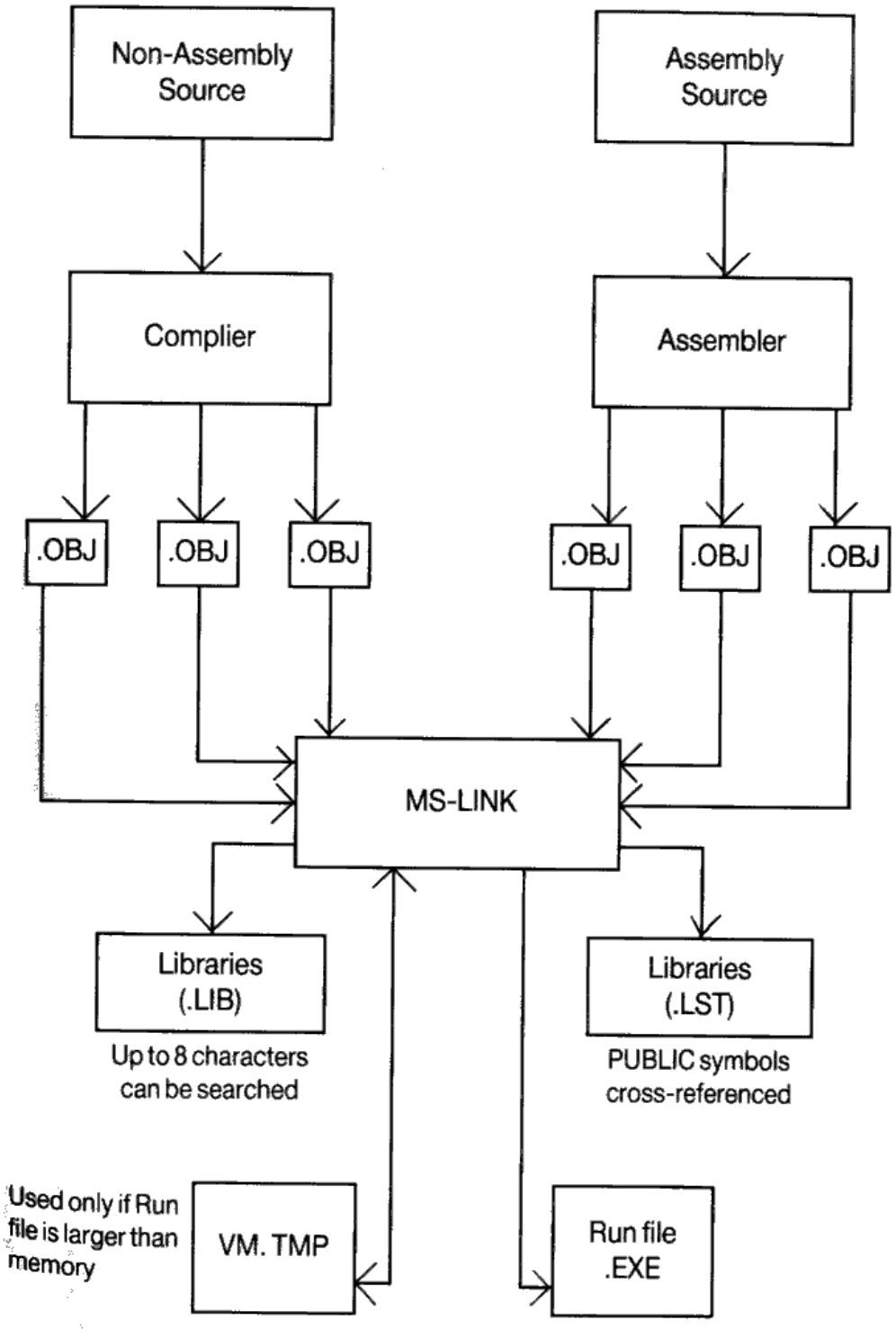
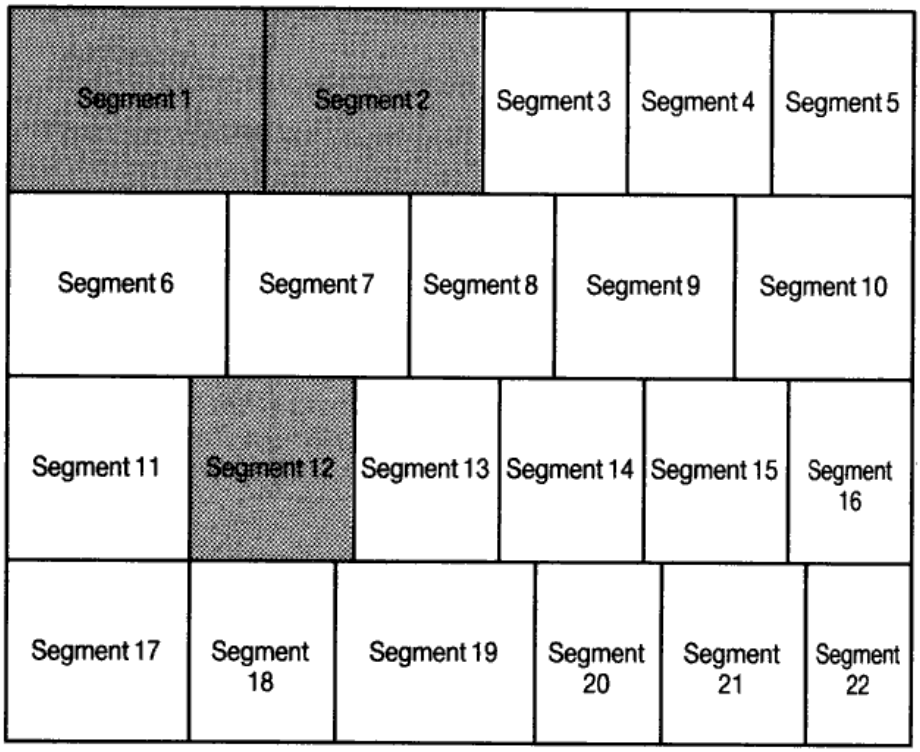


Figure 1. The MS-LINK Operation

10.3 Definitions you will need to know

Some of the terms used in this appendix are explained below to help you understand how MS-LINK works. Generally, if you are linking object modules compiled from BASIC, Pascal, or a high-level language, you will not need to know these terms. However, if you are writing and compiling programs in assembly language you will need to understand MS-LINK and the definitions of the memory divisions of MS DOS.

In MS DOS, memory can be divided into segments, classes and groups. Figure 2 illustrates these concepts.



Shaded area = a group (64K bytes addressable)

Figure 3. How Memory Is Divided

Example:

Segment Number	Name	Class
1	PROG.1	CODE
2	PROG.2	CODE
12	PROG.3	DATA

Note that segments 1, 2 and 12 have different segment names, and may or may not have the same segment class name. Segments 1, 2 and 12 form a group with a group address of the lowest address of segment 1 (ie., the lowest address in memory).

Each segment has a segment name and a class name from the first segment encountered to the last. All segments assigned to the same class are loaded into memory contiguously.

During processing, MS-LINK references segments by their addresses in memory (where they are located). MS-LINK does this by finding groups of segments.

A group is a collection of segments that fit within a 64K byte area of memory. The segments do not need to be contiguous to form a group (see illustration). The address of any group is the lower address of the segments within that group. At link time, MS-LINK analyses the groups, then references the segments by the address in memory fo that group. A program may consist of one or more groups.

If you are writing in assembly language, you may assign the group and class names in your program. In high-level languages (BASIC, COBOL, FORTRAN, Pascal), the naming is done automatically by the compiler.

10.4 Files that MS-LINK uses

MS-LINK:

1. Works with one or more input files.
2. Produces two output files.
3. May create a temporary disk file.
4. May be directed to search up to eight library files.

For each type of file, the user may give a three-part file specification. The format for MS-LINK file specifications is the same as that of a disk file:

[d:]<filename>[<.ext>)

where:

d:
is the drive designation. Permissible drive designations for MS-LINK are A: through O:. The colon is always required as part of the drive designation.

filename
is any legal filename of one to eight characters.

