# The Amstrad PPC Technical Reference Manual

SOFT 50031

AMSTRAD

# Preface

This Technical Reference Manual is intended primarily to assist writers of software for the Amstrad PPC, although in conjunction with the PPC 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 Colour Graphics Adapter (or Monochrome 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 PPC User Instructions and the Microsoft MSDOS Reference Manual.

Whilst the PPC implements a superset of the Industry Standard, this manual makes no attempt to identify those areas of the PPC 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.

Please note, that communications regulations do not allow the release of technical information relating to the circuitry or construction of the PC2000 modem.

All correspondence should be addressed to:

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

Written by Bill Weidenauer and edited by Susan Vass, Amstrad plc.

Published by Amstrad.

First Published 1987.

Cover Index Contents

# Table of Contents:

## Section 1 - Hardware

| | |
|---|---|
| 1.0 Introduction | 1 |
| 1.1 Central Processing Unit (CPU) | 1 |
| 1.2 Memory Layout | 2 |
| 1.3 Main Board I/O Channels | 3 |
| 1.4 Expansion Bus I/O Channels | 5 |
| 1.5 Direct Memory Access (DMA) | 5 |
| 1.5.1 DMA Page Registers | 6 |
| 1.5.2 DMA Initialization | 7 |
| 1.6 System Interrupts | 7 |
| 1.6.1 Interrupt Levels | 8 |
| 1.6.2 Interrupt Controller Initialization | 8 |
| 1.6.3 NMI Mask Control | 9 |
| 1.7 Programmable Interval Timers | 9 |
| 1.7.1 Timer Configuration | 10 |
| 1.7.2 Counter 1 initialization | 10 |
| 1.8 System Status and Control | 10 |
| 1.8.1 Port B - System Control | 11 |
| 1.8.2 Port A - Status-1 Input / Keyboard Code | 12 |
| 1.8.3 Port C - Status-2 Input | 13 |
| 1.8.4 Write System Status-1 | 14 |
| 1.8.5 Write System Status-2 | 14 |
| 1.8.6 System Reset | 14 |
| 1.9 Real Time Clock | 15 |
| 1.10 Parallel Printer Port | 15 |
| 1.10.1 Printer Data Latch | 16 |
| 1.10.2 Printer Control Latch | 16 |
| 1.10.3 Printer Status Channel | 17 |
| 1.11 The Internal display Adapter | 18 |
| 1.11.1 Colour Alpha Display | 19 |
| 1.11.2 Colour Graphics Display | 21 |
| 1.11.2.1 Low Resolution Graphics | 21 |
| 1.11.2.2 Medium Resolution Graphics | 22 |
| 1.11.2.3 High Resolution (640 X 200) Graphics Mode | 23 |
| 1.11.3 Monochrome Display | 23 |
| 1.11.4 BIOS Modes | 24 |
| 1.11.5 Colour Mode Registers. | 25 |
| 1.11.5.1 CGA Mode Control Register | 25 |

1.11.5.2 Colour Select Register 26
1.11.5.3 CGA Status Register 27
1.11.5.4 CGA Mode CRTC initialization. 27
1.11.5.5 Set and Clear Light Pen Latch 29
1.11.5.6 Operation Control/Status Register 29
1.11.5.7 CRTC Index Readback 30
1.11.5.8 CRTC Data Readback 30
1.11.6 Monochrome Mode Registers. 31
1.11.6.1 MDA Mode Control Register 31
1.11.6.2 MDA Status Register 32
1.11.6.3 Monochrome CRTC initialization. 32
1.12 Floppy Disk Controller 34
1.12.1 FDC Hardware Conditions 34
1.13 RS232C Asynchronous Serial Port 35
1.13.1 Serial Channnel Interface 35
1.13.2 Serial Channnel Pin Arrangement 36
1.14 Parallel Printer Interface 36
1.15 Keyboard Interface 38
1.15.1 Serial Clock and Serial Data 38
1.15.2 Keyboard to Main Board Interface 38
1.15.3 Main Board to Keyboard Interface 38
1.15.4 Keycodes 39
1.16 Modem Interface 39
1.16.1 Modem Connector 39
1.17 Light Pen Connector 40
1.18 Expansion Interface Connectors 41
1.19 Video Connector 44
1.20 Power Connectors 45
1.20.1 The Adapter Power Connector 45
1.20.2 The Display Power Connector 45
1.21 PPC Switch Settings 46