10.4.1 Input file Extensions

If no filename extensions are given in the input (object) file specifications, MS-LINK will recognize the following extensions by default:

.OBJ
-- Object code file from assembly or compilation.

.LIB
-- Library file created for some product such as FORTRAN or C.

10.4.2 Output file Extensions

MS-Link appends the following default extensions to the output (Run and List) files:

.EXE
-- Run file. (may not be overriden)

.MAP
-- List file. (may be overridden)

10.4.3 VM.TMP (Temporary) File

MS-Link uses available memory for the link session. If the files to be linked create an output file that exceeds available memory, MS-LINK will create a temporary file, name it VM.TMP, and put it in the disk in the default drive. If MS-LINK creates VM.TMP, it will display the message:

VM.TMP has been created.
Do not change diskette in drive, <d:>

Once this message has been displayed, you must not remove the disk from the default drive until the link session ends. if the disk is removed, the operation of MS-LINK will be unpredictable, and MS-LINK might display the error message:

Unexpected end of file on VM.TMP

The contents of VM.TMP are written to the Run file. (The name of the file is written at the Run file prompt). VM.TMP is a working file only and is deleted at the end of the linking session.

WARNING

Do not use VM.TMP as a filename for any file. If you have a file named VM.TMP on the default drive and MS-LINK requires the VM.TMP file, MS-LINK will delete the VM.TMP already on disk and create a new VM.TMP. Thus the contents of the previous VM.TMP file will be lost.

10.5 How to start MS-LINK

MS-LINK requires two types of input: a command to start MS-LINK and responses to command prompts. In addition, seven switches control MS-LINK features. Usually, you will type all the commands to MS-LINK on the terminal keyboard. As an option, answers to the command prompts and any switches may be contained in a response file. Command characters can be used to assist you while giving commands to MS-LINK.

MS-LINK may be started in any of three ways. The first method is to type the commands in response to individual prompts. In the second method, you type all commands on the line used to start MS-LINK. To start MS-LINK by the third method, you must create a response file that contains all the necessary commands and tell MS-LINK where that file is when you start MS-LINK.

Summary of Methods to start MS-LINK

Method 1 LINK prompted.
Method 2 LINK <filenames> [/switches]
Method 3 LINK <filespec>

10.5.1 Method 1: Prompts

To start MS-LINK with Method 1, type:

```
LINK
```

MS-LINK will be loaded into memory. MS-LINK will then display four text prompts that appear one at a time. You answer the prompts to command MS-LINK to perform specific tasks.

At the end of each line, you may type one or more switches, preceded by the switch character, a forward slash (/)

The command prompts are summarised below and described in more detail in the "Command Prompts" section.

Object Modules [.OBJ]:
: Input '.OBJ' files to be linked. They must be separated by blank spaces or plus signs (+). If a plus sign is the last character typed, the prompt will reappear. There is no default; a response is required.

Run File [Object-file.EXE]:
: Give filename for executable object code. The default is first-object-filename.EXE. (You cannot change the output extension.)

List File [Run-file.MAP]:
: Give filename for listing. The default is RUN filename.

Libraries []:
: List filenames to be searched, separated by blank spaces or plus signs (+). If a plus sign is the last character typed, the prompt will reappear. The default is to search for default libraries in the object modules. (Extensions will be changed to .LIB.)

10.5.2 Method 2: Command Line

To start MS-LINK with Method 2, type all commands on one line. the entries following LINK are responses to the command prompts. The entry fields for the different prompts must be separated by commas. Use the following syntax:

```
LINK <object-list>, <runfile>, <listfile>, <lib-list> [/switch...]
```

where :-

- *object-list* is a list of object modules, separated by plus signs
- *runfile* is the name of the file to receive the executable output
- *listfile* is the name of the file to receive the listing.
- *lib-list* is a list of library modules to be searched.

/switch refers to optional switches, which may be placed following any of the response entries (just before any of the commas or after the *<lib-list>*, as shown.

To select the default for a field, simply type a second comma with no spaces between the two commas.

Example:

```
LINK FUN+TEXT+TABLE+CARE/P/M+FUNLIST,COBLIB.LIB
```

This command causes MS-LINK to be loaded, then the object modules FUN.OBJ, TEXT.OBJ, TABLE.OBJ, and CARE.OBJ are loaded. MS-LINK then pauses (as a result of using the /P switch). MS-LINK links the object modules when you press any key, and produces a global symbol map (the /M switch); defaults to FUN.EXE Run file; creates a List file named FUNLIST.MAP; and searches the library file COBLIB.LIB.

10.5.3 Method 3: Response File

To start MS-LINK with Method 3, type:

```
LINK @<filespec>
```

where: *filespec* is the name of a response file. A response file contains answers to MS-LINK prompts (shown in Method 1) and may also contain any of the switches. When naming a response file, the use of filename extensions is optional. Method 3 permits the command that starts MS-LINK to be entered from the keyboard or within a batch file without requiring you to take any further action.

To use this option, you must create a response file containing several lines of text, each of which is the response to an MS-LINK prompt. The response must be in the same order as the MS-LINK prompts discussed in Method 1. If desired, a long response to the Object Modules: or Libraries: prompt may be typed on several lines by using a plus sign (+) to continue the same response onto the next line.

Use switches and command characters in the response file the same way as they are used for responses typed on the terminal keyboard.

When the MS-LINK session begins, each prompt will be displayed in order with the responses from the response file. If the response file does not contain answers for all the prompts, (in the form of filenames, the semicolon character or carriage returns), MS-LINK will display the prompt which does not have a response, then wait for you to type a legal response. When a legal response has been typed, MS-LINK continues the link session.

Example Response File:

```
FUN TEXT TABLE CARE
/PAUSE/MAP
FUNLIST
COBLIB.LIB
```

This response file tells MS-LINK to load the four object modules named FUN, TABLE and CARE. MS-LINK pauses before producing a public symbol map to permit you to swap disks (see discussion under /PAUSE in the "Switches" section before using this feature). When you press

any key, the output files will be named FUN.EXE and FUNLIST.MAP. MS-LINK will search the library file COBLIB.LIB, and will use the default settings for the switches.

10.6 COMMAND CHARACTERS

MS-LINK provides three command characters:

Plus sign

Use the plus sign (+) to separate entries and to extend the current line in response to the Object Modules: and Libraries: prompts. (A blank space may be used to separate object modules.) To type a large number of responses (each may be very long), type a plus sign followed by the [RETURN] key at the end of the line to extend it. If the plus sign is the last character typed before pressing the [RETURN] key, MS-LINK will prompt you for more module names. When the prompt 'Object Modules:' or the prompt 'Libraries:' appears again, continue to type responses. When all the modules to be linked and libraries to be searched have been input, be sure the response line ends with the last module name and a [RETURN] and not a plus sign and [RETURN].

Example:

```
Object Modules [.OBJ]: FUN TEXT TABLE CARE <RETURN>
Object Modules [.OBJ]: FOO+FLIPPFLOP+JUNQUE <RETURN>
Object Modules [.OBJ]: CORSAIR <RETURN>
```

Semicolon (;) :

To select default responses to the remaining prompts, use a single semicolon (;) followed by a [RETURN] at any time after the first prompt (Run file:). This feature saves time and overrides the need to press a series of [RETURN] keys.

Once the semicolon has been typed and entered (by pressing the [RETURN] key), you can no longer respond to any of the prompts for that link session. Therefore, do not use the semicolon to skip some prompts. To skip prompts, use the [RETURN] key.

Example:

```
Object Modules [.OBJ]: FUN TEXT TABLE CARE [RETURN]
Run Module [FUN.EXE]: ;[RETURN]
```

No other prompts will appear, and MS-LINK will use the default values (including FUN.MAP for the List file).

<CONTROL-C>

Use the [Ctrl][C] keys to abort the link session at any time. If you type an erroneous response, such as the wrong filename or an incorrectly spelled filename, you must press [Ctrl][C] to exit MS-LINK then restart MS-LINK. If the error has been typed but you have not pressed the [RETURN] key, you may delete the erroneous characters with the backspace key, but for that line only.

10.7 Command Prompts

MS-LINK asks you for responses to four text prompts. When you have typed a response to a prompt and pressed [RETURN], the next prompt appears. When the last prompt has been answered, MS-LINK begins linking automatically without further command. when the link session is finished, MS-LINK exits to the operating system. When the operating system prompt appears, Ms-LINK has finished successfully. If the link session is unsuccessful, MS-LINK will display the appropriate error message.

MS-LINK prompts you for the names of Object, Run, and List files, and for Libraries. The prompts are listed in order of appearance. The default response is shown in square brackets ([ ]) following the prompt for prompts which can default to preset responses. The object Modules: prompt has no preset filename response and requires you to type a filename.

Object Modules [.OBJ]:

Type a list of the object modules to be linked. MS-LINK assumes by default that the filename extension is .OBJ. If an object module has any other filename extension, the extension must be given. Otherwise, the extension may be omitted.

Modules must be separated by plus signs (+).

Remember that MS-LINK loads segments into classes in the order encountered. You can use this information to set the order in which the object modules will be read by MS-LINK.

Run File [First-Object-filename.EXE]:

Typing a filename will create a file for storing the Run (executable) file that results from the link session. All Run files receive the filename extension .EXE, even if you specify an extension other than .EXE.

If no response is typed to the 'Run File:' prompt, MS-LINK uses the first filename typed in response to the Object Modules: prompt as the RUN filename.

Example:

Run File [FUN.EXE]: B:PAYROLL/P

This response directs MS-LINK to create the Run file PAYROLL.EXE on drive B:. Also MS-LINK will pause, which allows you to insert a new disk to receive the Run file.

List File [Run-Filename.MAP]:

The List file contains an entry for each segment in the input (object) modules. Each entry also shows the addressing in the Run file.

The default response is the Run filename with the default filename extension '.MAP'.

Libraries []:

The valid responses are up to eight library filenames or simply a [RETURN] (A [RETURN] means default library search). Library files must have been created by a library utility. In default, MS-LINK will assume that the filename extension is .LIB for library files.

Library filenames must be separated by blank space or plus signs (+).

MS-LINK searches library files in the order listed to resolve external references. When it finds the module that defines the external symbol, MS-LINK processes that module as another object module.

If MS-LINK cannot find a library file on the disk drives, it will display the message:

Cannot find library <library-name> Type new drive letter:

Press the letter for the drive designation (for example, B).

10.8 MS-LINK SWITCHES

The seven MS-LINK switches control various MS-LINK functions. Switches must be typed at the end of a prompt response, regardless of which method is used to start MS-LINK. Switches may be grouped at the end of any response, or may be scattered at the end of several. If more than one switch is typed at the end of one response, each switch must be preceded by a forward slash (/).

All switches may be abbreviated. The only restriction is that an abbreviation must be sequential from the first letter through the last typed and no gaps or transpositions are allowed. Some legal and illegal abbreviations for the /DSALLOCATE switch are as follows:

Legal	Illegal
/D	/DSL
/DS	/DAL
/DSA	/DLC
/DSALLOCA	/DSALLOCT
/DSALLOCATE	

Using the /DSALLOCATE switch tells MS-LINK to load all data at the high end of the Data Segment. Otherwise MS-LINK loads all data at the low end of the Data Segment. At run time, the DS pointer is set to the lowest possible address to allow the entire DS segment to be used. Use of the /DSALLOCATE switch in combination with the default load low (that is, the /HIGH switch is not used) premits the user application to dynamically allocate any available memory below the area specifically allocated with DGroup, yet to remain accessible by the same DS pointer. This dynamic allocation is needed for Pascal and FOTRAN programs.

Note:

Your application program may dynamically allocate up to 64K bytes (or the actual amount of memory available) less the amount allocated within DGroup.

/HIGH

Use of the /High switch causes MS-LINK to place the Run file as high as possible in memory. Otherwise, MS-Link places the Run file as low as possible.

IMPORTANT

Do not use the /High switch with Pascal or FORTRAN programs.

/LINENUMBERS

The /LINENUMBERS switch tells MS-LINK to include in the List file the line numbers and addresses of the source statements in the input modules. Otherwise, line numbers are not included in the List file.

Note:

Not all compilers produce object modules that contain line number information. In these cases, of course, MS-LINK cannot include line numbers.

/MAP

/Map directs MS-LINK to list all public (global) symbols defined in the input modules. If /MAP is not given, MS-LINK will list only errors (including undefined globals).

The symbols are listed alphabetically. For each symbol, MS-LINK lists its value and its segment:offset location in the Run file. The symbols are listed at the end of the List file.

/PAUSE

The /PAUSE switch causes MS-LINK to pause in the link session when the switch is encountered. Normally, MS-LINK performs the linking session from beginning to end without stopping. This switch allows the user to swap the disks before MS-LINK outputs the Run (.EXE) file.

When MS-LINK encounters the /PAUSE switch, it displays the message:

About to generate .EXE file Change disks <hit any key>

MS-LINK resumes processing when the user presses any key.

CAUTION

Do not remove the disk which will receive the List file, or the disk used for the VM.TMP file, if one has been created.

/STACK:<number>

Number represents any positive numeric value (in hexadecimal radix) up to 65536 bytes. If a value from 1 to 511 is typed, MS-LINK will use 512. If the /STACK switch is not used for a link session, MS-LINK will calculate the necessary stack size automatically.

All compilers and assemblers should provide information in the object modules that allow the linkter to compute the required stack size.

At least one object (input) module must contain a stack allocation statement. If not, MS-LINK will display the following error message:

WARNING: NO STACK SEGMENT

/NO

/NO is short for NO DEFAULT LIBRARY SEARCH. This switch tells MS-LINK to not search the default (product) libraries in the object modules. For example, if you are linking object modules in Pascal, specifying the /NO switch tells MS-LINK to not automatically search the library named PASCAL.LIB to resolve external references.

10.9 SAMPLE MS-LINK SESSION

This sample shows you the type of information that is displayed during an MS-LINK session.

In response to the MS-DOS prompt, type:

```
LINK
```

The system displayes the following messages and prompts:

MICROSOFT Object Linker V.2.00 (C) Copyright 1982 by Microsoft Inc.

Object Modules [.OBJ]: `IO SYSINT` <RETURN>
Run File [IO.EXE]: <RETURN> list file [NUL.MAP]: `IO /MAP` <RETURN>
Libraries [.LIB]: ;<RETURN>

Notes:

1. By specifying /MAP, you get both an alphabetic listing and a chronological listing of public symbols.
2. By responding PRN to the List File: prompt, you can redirect your output to the printer.
3. By specifying the /LINE switch, MS-LINK gives you a listing of all line numbers for all modules. (Note that the /LINE switch can generate a large volume of output.)
4. By pressing <RETURN> in response to the Libraries: prompt, an automatic library search is performed.

Once MS-LINK locates all libraries, the linker map displays a list of segments in the order of their appearance within the load module. The list might look like this:

```
Start   Stop    Length  Name
00000H  009ECH  09EDh   CODE
009F0H  01166H  0777H   SYSINTSEG
```

The information in the Start and Stop columns shows the 20-bit hex address of each segment relative to location zero. Location zero is the beginning of the load module.

The addresses displayed are not the absolute addresses where these segments are loaded. Consult the MS-DOS 2.0 Macro Assembler Manual for information on how to determine where relative zero is actually located, and also on how to determine the absolute address of a segment.