2.0 Firmware 47
2.1 Power-Up Initialization and Self Test 47
2.2 Power-Up Self Tests 52
2.2.1 Test Procedure. 52
2.2.2 Test Methods. 53
2.2.3 ROS Checksum Test. 53
2.2.4 Direct Memory Access Controller test. 54
2.2.5 Programmable Interval Timer test. 54
2.2.6 Programmable Peripheral Interface test. 54
2.2.7 Real Time Clock test. 54
2.2.8 Asynchronous Communications Element test. 54
2.2.9 Printer Parallel Port test. 54
2.2.10 System RAM test. 54

2.2.11 Programmable Interrupt Controller test. 55
2.2.12 Disk test. 55
2.2.13 Keyboard Interface test. 55
2.3 ROM Firmware Interrupts. 55
2.3.1 Interrupt 2: Parity Error (NMI). 55
2.3.2 Interrupt 5: Print Screen. 56
2.3.3 Interrupt 8: System Clock Interrupt 56.
2.3.4 Interrupt 9: Keyboard Interrupt. 57
2.3.4.1 Special Key Actions 61
2.3.5 Interrupt 14: Floppy Disk Controller. 62
2.3.6 Interrupt 16: Video I/O. 62
2.3.7 Interrupt 17: System Configuration. 71
2.3.8 Interrupt 18: Memory Size. 72
2.3.9 Interrupt 19: Disk I/O. 72
2.3.9.1 Hard Disk Call Parameters and Registers 77
2.3.10 Interrupt 20: Serial I/O. 79
2.3.11 Interrupt 22: Keyboard I/O. 83
2.3.12 Interrupt 23: Printer I/O. 87
2.3.13 Interrupt 24: System Restart. 88
2.3.14 Interrupt 24: Disk Bootstrap. 89
2.3.15 Interrupt 26: System Clock & Real Time Clock. 89
2.3.16 Interrupt 27: Keyboard Break Interrupt. 93
2.3.17 Interrupt 28: External Ticker Interrupt. 93
2.3.18 Interrupt 29: Video Parameter Table. 93
2.3.19 Interrupt 30: Disk Parameter Table. 95
2.3.20 Interrupt 31: Video Matrix Table. 95
2.4 System RAM Variables. 96
2.5 Non Volatile RAM. 103
2.6 ROS Messages. 104
2.6.1 Non-Fatal ROS Messages. 104
2.6.2 Fatal ROS Messages. 105

3.0 Reference Information. 106
3.1 Language Links. 106
3.2 Processor Memory Usage. 106
3.3 Asynchronous Communications Element (8250) Registers. 107
3.4 High Performance Programmable DMA Controller (8237A-4) Registers 110
3.5 Programmable Interrupt Controller (8259A-2) Command Words. 112
3.6 Programmable Interval Timer (8253) Registers. 115
3.7 Real Time Clock (HD146818) Registers 116
3.7.1 Time, Calendar and Alarm Locations 117
3.7.2 RTC Register Locations 119
3.8 Floppy Disk Controller (uPD765A). 121