Because the /MAP switch was used, MS-LINK displays the public symbols by name and value. For example:

```
ADDRESS         PUBLICS BY NAME

009F:0012       BUFFERS
009F:0005       CURRENT_DOS_LOCATION
009F:0011       DEFAULTDRIVE
009F:000B       DEVICE_LIST
009F:0013       FILES
009F:0009       FINAL_DOS_LOCATION
009F:000F       MEMORY_SIZE
009F:0000       SYSINIT

ADDRESS         PUBLICS BY VALUE

009F:0000       SYSINIT
009F:0005       CURRENT_DOS_LOCATION
009F:0009       FINAL_DOS_LOCATION
009F:000B       DEVICE_LIST
009F:000F       MEMORY_SIZE
009F:0011       DEFAULTDRIVE
009F:0012       BUFFERS
009F:0013       FILES
```

10.10 ERROR MESSAGES

All errors cause the link session to abort. After the cause has been found and corrected, MS-LINK must be re-run. The following messages are displayed by MS-LINK:

Attempt to access data outside of segment bounds, possibly bad object module
: There is probably a bad Object file.

Bad numeric parameter
: Numeric value is not in digits.

Cannot open temporary file
: MS-LINK is unable to create the file VM.TMP because the disk directory is full. Insert a new disk. Do not remove the disk that will receive the List.MAP file.

Error: dup record too complex
: DUP record in assembly language module is too complex. Simplify DUP record in assembly language program.

Error: Fixup offset exceeds field width
: An assembly language instruction refers to an address with a short instruction instead of a long instruction. Edit assembly language source and reassemble.

Input file read error
: There is probably a bad Object file

Invalid object module
: An object module(s) is incorrectly formed or incomplete (as when assembly is stopped in the middle).

Symbol defined more than once
: MS-LINK found two or more modules that define the same symbol.

Program size or number of segments exceeds capacity of linker
: The total size may not exceed 384K bytes and the number of segments may not exceed 255.

Requested stack size exceeds 64K
: Specify a size greater than or equal to 64K bytes with the /STACK switch.

Segment size exceeds 64K
: 64K bytes is the addressing system limit.

Symbol table capacity exceeded
: Very many and/or very long names were typed, exceeding the limit of approximately 25K bytes.

Too many external symbols in one module
: The limit is 256 external symbols per module.

Too many groups
: The limit is 10 groups.

Too many libraries specified.
: The limit is 8 libraries.

Too many public symbols

The limit is 1024 public symbols.

Too many segments or classes

The limit is 256 (segments and classes being taken together).

Unresolved externals: <list>

The external symbols listed have no defining module among the modules or library files specified.

VM read error

This is a disk error; it is not caused by MS-LINK.

Warning: no stack segment

None of the object modules specified contains a statement allocating stack space, but you typed the /STACK switch.

Warning: segment of absolute or unknown type

There is a bad object module or an attempt has been made to link modules that MS-LINK cannot handle (e.g. an absolute object module).

Write error in tmp file

No more disk space remains to expand the VM.TMP file.

Write error on run file

Usually, there is not enough disk space for the Run file.

Appendix 11: COMMAND.COM

When MS-DOS is initially loaded the system commands processor, COMMAND.COM is loaded and becomes the resident command line processor. You can however call the command processor by using, the syntax below:

command [<drive:> <pathname>] [<cttydev>] [/p] [/c <string>] [/e <environment size>]

This command starts a new command processor (the MS-DOS program that contains all internal commands).

The command processor is loaded into memory in two parts: the transient part and the resident part. Some application programs write over the transient part of COMMAND.COM when they run. When this happens, the resident part of the command processor looks for the COMMAND.COM file on disk so it can reload the transient part.

The <drive:> <pathname> options tell the command processor where to look for the COMMAND.COM file it it needs to reload the transient part into memory.

<cttydev> allows you to specify a different device (such as aux) for input and output. See the Ctty command in this chapter for more information.

The /e switch specifies the environment size in bytes. The size may range between 128 and 32768 bytes. The default value is 128 bytes.

If <environment size> is less than 128 bytes, MS-DOS defaults to 128 bytes and gives the message:

Invalid environment size specified

If <environment size> is greater than 32768 bytes, MS-DOS gives the same message, but defaults to 32768 bytes.

The /p switch tells COMMAND.COM not to exit to any higher level.

The /c switch, if used, should be the last switch in the command. It tells the command processor to execute the command or commands specified by <string> and then return.

Example:

command /c chkdsk b:

This example tells the command processor to:

1. Start a new command processor under the current program.
2. Run the command "chkdsk b:"
3. Return to the first command processor.

Appendix 12: The DEBUG Utility Program (DEBUG)

12.1 INTRODUCTION

The Microsoft DEBUG Utility (DEBUG) is a debugging program that provides a controlled testing environment for binary and executable object files. Note that EDLIN is used to alter source files; DEBUG is EDLIN's counterpart for binary files. DEBUG eliminates the need to reassmeble a program to see if a problem has been fixed by a minor change. It allows you to alter the contents of a file or the contents of a CPU register, and then to immediately re-execute a program to check on the validity of the changes.

All DEBUG commands may be aborted at any time by pressing [Control][C]. [Control][S] suspends the display, so that you can read it before the output scrolls away. Entering any key other than [Control][C] or [Control][S] restarts the display. All of these commands are consistent with

the control character functions available at the MS-DOS command level.

12.2 HOW TO START DEBUG

DEBUG may be started two ways. By the first method, you type all commands in response to the DEBUG prompt (a hyphen). By the second method, you type all commands on the line used to start DEBUG.

Summary of Methods to Start DEBUG

Method 1
    DEBUG
Method 2
    DEBUG [<filespec> [<arglist>]]

12.2.1 Method 1: DEBUG

To start DEBUG using method 1, type:

    DEBUG

DEBUG responds with the hyphen (-) prompt, signaling that it is ready to accept your commands. Since no filename has been specified, current memory, disk sectors or disk files can be worked on by using other commands.

WARNINGS

1. When DEBUG (Version 2.0) is started, it sets up a program header at offset 0 in the program work area. On previous versions of DEBUG, you could overwrite this header. You can still overwrite the default header if no <filespec> is given to DEBUG. If you are debugging a .COM or .EXE file, however, do not tamper with the program header below address 5CH, or DEBUG will terminate.
2. Do not restart a program after the message, Program terminated normally is displayed. You must reload a program with the N and L commands for it to run properly.

12.2.2 Method 1: Command Line

To start DEBUG using a command line, type:

    DEBUG [<filespec> [<arglist>]]

For example, if a <filespec> is specified, then the following is a typical command to start DEBUG:

    DEBUG FILE.EXE

DEBUG then loads FILE.EXE into memory starting at 100 hexadecimal in the lowest available segment. The BX:CX registers are loaded with the number of bytes placed into memory.

An <arglist> may be specified if <filespec> is present. The <arglist> is a list of filename parameters and switches that are to be passed to the program <filespec>. Thus, when <filespec> is loaded into memory, it is loaded as if it had been started with the command:

    <filespec> <arglist>

Here, <filespec> is the file to be debugged, and the <arglist> is the rest of the command line that is used when <filespec> is invoked and loaded into memory.

12.3 COMMAND INFORMATION

Each DEBUG command consists of a single letter followed by one or more parameters. Additionally, the control characters and the special editing functions described in the MS-DOS User's Guide, apply inside DEBUG.

If a syntax error occurs in a DEBUG command, DEBUG reprints the command line and indicates the error with an up-arrow (↑) and the word "error."

For example:

```
dcs:100 cs:110
        ^  Error
```

Any combination of uppercase and lowercase letters may be used in commands and parameters.

The DEBUG commands are summarized in Table 11.1 and are described in detail, with examples, following the description of command parameters.