| | |
|---|---|
| Appendix 1: MS-DOS System Configuration | 137 |
| Appendix 2: Country Dependent Information for MS-DOS 3 | 144 |
| Appendix 3: RS232C Connections | 145 |
| Appendix 4: Printer Lead (PL-2) Wiring Specification | 153 |
| Appendix 5: PPC Power Usage | 155 |
| Appendix 6: ROM Character Set | 156 |
| Appendix 7: Keyboard Keycodes | 158 |
| Appendix 8: Keyboard Layouts | 159 |
| Appendix 9: The LINKER Program | 164 |
| Appendix 10: COMMAND.COM | 179 |
| Appendix 11: The DEBUG Utility Program (DEBUG) | 180 |
| Appendix 12: The EXE2BIN Utility Program | 202 |
| Appendix 13: The EXIT Command | 204 |
| Appendix 14: The RECOVER Utility Program | 205 |
| Appendix 15: The SHARE Utility Program | 206 |
| Appendix 16: The FDISK Utility Program | 207 |
| Appendix 17: The BACKUP Command | 218 |
| Appendix 18: The RESTORE Command | 221 |

# Amstrad PPC Technical Manual

## 1.0 Introduction

This manual provides a comprehensive description of the Amstrad PPC hardware and firmware. General information about the PPC and the delivered operating system software is contained in the Amstrad PPC User Instructions. This manual is intended to satisfy the needs of advanced developers who must have access to the various resources available within the PPC640 and PPC512.

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 a low power 8086-2 microprocessor with 1 Megabyte memory addressing capability (See Figure 1.1), 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 µS |
| 16 to 8-bit convert (Memory) | 12 | 2.0 µS |
| 8-bit (I/O) | 6 | 1.25 µS |
| 16 to 8-bit convert (I/O) | 16 | 2.5 µS |

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. Note that the PPC512 has 512K bytes installed memory ending at address 7FFFFh and the address space from 80000h to 9FFFFh may be extended in external 32K byte blocks (up to the 640K maximum).

The 128K byte address space from A0000 to BFFFF is reserved for video regeneration buffers, and is not used by CPU programs. The PPC Internal Display Adapter (IDA) uses the 64K byte address range from B0000 to BFFFF. The segmentation of this memory range is dependent on the display mode (See section 1.11). External display adapters also use this memory address range for their display buffers.

The 192K byte address space from C0000 to EFFFF is reserved for external expansion ROM address space. Hard Disk controllers use the range from C8000 to C9FFF. Additional hard disk controllers may also use the area from CA000 to CD000. The PPC 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.

**MEMORY LAYOUT**

| | | | |
|---|---|---|---|
| 00000 | 9FFFF | ON-BOARD DYNAMIC RAM | 640K Byte System Memory |
| | A0000 BFFFF | 128K BYTES VIDEO DISPLAY BUFFERS | |
| 1M BYTE ADDR RANGE | C0000 EFFFF | 192K BYTES EXPANSION ROMS | |
| | F0000 FBFFF | 48K BYTES ROS ROM BLOCK REPEATS | 64K Byte System ROM area |
| | FC000 | 16K BYTES (ROS) RESIDENT OPERATING SYSTEM ROM | |
| FFFFF | | | |

## 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 | Reserved | Reserved |

| ADDRESS(hex) | OUTPUT USE | INPUT USE |
|---|---|---|
| 079 | Do Not Use | Do Not Use |
| 07A | Reserved | Reserved |
| 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 |
| 2F8 - 2FF | Modem UART Tx Data/Control | Modem UART Rx Data/Control |
| 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 - 3B7 | Mono Mode CRTC Registers | Mono Mode CRTC Registers |
| 3B8 - 3BF | Mono Mode Control Registers | Mono Mode CRTC Registers |
| 3D0 - 3D7 | Colour Mode CRTC Registers | Colour Mode CRTC Registers |
| 3D8 - 3DF | Colour Mode Control Registers | Colour Mode Control 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 | COM1 8250 UART Tx Data/Control | COM1 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 - 2FF | Reserved |
| 300 - 31F | External Prototyping Card |
| 320 - 32F | External Hard Disk Controller |

| ADDRESS(hex) | USE |
|---|---|
| 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 Colour/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 PPC 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 1.25µS. Channel 0 transfers have a four-clock bus cycle of 1µS.