Table 12.1 DEBUG Commands

DEBUG Command	Function
A[<address>]	Assemble
C<range> [<address>]	Compare
D[<range>]	Dump
E<address> [<list>]	Enter
F<range> <list>	Fill
G[= <address> [<address> ...]]	Go
H<value> <value>	Hex
I<value>	Input
L[<address> [<drive:> <record> <record>]]	Load
M <range> <address>	Move
N <filename> [<filename>]	Name
O <value> <byte>	Output
Q	Quit
R[<register-name>]	Register
S<range> <list>	Search
T[= <address> [<value>]]	Trace
U[<range>]	Unassemble
W[<address> [<drive:> <record> <record>]]	Write

PARAMETERS

All DEBUG commands accept parameters, except the Quit command. Parameters may be separated by delimiters (spaces or commas), but a delimiter is required only between two consecutive hexadecimal values. Thus, the following commands are equivalent:

```
dcs:100 110
d cs:100 110
d,cs:100 110
```

PARAMETER DEFINITION

<drive> A one-digit hexadecimal value to indicate which drive a file will be loaded from or written to. the valid values are 0-3. These values designate the drives as follows: 0=A:, 1=B:, 2=C:, 3=D:.

<byte> A two-digit hexadecimal value to be placed in or read from an address or reigster.

<record> A 1- to 3-digit hexadecimal value to be used to indicate the logical record number on the disk and the number of disk sectors to be written or loaded. Logical records correspond to sectors. However, their numbering differs since they represent the entire disk space.

<value> A hexadecimal value up to four digits used to serial a port number or the number of times a command should repeat its functions.

<address> A two-part designation consisting of either an alphabetic segment register designation or a four-digit segment address plus an offset value. The segment designation or segment address may be omitted, in which case the default segment is used. DS is the default segment for all commands except G, L, T, U and W, for which the default segment is CS. All numeric values are hexadecimal.

For example:

```
CS:0100
04BA:0100
```

The colon is required between a segment designation (whether numeric or alphabetic) and an offset.

<range> Two <address>es: e.g., <address> <address>; or one <address>, an L, and a <value> e.g. <address> L <value> where:

<value> is the number of lines the command should operate on, and L80 is assumed. The last form cannot be used if another hex value follows the <range>, since the hex value would be interpreted as the second <address> of the <range>.

Examples:

```
CS:100 110
CS:100 L 10
CS:100
```

The following is illegal:

```
CS:100 CS:110
       ^ Error
```

PARAMETER DEFINITION

The limit for <range> is 10000 hex. To specify a value of <10000> hex within four digits, type 00 (or 0).

A series of <byte> values or of <string>s. <list> must be the last parameter on the command line.

<list> Example:

fcs:100 42 45 52 54 41

Any number of characters enclosed in quote marks. Quote marks must be either single (') or double ("). If the delimiter quote marks must appear within a <string>, the quote marks must be doubled. For example, the following strings are legal:

'This is a "string" is okay.'
'This is a "string" is okay.'

However, this string is illegal:

'This is a 'string' is not.'

Similarly, these strings are legal:

<string> "This is a 'string' is okay."
"This is a ""string"" is okay."

However, this string is illegal:

"This is a "string" is not."

Note that the double quote marks are not necessary in the following strings:

"This is a "string" is not necessary."
'This is a ""string"" is not necessary.'

The ASCII values of the characters in the string are used as a <list> of byte values.

NAME Assemble
PURPOSE Assembles 8086/8087/8088 mnemonics directly into memory.
SYNTAX A[<address>]

If a syntax error is found, DEBUG responds with

^ Error

and re-displays the current assembly address.

All numeric values are hexadecimal and must be entered as 1-4 characters. Prefix mnemonics must be specified in front of the opcode to which they refer. They may also be entered on a separate line.

The segment override mnemonics are CS:, DS:, ES:, and SS:. The mnemonic for the far return is RETF. String manipulation mnemonics must explicitly state the string size. For example, use MOVSW to move word strings and MOVSB to move byte strings.

The assembler will automatically assemble short, near or far jumps and calls, depending on byte displacement to the destination address. These may be overridden with the NEAR or FAR prefix. For example:

```
0100:0500 JMP 502       ;a 2-byte short jump
0100:0502 JMP NEAR 505  ;a 3-byte near jump
0100:0505 JMP FAR 50A   ;a 5-byte far jump
```

The NEAR prefix may be abbreviated to NE, but the FAR prefix cannot be abbreviated.

DEBUG cannot tell whether some operands refer to a word memory location or to a byte memory location. In this case, the data type must be explicitly stated with the prefix 1WORD PTR1 or 1BYTE PTR1. Acceptable abbreviations are 1WO1 and 1BY1. For example:

```
NEG     BYTE PTR [128]
DEC     WO [SI]
```

DEBUG also cannot tell whether an operand refers to a memory location or to an immediate operand. DEBUG uses the common convention that operands enclosed in square brackets refer to memory. For example:

```
MOV     AX,21   ;Load AX with 21H
MOV     AX,[21] ;Load AX with the
                ;contents
```

```
                        ;of memory location 21H
```

Two popular pseudo-instructions are available with Assemble. The DB opcode will assemble byte values directly into memory. The DW opcode will assemble word values directly into memory. For example:

```
        DB      1,2,3,4,"THIS IS AN EXAMPLE"
        DB      'THIS IS A QUOTE:"'
        DB      "THIS IS A QUOTE:'"
        DW      1000,2000,3000,"BACH"
```

Assemble supports all forms of register indirect commands. For example:

```
        ADD     BX,34[BP+2].[SI-1]
        POP     [BP+DI]
        PUSH    [SI]
```

All opcode synonyms are also supported. For example:

```
        LOOPZ   100
        LOOPE   100

        JA      200
        JNBE    200
```

For 8087 opcodes, the WAIT or FWAIT must be explicitly specified. For example:

```
FWAIT FADD ST,ST(3)     ;This line assembles
                        ;an FWAIT prefix
LD TBYTE PTR [BX]       ;This line does not
```

NAME Compare
PURPOSE Compares the portion of memory specified by <range> to a portion of the same size beginning at <address>.
SYNTAX C<range> <address>

If the two areas of memory are identical, there is no display and DEBUG returns with the MS-DOS prompt. If there are differences, they are displayed in this format:

<address1> <byte1;> <byte2;> <address2>

For example, The following commands have the same effect:

C100,1FF 300

or

C100 L100 300

Each command compares the block of memory from 100 to 1FFH with the block of memory from 300 to 3FFH.

NAME Dump
PURPOSE Displays the contents of the specified region of memory.
SYNTAX D[<range>]

If a range of addresses is specified, the contents of the range are displayed. If the D command is typed without parameters, 128 bytes are displayed at the first address (DS:100) after the address displayed by the previous Dump command.

The dump is displayed in two portions: a hexadecimal dump (each byte is shown in hexadecimal value) and an ASCII dump (the bytes are shown in ASCII characters). Nonprinting characters are denoted by a period (.) in the ASCII portion of the display. Each display line shows 16 bytes with a hyphen between the eighth and ninth bytes. At times, displays are split in this manual to fit them on the page. Each displayed line begins on a 16-byte boundary.

If you type the command:

dcs:100 10F

DEBUG displays the dump in the following format:

```
        04BA:0100 54 4F 4D 20 53 41 57 59-45 52 24 00 00 09 4E 44  TOM SAWYER$...ND
```

If you type the following command:

D

The display is formatted as described above. Each line of the display begins with an address, incremented by 16 from the address on the previous line. Each subsequent D (typed without parameters) displays the bytes immediately following those last displayed.

If you type the command:

```
DCS:100 L20
```

the display is formatted as described above, but 20H bytes are displayed.

If then you type the command:

```
DCS:100 115
```

the display is formatted as described above, but all the bytes in the range of lines from 100H to 115H in the CS segment are displayed.

NAME Enter
PURPOSE Enters byte values into memory at the specified <address>.
SYNTAX E<address> [<list>]

If the optional <list> of values is typed, the replacement of byte values occurs automatically. (If an error occurs, no byte values are changed.)