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 |

| Bit | Output Use |
|---|---|
| 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<br>Autoinitialise<br>Increment<br>Single Mode |
| Command Register | Disable Memory to Memory<br>Enable Controller<br>Normal Timing<br>Fixed Priority<br>Late Write<br>DREQ Active High<br>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 PPC. 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. |

| Level | Assigned Function |
|---|---|
| 2 | 146818 Real Time Clock IRQ output.<br>Available on the expansion bus.<br>May be used by Enhanced Graphics Adapter |
| 3 | Used by Modem Serial Port (COM2)<br>and external SDLC Serial Port.<br>Available on the expansion bus. |
| 4 | Primary Serial port (COM1).<br>Available on the expansion bus.<br>Used by external SDLC Serial Port. |
| 5 | Hard Disk Controller. Available on the expansion bus. |
| 6 | 765 Floppy Disk Controller INT output.<br>Available on the expansion bus. |
| 7 | Parallel Printer Port.<br>Available on the expansion bus.<br>Used by external Printer Port (secondary) and Printer Port (ternary) on external Monochrome VDU Controller. |

### 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 respectively. 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 µS. 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. |

| Bit (PBn) | Output Use |
|---|---|
| 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 2 operations 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 | External Extended Graphics Adapter Installed |
| 0 | 1 | Colour Graphics Adapter Emulation (or external CGA) with alpha, 40 X 25 chars, bright white on black. |
| 1 | 0 | Colour Graphics Adapter Emulation (or external CGA) with alpha, 80 X 25 chars, bright white on black. |
| 1 | 1 | Monochrome Display Adapter Emulation (or external MDA) with alpha, 80 X 25 chars. |

When the Internal Display Adapter (IDA) is enabled then switches 1 - 3 determine the IDA configuration and switches 4 & 5 determine the default display mode. See section 1.22 for these details.

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. |

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. |

### 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 512µS 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 the battery pack. 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 is used to store time of last usage. The ROS firmware maintains a checksum of the NVR and will reset the RTC to default configuration during startup whenever the checksum value is incorrect.

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.

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 2.5).

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 | PPC SW 4 (inverted) [RO] | | |
| 6 | PPC SW 5 (inverted) [RO] | | |
| 5 | FDC Change Line [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 7 is a read-only bit which reflects the state of switch 4 (at the side of the machine) and returns a logic "0" when the switch is in the "on" position and a logic "1" for the "off" position.

Bit 6 is a read-only bit which reflects the state of switch 5 and returns a logic "0" when the switch is in the "on" position and a logic "1" for the "off" position.

Bit 5 is diskette change line status and is true (logic "1") when the selected floppy disk's door is open or no diskette is installed. This signal is latched on the drive and remains true until the disk drive is selected and it receives a step pulse from the FDC. This bit is only valid when a floppy drive is active and selected.

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 1µS before the strobe is made active, and must remain valid for at least 1µS after strobe goes inactive. The strobe duration must be between 1µS and 500 µS. 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. |

| LK1 | LK2 | LK3 | ROS Language |
|---|---|---|---|
| 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 PPC Internal Display Adapter.

The Internal Display Adapter (IDA) is implemented using a gate array and a 6845 CRTC. It provides video for either the internal Liquid Crystal Display (LCD) or video output to a standard (PC1640) video connector at the rear of the PPC. The IDA is completely software compatible with the industry standard MDA & CGA environments and when outputting to an external display (in VDU mode) the video signals and scan frequencies are compatible with industry standard monitors.

The IDA provides either a Colour Graphics Adapter (CGA) emulation mode or a Monochrome Display Adapter (MDA) emulation mode. The video screen memory (or regeneration buffer) starting address is configured to B8000 for CGA emulation mode and to B0000 for MDA emulation mode.

The CGA display mode supports text or graphics in 16 colours on a Colour Display (CD) with a maximum resolution of 640 dots by 200 lines. Since colours cannot be displayed on the internal LCD a pixel will either be ON or OFF depending on whether it is the foreground colour or the background colour.

The MDA display mode supports black and white text in normal, intense, inverse, blink, or underline with a character cell resolution of 8 x 14. There are no graphics modes in MDA emulation mode. When using the internal LCD in MDA emulation mode the software environment is the same as that of a standard MDA device, but the character cell resolution is 8 x 8, intensified or highlighted characters are displayed with the vertical bars of each character one dot thinner than they would otherwise be and underlining is not supported.

In CGA emulation mode, the colour palette consists of sixteen colours as follows:

| 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 |

| Intensity | Red | Green | Blue | Colour |
|---|---|---|---|---|
| 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 |

### 1.11.1 Colour Alpha Display

In colour mode, two Alpha modes are available: either 40 characters by 25 rows or 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 ROS bios supports up to 8 or 4 separate display pages for 40 and 80 column modes respectively. This same display mapping is also repeated at BC000h.

The character set is formed by a ROM character generator and each of the 256 characters is made up of a 8 by 8 pixel matrix.

The starting address in the display RAM is programmed via the 6845 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 6 gives the 256 character codes and their respective default character representations.

The attribute byte allows a choice of either 16 foreground and 8 background colours per character, plus blinking, or a choice of 16 colours for both foreground and background without blinking.

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) |
| 1 | Green (Foreground) |
| 0 | Blue (Foreground) |

Bit 7, the Intensity or Enable Blink Bit, changes function based the Mode Control Register. The Mode Control Register (I/O address 03D8h) bit 5 selects between Intensity or Blink.

When driving the internal LCD display in colour modes, the foreground and background colours are given the weightings such that Intensity = 1, Blue = 2, Red = 4, Green = 8 and compared. If the background has a higher value than the foreground the character is inverted. In addition, if bit 7 of the CGA mode control register (section 1.11.5) is set characters with non-black backgrounds are also inverted. These two inversion processes are cumulative so that characters may be doubly inverted and appear non-inverted. As with MDA characters on the LCD, text with an intensified foreground will have verticals one dot thinner.

## 1.11.2 Colour Graphics Display

CGA emulation mode supports graphics with 200 scan lines with a choice of two horizontal resolutions, either 320 pixels per scan line with four colours per pixel or 640 pixels per scan line with a two colours per pixel.

An additional low resolution mode is also supported with 100 scan lines and a horizontal resolution of 160 pixels per scan line with sixteen colours per pixel.

The regeneration buffer for colour graphics modes starts at B8000 and requires 16K bytes and is repeated at BC000. The ROS supports either 8 or 4 display buffer pages for the 320 or 640 resolution modes respectively.

### 1.11.2.1 Low Resolution Graphics

Low resolution graphics mode is actually text mode and not a true bit mapped graphics mode in the sense that high and medium resolution graphics are. The CRTC is programmed such that there are 100 lines on the screen and the mode register is set up the same as text mode. There is no support for this mode in the ROS and the CRTC must be initialized as described in section 1.11.5.4.

In Low Resolution Graphics Mode (160x100), each logical pixel is mapped to a block of 4 x 2 physical pixels on the LCD. Each logical pixel also has 4 bits to determine which of the 16 colours it should be displayed in. These 4 bits are mapped directly to both rows of 4 physical pixels representing the logical pixels in question this providing 16 different pixel patterns to represent the 16 colours on the LCD.

The mapping of display ram in this graphics mode is actually an extension of the text mode scheme. The even bytes are all filled with a special graphics character, 0DEh, which contains a vertical bar. The odd bytes are then programmed in two four bit pairs with the leftmost pixel being the four high order bits. Each display line therefore requires 160 bytes and for the full 100 line display 16000 bytes are required.

The mapping of the odd byte(s) of graphics RAM in low resolution mode is as follows:

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

### 1.11.2.2 Medium Resolution Graphics

In Medium Resolution Graphics Mode (320x200), 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 in left to right fashion. The two bit field for each pixel specifies one of four colours 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 |

In Medium Resolution LCD mode, each logical pixel is mapped to a pair of physical pixels on the LCD rather than a single pixel of a particular colour as described above. Changing palettes will not, therefore, change the display.