If the <address> is typed without the optional <list>, DEBUG displays the address and its contents, then repeats the address on the next line and waits for your input. At this point, the Enter command waits for you to perform one of the following actions:

1. Replace a byte value with a value you type. Simply type the value after the current value. If the value typed in is not a legal hexadecimal value, or if more than two digits are typed, the illegal or extra character is not echoed.
2. Press the [SPACEBAR] to advance to the next byte. To change the value, simply type the new value as described in (1.) above. If you space beyond an 8-byte boundary, DEBUG starts a new display line with the address displayed at the beginning.
3. Type a HYPHEN (-) to return to the preceding byte. If you decide to change a byte behind the current position, typing the hyphen returns the current position to the previous byte. When the hyphen is typed, a new line is started with the address and its byte value displayed.
4. Press the [RETURN] key to terminate the Enter command. The [RETURN] key may be pressed at any byte position.

For example, assume that the following command is typed:

```
ECS:100
```

DEBUG displays:

```
04BA:0100 EB .
```

To change this value to 41, type 41 as shown:

```
04BA:0100 EB.41
```

To step through the subsequent bytes, press the Spacebar to see:

```
04BA:0100 EB.41   10.     00.     BC.
```

To change BC to 42:

```
04BA:0100 EB.41   10.     00.     BC.42
```

Now, realizing that 10 should be 6F, type the hyphen as many times as needed to return to byte 01 (value 10), then replace 10 with 6F:

```
04BA:0100 EB.41   10.     00.     BC.42-
04BA:0102 00.-
04BA:0101 10.6F
```

Pressing the [RETURN] key ends the Enter command and returns to the DEBUG command level.

NAME Fill
PURPOSE Fills the addresses in the <range> with the values in the <list>.
SYNTAX F<range> <list>

If the <range> contains more bytes than the number of values in the <list>, the <list> will be used repeatedly until all bytes in the <range> are filled. If the <list> contains more values than the number of bytes in the <range>, the extra values in the <list> will be ignored. If any of the memory in the <range> is not valid (bad or nonexistent), the error wil occur in all succeeding locations.

For example, assume that the following command is typed:

```
F 04BA:100 L 100 57 4B 57 42 41
```

DEBUG fills memory locations 04BA:100 through 04BA:1FF with the bytes specified. The five values are repeated until all 100H bytes are filled.

NAME Go
PURPOSE Executes the program currently in memory.
SYNTAX G[=<address> [<address> ...]]

If only the Go command is typed, the program executes as if the program had run outside DEBUG.

If =<address> is set, execution begins at the address specifed. The equal sign (=) is required, so that DEBUG can distinguish the start =<address> from the breakpoint <address>es.

With the other optional addresses set, execution stops at the first <address> encountered, regardless of that address's position in the list of addresses to halt execution or program branching. When program execution reaches a breakpoint, the registers, flags and decoded instruction are displayed for the last instruction executed. (The result is the same as if you had typed the Register command for the breakpoint address.)

Up to ten breakpoints may be set. Breakpoints may be set only at addresses containing the first byte of an 8086 opcode. If more than ten breakpoints are set, DEBUG returns the BP Error message.

The user stack pointer must be valid and have 6 bytes available for this command. The G command uses an IRET instruction to cause a jump to the program under test. The user stack pointer is set, and the user flags, Code Segment register, and Instruction Pointer are pushed on the user stack. (Thus, if the user stack is not valid or is too small, the operating system may crash.) An interrupt code (0CCH) is placed at the specified breakpoint address(es).

When an instruction with the breakpoint code is encountered, all breakpoint addresses are restored to their original instructions.

If execution is not halted at one of the breakpoints, the interrupt codes are not replaced with the original instructions.

For example, assume that the following command is typed:

```
GCS:7550
```

The program currently in memory executes up to the address 7550 in the CS segment. DEBUG then displays registers and flags, after which the Go command is terminated.

After a breakpoint has been encountered, if you type the Go command again, then the program executes just as if you had typed the filename at the MS-DOS command prompt level. The only difference is that program execution begins at the instruction after the breakpoint rather than at the usual start address.

NAME Hex
PURPOSE Performs hexadecimal arithmetic on the two parameters specified.
SYNTAX H<value> <value>

First, DEBUG adds the two parameters, then subtracts the second parameter from the first. The results of the arithmetic are displayed on one line; first the sum, then the difference.

For example, assume that the following command is typed:

```
H19F 10A
```

DEBUG performs the calculations and then displays the result:

```
02A9 0095
```

NAME Input
PURPOSE Inputs and displays one byte from the port specified by value.
SYNTAX I<value>

A 16-bit port address is allowed.

For example, assume that you type the following command:

```
I2F8
```

Assume also that the byte at the port is 42H. DEBUG inputs the byte and displays the value:

```
42
```

NAME Load
PURPOSE Loads a file into memory.
SYNTAX L<address> [<drive:> <record> <record>]

Set BX:CX to the number of bytes read. The file must have been named either when DEBUG was started or with the N command. Both the DEBUG invocation and the N command format a filename properly in the normal format of a file control block at CS:5C.

If the L command is typed without any parameters, DEBUG loads the file into memory beginning at address CS:100 and sets BX:CX to the number of bytes loaded. If the L command is typed with an address parameter, loading begins at the memory <address> specified. If L is typed with all parameters, absolute disk sectors are loaded, not a file. The <record>s are taken from the <drive:> specified (the drive designation is numeric here--0=A:, 1=B:, 2=C:, etc.); DEBUG begins loading with the first <record> specified, and continues until the number of sectors specified in the second <record> have been loaded.

Assume that the following commands are typed:

```
A>DEBUG
-NFILE.COM
```

Now, to load FILE.COM, type:

```
L
```

DEBUG loads the file and then displays the DEBUG prompt. Assume that you want to load only portions of a file or certain records from a disk. To do this, type:

```
L04BA:100 2 0F 6D
```

DEBUG then loads 109 (6D hex) records, beginning with logical record number 15, into memory beginning at address 04BA:100. When the records have been loaded, DEBUG simply returns the - prompt.

If the file has a .EXE extension, it is relocated to the load address specified in the header of the .EXE file: the <address> parameter is always ignored for .EXE files. The header itself is stripped off the .EXE file before it is loaded into memory. Thus the size of an .EXE file on disk will differ from its size in memory.

If the file named by the Name command or specified when DEBUG is started is a .HEX file, then typing the L command with no parameters causes DEBUG to load the file beginning at the address specified in the .HEX file. If the L command includes the option <address>, DEBUG adds the <address> specified in the L command to the address found in the .HEX file to determine the start address for loading the file.

NAME Move
PURPOSE Moves the block of memory specified by <range> to the location beginning at the address specified.
SYNTAX M<range> <address>

Overlapping moves (i.e., moves where part of the block overlaps some of the current addresses) are always performed without loss of data. Addresses that could be overwritten are moved first. The sequence for moves from higher addresses to lower addresses is to move the data beginning at the block's lowest address and then to work towards the highest. The sequence for moves from lower addresses to higher addresses is to move the data beginning at the block's highest address and to work towards the lowest.

Note that if the addresses in the block being moved will not have new data written to them, the data there before the move will remain. The M command copies the data from one area into another, in the sequence described, and writes over the new addresses. This is why the sequence of the move is important.

Assume that you type:

```
MCS:100 110 CS:500
```

DEBUG first moves address CS:110 to address CS:510, then CS:10F to CS:50F, and so on until CS:100 is moved to CS:500. You should type the D command, using the <address> typed for the M command, to review the results of the move.

NAME Name
PURPOSE Sets filenames.
SYNTAX N<filename> [<filename> ...]

The NAME command performs two functions. Firstly, NAME is used to assign a filename for a later Load or Write command. Thus, if you start DEBUG without naming any file to be debugged, then the N<filename> command must be typed before a file can be loaded. Secondly, NAME is used to assign filename parameters to the file being debugged. In this case, Name accepts a list of parameters that are used by the file being debugged.

These two functions overlap. Consider the following set of DEBUG commands:

```
-NFILE1.EXE
-L
-G
```

Because of the effects of the NAME command, NAME will perform the following steps:

1. (N)ame assigns the filename FILE1.EXE to the filename to be used in any later Load or Write commands.
2. (N)ame also assigns the filename FILE1.EXE to the first filename parameter used by any program that is later debugged.
3. (L)oad loads FILE1.EXE into memory.
4. (G)o causes FILE1.EXE to be executed with FILE1.EXE as the single parameter (that is, FILE1.EXE is executed as if FILE1.EXE had been

typed at the command level).