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 layout of a byte of graphics RAM in medium 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 High Resolution (640 x 200) Graphics Mode**

In High 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 colours are either black (pixel bit off) or pixel bit on with video in one of the 16 colours 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.

### 1.11.3 Monochrome Display

In monochrome mode there is only one Alpha mode available, 80 characters by 25 rows. When driving an external monochrome display, the character output is 8 x 14 pixel characters, whereas on the LCD a 8 x 8 character matrix is used.

The display RAM requirement is 4K bytes of display regeneration buffer area from B0000h to B0FFFh and is repeated in 1000h boundaries up to and including B7000h.

The 6845 CRTC Starting Address is programmed in the same way as in colour modes and the two byte character and attribute pairs are arranged in the display RAM just as in the colour modes. The attribute byte, however assumes different functions from the colour attributes since there are no IRGB signals sent to the monitor, but Video and 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 on the Mode control register. The Mode control Register (I/O address 03B8h) Bit 5 selects between Intensity or Blink Functions.

Bit 3 is the foreground intensity bit and controls the intensity when not in inverse video or blanked. The internal LCD displays higher intensity characters using a hardware algorithm to display the vertical lines one dot thinner than in normal intensity. Also, underlining is not supported on the LCD.

### 1.11.4 BIOS Modes

The ROS sets up the hardware to support eight different modes for the various displays available on the PPC range. The following table gives the modes supported by the ROS.

| ROS Mode | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| Type | Text | Text | Text | Text | Graph | Graph | Graph | Text |
| Columns | 40 | 40 | 80 | 80 | 320 | 320 | 640 | 80 |
| Rows | 25 | 25 | 25 | 25 | 200 | 200 | 200 | 25 |
| Colour(s) | 16 | 16 | 16 | 16 | 4 | B/W | 2 | Mono |
| Char Cell Size | 8x8 | 8x8 | 8x8 | 8x8 | 8x8 | 8x8 | 8x8 | 8x14 |
| Regen Origin | B8000 | B8000 | B8000 | B8000 | B8000 | B8000 | B8000 | B0000 |
| Regen Size | 16384 | 16384 | 16384 | 16384 | 16384 | 16384 | 16384 | 4096 |
| Page Size | 2048 | 2048 | 4096 | 4096 | 16384 | 16384 | 16384 | 4096 |
| Number Pages | 8 | 8 | 4 | 4 | 2 | 2 | 2 | 1 |

The Regeneration Buffer Origin is stated in hexadecimal notation since it is an address quantity. All other values are in decimal notation.

The ROS Video I/O 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.

## 1.11.5 Colour Mode Registers

In colour mode there are seven registers for mode, colour selection, additional control functions.

### 1.11.5.1 CGA Control Register

The CGA Mode Control Register is a read/write 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. On an industry standard CGA this is a write only register.

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

| Bit | Output Use |
|---|---|
| 7 | LCD Inverse |
| 6 | LCD Lo-Res Graphics |
| 5 | Enable Blinking Chars (disable intensified backgrounds) |
| 4 | Select Graphics Mode 2 (de-select graphics mode 1) |
| 3 | Enable Video Display |
| 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) |

Bit 0 is used for controlling 40 or 80 column text selection. When bit 0 is reset to "0" 40 column text is enabled. When bit 0 is set to "1" then 80 column text is enabled. Bit 0 has no effect Graphics modes.

Bit 1 selects between text and graphics modes. When bit 0 is reset to "0" then text mode is selected and when bit 0 is set to "1" then graphics mode is selected.

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 VDU 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 the VDU mode control register should then be set.

When bit 3 is reset to "0" the display is blanked by disabling foreground video. When bit 3 is set to "1" foreground video is enabled.

Bit 4 Selects between high resolution and medium resolution graphics modes. When bit 4 is set to "1" then 640x200 graphics resolution is selected and when bit 4 is reset to "0" then 320x200 graphics resolution is selected. Bit 4 has no effect in Alpha modes.