A more useful chain of commands might look like this:

```
-NFILE1.EXE
-L
-NFILE2.DAT FILE3.DAT
-G
```

Here, Name sets FILE1.EXE as the filename for the subsequent Load command. The Load command loads FILE1.EXE into memory, and then the Name command is used again, this time to specify the parameters to be used by FILE1.EXE. Finally, when the Go command is executed, FILE1.EXE is executed as if FILE1 FILE2.DAT FILE3.DAT had been typed at the MS-DOS command level. Note that if a Write command were executed at this point, then FILE1.EXE--the file being debugged--would be saved with the name FILE2.DAT! To avoid such undesired results, you should always execute a Name command before either a Load or a Write.

There are four regions of memory that can be affected by the Name command:

CS:5C
    FCB for file 1
CS:6C
    FCB for file 2
CS:80
    Count of characters
CS:81
    All characters typed

A File Control Block (FCB) for the first filename parameter given to the Name command is set up at CS:5C. If a second filename parameter is typed, then an FCB is set up for it beginning at CS:6C. The number of characters typed in the Name command (exclusive of the first character, "N") is given at location CS:80. The actual stream of characters given by the Name command (again, exclusive of the letter "N") begins at CS:81. Note that this stream of characters may contain switches and delimiters that would be legal in any command typed at the MS-DOS command level.

A typical use of the NAME command is:

```
DEBUG PROG.COM
-NPARAM1 PARAM2/C
-G
-
```

In this case, the GO command executes the file in memory as if the following command had been typed:

```
PROG PARAM1 PARAM2/C
```

Testing and debugging therefore reflect a normal runtime environment for PROG.COM.

NAME Output
PURPOSE Sends the <byte> specified to the output port specified by <value.>
SYNTAX O<value> <byte>

A 16-bit port address is allowed.

For example, typing:

```
O2F8 4F
```

causes DEBUG to output the byte value 4F to output port 2F8.

NAME Quit
PURPOSE Terminates the DEBUG utility.
SYNTAX Q

The Q command takes no parameters, and exits DEBUG without saving the file currently being operated on. You are returned to the MS-DOS command level.

For example, to end the debugging session, type:

```
Q<Return>
```

DEBUG has been terminated, and control returns to the MS-DOS command level.

NAME Register
PURPOSE Displays the contents of one or more CPU registers.

SYNTAX R[<register-name>]

If no <register-name> is typed, the R command dumps the register save area and displays the contents of all registers and flags.

If a register name is typed, the 16-bit value of that register is displayed in hexadecimal, and then a colon appears as a prompt. You then either type a <value> to change the register, or simply press the [RETURN] key if no change is wanted.

The only valid <register-name>s are:

AX BP SS
BX SI CS
CX DI IP (IP and PC both refer to the Instruction Pointer)
DX DS PC
SP ES F

Any other entry for <register-name> results in a br Error message.

If F is entered as the <register-name>, DEBUG displays each flag with a two-character alphabetic code. To alter any flag, type the opposite two-letter code. The flags are either set or cleared.

The flags are listed below with their codes for SET and CLEAR:

FLAG NAME	SET	CLEAR
Overflow	OV	NV
Direction	DN Decrement	UP Increment
Interrupt	EI Enabled	DI Disabled
Sign	NG Negative	PL Plus
Zero	ZR	NZ
Auxiliary Carry	AC	NA
Parity	PE Even	PO Odd
Carry	CY	NC

Whenever you type the command RF, the flags are displayed in the order shown above in a row at the beginning of a line. At the end of the list of flags, DEBUG displays a hyphen (-). You may enter new flag values as alphabetic pairs. The new flag values can be entered in any order. You do not have to leave spaces between the flag entries. To exit the R command, press the [RETURN] key. Flags for which new values were not entered remain unchanged.

If more than one value is entered for a flag, DEBUG returns a DF Error message. If you enter a flag code other than those shown above, DEBUG returns a BF Error message. In both cases, the flags up to the error in the list are changed; flags at and after the error are not.

At startup, the segment registers are set to the bottom of free memory, the Instruction Pointer is set to 0100H, all flags are cleared, and the remaining registers are set to zero.

For example, typing:

```
R
```

causes DEBUG to display all registers, flags and the decoded instruction for the current location. If the location is CS:11A, then the display will look similar to this:

```
AX=0E00  BX=00FF  CX=0007  DX=01FF  SP=039D  BP=0000
SI=005C  DI=0000  DS=04BA  ES=04BA  SS=04BA  CS=04BA
IP=011A   NV UP DI NG NZ AC PE NC
04BA:011A CD21        INT 21
```

If you type:

```
RF
```

DEBUG will display the flags:

```
NV UP DI NG NZ AC PE NC  -
```

Now, type any valid flag designation, in any order, with or without spaces.

```
NV UP DI NG NZ AC PE NC  -PL EI CY<RETURN>
```

DEBUG responds only with the DEBUG prompt. To see the changes, type either the R or RF command:

```
RF
NV UP EI PL NZ AC PE CY  -
```

Press the [RETURN] key to leave the flags this way, or to specify different flag values.

NAME Search
PURPOSE Searches the <range> specified for the <list> of bytes specified.
SYNTAX S<range> <list>

The <list> may contain one or more bytes, each separated by a space or comma. If the <list> contains more than one byte, only the first address of the byte string is returned. If the <list> contains only one byte, all addresses of the byte in the <range> are displayed.

If you type:

```
SCS:100 110 41
```

DEBUG displays a response similar to this:

```
04BA:0104
04BA:010D
-
```

NAME Trace
PURPOSE Executes one instruction and displays the contents of all registers and flags, and the decoded instruction.
SYNTAX T[=<address>][<value>]

If the optional =<address> is typed, tracing occurs at the =<address> specified. The optional <value> causes DEBUG to execute and trace the number of steps specified by <value>

The T command uses the hardware trace mode of the 8086 or 8088 microprocessor. Consequently, you may also trace instructions stored in ROM (Read Only Memory).

For example if you type:

```
T
```

DEBUG returns a display of the registers, flags and decoded instruction for that one instruction. Assume that the current position is 04BA:011A; DEBUG might return the display:

```
AX=0E00  BX=00FF  CX=0007  DX=01FF  SP=039D  BP=0000
SI=005C  DI=0000  DS=04BA  ES=04BA  SS=04BA  CS=04BA
IP=011A   NV UP DI NG NZ AC PE NC
04BA:011A CD21        INT 21
```

If you type:

```
T=011A 10
```

DEBUG executes sixteen (10 hex) instructions beginning at 011A in the current segment, and then displays all registers and flags for each instruction as it is executed. The display scrolls away until the last instruction is executed. Then the display stops, and you can see the register and flag values for the last few instructions performed. Remember that [Control][S] suspends the display at any point, so that you can study the registers and flags for any instruction.

NAME Unassemble
PURPOSE Disassembles bytes and displays the source statements that correspond to them, with addresses and byte values.
SYNTAX U[<range>]

The display of disassembled code looks like a listing for an assembled file. If you type the U command without parameters, 20 hexadecimal bytes are disassembled at the first address after that displayed by the previous Unassemble command. If you type the U command with the <range> parameter, then DEBUG disassembles all bytes in the range. If the <range> is given as an <address> only, then 20H bytes are disassembled.

If you type:

```
U04BA:100 L10
```

DEBUG disassembles 16 bytes beginning at address 04BA:0100:

```
04BA:0100 206472      AND     [SI+72],AH
04BA:0103 69          DB      69
04BA:0104 7665        JBE     016B
04BA:0106 207370      AND     [BP+DI+70],DH
04BA:0109 65          DB      65
04BA:010A 63          DB      63
04BA:010B 69          DB      69
```

```
04BA:010C 66          DB      66
04BA:010D 69          DB      69
04BA:010E 63          DB      63
04BA:010F 61          DB      61
```

If you type:

```
U04ba:0100 0108
```

the display shows:

```
04BA:0100 206472      AND     [SI+72],AH
04BA:0103 69          DB      69
04BA:0104 7665        JBE     016B
04BA:0106 207370      AND     [BP+DI+70],DH
```

If the bytes in some addresses are altered, the disassembler alters the instruction statements. The U command can be typed for the changed locations, the new instructions viewed, and the disassembled code used to edit the source file.

NAME Write
PURPOSE Writes the file being debugged to a disk file.
SYNTAX W[<address> [<drive:> <record> <record>]]

If you type W with no parameters, BX:CX must already be set to the number of bytes to be written; the file is written beginning from CS:100. If the W command is typed with just an address, then the file is written beginning at that address. If a G or T command has been used, BX:CX must be reset before using the Write command without parameters. Note that if a file is loaded and modified, the name, length, and starting address are all set correctly to save the modified file (as long as the length has not changed).

The file must have been named either with the DEBUG information command or with the N command (refer to the Name command earlier in this manual). Both the DEBUG invocation and the N command format a filename properly in the normal format of a file control block at CS:5C.

If the W command is typed with parameters, the write begins from the memory address specified; the file is written to the <drive:> specified (the drive designation is numeric here--0=A:, 1=B:, 2=C: etc.); DEBUG writes the file beginning at the logical record number specified by the first <record> DEBUG continues to write the file until the number of sectors specified in the second <record> have been written.

Warning

Writing to absolute sectors is EXTREMELY dangerous because the process bypasses the file handler.

If you type:

```
NEXAMPLE.DAT
W
```

and

```
BX=0001 CX=03B7,
```

DEBUG displays a message reporting the file size:

```
Writing 103B7 bytes
```

Then DEBUG writes the file (EXAMPLE.DAT) to disk and displays the DEBUG prompt (-) when finished.

If you type:

```
WCS:100 1 37 2B
```

DEBUG writes out the contents of memory, beginning with the address CS:100 to the disk in drive B:. The data written out starts in disk logical record number 37H and consists of 2BH records. When the write is complete, DEBUG displays the prompt.

12.5 ERROR MESSAGES

During the DEBUG session, you may receive any of the following error messages. Each error terminates the DEBUG command under which it occurred, but does not terminate DEBUG itself.

ERROR CODE	DEFINITION
BF	Bad flag

ERROR CODE	DEFINITION

You attempted to alter a flag, but the characters typed were not one of the acceptable pairs of flag values. See the Register command for the list of acceptable flag entries.

BP	Too many breakpoints

You specified more than ten breakpoints as parameters to the G command. Re-type the Go command with ten or fewer breakpoints.

BR	Bad register

You typed the R command with an invalid register name. See the Register command for the list of valid register names.

DF	Double flag

You typed two values for one flag. You may specify a flag value only once per RF command.

Appendix 13: The EXE2BIN Utility Program

The EXE2BIN utility program converts .EXE (executable) files to BINary format.

SYNTAX

exe2bin [<drive:>]<pathname> [<drive>][<pathname>]

This command is useful only if you want to convert .EXE (executable) files to binary format. The file named by <pathname> is the input file. If no extension is specified, it defaults to .EXE. The input file is converted to .BIN file format (memory image of the program) and placed in the output file (second pathname). If you do not specify a drive name, the drive of the input file will be used. If you do not specify a filename extension in the output filename, the new file will be given an extension of .BIN.

The input file must be in valid .EXE format produced by the linker. The resident, or actual code and data part of the file must be less than 64K. There must be no STACK segment.

Two kinds of conversions are possible, depending on whether the initial CS:IP (Code Segment: Instruction Pointer) is specified in the .EXE file:

1. If CS:IP is not specified in the .EXE file, a pure binary conversion is assumed. If segments fixups are necessary (that is, the program contains instructions requiring segment relocation), you will be prompted for the fixup value. This value is the absolute segment at which the program is to be loaded. The resultant program will be usable only when loaded at the absolute memory address specified by a user application. The command processor will not be able to load the program.
2. If CS:IP is 00:100H, it is assumed that the file will run as a .COM file with the location pointer set at 100H by the assembler statement ORG; the first 100H bytes of the file are deleted. No segment fixups are allowed, as .COM files must be segment relocatable; that is, they must assume the entry conditions explained in the Microsoft Macro Assembler Manual. Once the conversion is complete, you may rename the output file with a .COM extension. Then the command processor will be able to load and execute the program in the same way as the .COM programs supplied on you MS-DOS disk.

MESSAGES:

File cannot be converted

CS:IP does not meet either of the criteria specified above, or it meets the .COM file criterion but has segment fixups. This message is also displayed if the file is not a valid executable file.

- File not found

- The file is not on the disk specified.

- Insufficient memory

- There is not enough memory to run Exe2bin.

- File creation error

Exe2bin cannot create the output file. Run Chkdsk to determine if the directory is full, or if some other condition caused the error.

- Insufficient disk space

- There is not enough disk space to create a new file.

- Fixups needed - base segment (hex):

The source (.EXE) file contained information indicating that a load segment is required for the file. Specify the absolute segment address at which the finished module is to be located.

- File cannot be converted.

The input file is not in the correct format.

WARNING - Read error in EXE file. Amount read less than size in header

This is a warning message only. It means that the .EXE header is inconsistent with the size of the file.

Appendix 14: The EXIT Command

The EXIT command exits the program COMMAND.COM (the command processor) and returns to a previous level, if one exists.

SYNTAX

exit

This command can be used when you are running an application program and want to start the MS-DOS command processor, then return to

your program. For example, to look at a directory on drive B while running an application program, you must issue an EXEC of the command interpreter (system call 4BH). The system prompt will appear. You can now type the Dir command and MS-DOS will display the directory listing. When you type EXIT, you return to the previous level (your application program).

Appendix 15: The RECOVER Utility Program

The RECOVER utility recovers a file or an entire disk containing bad sectors.

SYNTAX

recover [<drive:>]
or
recover <drive:>[<pathname>]

If a sector on a disk is bad, you can recover either the file containing that sector (without the bad sector) or the entire disk (if the bad sector was in the directory).

To recover a particular file, type:

recover <filename>

This causes MS-DOS to read the file sector by sector and to skip the bad sector(s). When MS-DOS finds the bad sector(s), the sector(s) are marked and MS-DOS will no longer allocate your data to that sector.

To recover a disk, type

recover <drive:>

where <drive:> is the letter of the drive containing the disk to be recovered.

MESSAGES:

File not found
: MS-DOS cannot find the file that you specified. Check to see that the pathname is accurate and that the file exists in the directory you specified.

(xxxx) of (xxxx) bytes recovered
: This message tells you the number of bytes that MS-DOS was able to recover from the disk.

Warning - directory full
: The root directory is too full for Recover processing. Delete some files in the root directory to free space.

Appendix 16: The SHARE Utility Program

The SHARE utility installs file sharing and locking.

SYNTAX

share [/f:<space:>][/L:<locks>]

The SHARE command is only used when networking is active. It is included in the AUTOEXEC.BAT file to install shared files. Refer to the Microsoft Networks Manager's Guide to learn about shared files.

Use the /f:<space> switch to allocate file space (in bytes) for the area MS-DOS uses to record filesharing information. Each file that is onpen needs the length of the full filename plus 11 bytes (the average pathname is 20 bytes). The default value for the /f switch is 2048 bytes.

The L:<locks> switch allocates the number of locks you want to allow. The default value for the /L switch is 20 locks.

Once you have used the SHARE command in an MS-DOS session, all read and write requests are checked by MS-DOS.

For example, the following example loads file sharing and uses the default values for the /f and /L switches:

share

SHARE messages:

- Incorrect parameter
: One of the options you specified is wrong.

- Not enough memory
: There is not enough memory for MS-DOS to run the command.

- Share Already Installed
: You can install Share only once.

INDEX