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 6 provides hardware support for the CGA low resolution 160 x 100 sixteen colour graphics mode. It has no effect in external video mode but in LCD mode it enables each of the 16 colours to be displayed as a unique dot pattern.

Bit 7 adds an extra programmable feature to the interpretation of CGA text attributes when displayed on the LCD. When bit 7 is set to "1" in text mode, characters with non-black backgrounds will be displayed in inverse video. This inversion process is in addition to the non-programmable inversion which takes place place when the background colour is greater than the foreground. A character which is inverted by both of these processes will appear non-inverted. This inversion process effects the entire text screen on a character by character basis. This bit has no effect when driving an external CD display.

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 Colour Select Register

The CGA Colour 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-colour graphics mode the overall screen palette is controlled by setting the CGA Colour 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 (always reset). |
| 3 | Frame Flyback (VSYNC) Time. |
| 2 | Light-pen switch made. |
| 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 = "1" when the light-pen switch is closed and bit 2 = "0" 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 CRTC initialization.

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. Note that this set of I/O addresses is partially decoded such that all even I/O addresses between 3D0 - 3D6 access the CRTC index register and all odd I/O address between 3D1 - 3D7 access the selected CRTC data register.

The PPC gate array implementation incorporates a special "Auto-Switch" feature which enables the LCD to appear as a standard display device despite the fact that non-standard CRTC settings are required when dealing with the LCD. The auto-switch prevents CRTC registers 0 - 11 from being altered when bit 6 of the operation control port is set. Any attempt to write to these registers are trapped and PPC unique readback ports are provided so that the ROS's NMI service routine can readback the values written to these registers. Any write access to either the prohibited CRTC registers, the Operation Control port or the CGA control port will trigger a Non-Maskable Interrupt (NMI) when bit 7 of the Operation Control port is set. Bits 5-7 of the CRTC Index readback register can then be read by the NMI service routine to determine which of the three events triggered the NMI. In this way the NMI service routine is alerted to any attempted mode change and can change modes itself by loading the CRTC registers with the appropriate values for the operation in the desired LCD mode. The auto-switch also prevents accidental (non-ROS) changes to bits 6 and 7 of the CGA Control port.

The CRTC registers must be programmed according to the mode of operation required in conjunction with the CGA Mode and Colour 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 Colour select registers as required, (maintaining video disabled), initialize display RAM as required, enable video.

The CRTC registers are initialized for LCD CGA mode as follows:

| Register Number (Hex) | Register Name | Alpha 40 Char Mode (Decimal) | Alpha 80 Char Mode (Decimal) | Graphics Low-res (Decimal) | Graphics High-res (Decimal) |
|---|---|---|---|---|---|
| 00 | Horizontal Total (-1) | 41 | 83 | 83 | 43 |
| 01 | Horizontal Displayed | 40 | 80 | 80 | 40 |
| 02 | Horiz. Sync Position | 41 | 81 | 81 | 42 |
| 03 | Horiz. Sync Width | 01 | 01 | 01 | 01 |
| 04 | Vertical Total | 26 | 26 | 107 | 107 |
| 05 | Vertical Total Adjust | 0 | 0 | 0 | 0 |
| 06 | Vertical Displayed | 26 | 26 | 107 | 107 |

| Register Number (Hex) | Register Name | Alpha 40 Char Mode (Decimal) | Alpha 80 Char Mode (Decimal) | Graphics Low-res (Decimal) | Graphics High-res (Decimal) |
|---|---|---|---|---|---|
| 07 | Vertical Sync Position | 25 | 25 | 100 | 100 |
| 08 | Interlace | 2 | 2 | 2 | 2 |
| 09 | Max. Scan Address | 7 | 7 | 1 | 1 |
| 0A | Cursor Start | 6 | 6 | 6 | 6 |
| 0B | Cursor End | 7 | 7 | 7 | 7 |
| 0C | Start Address High | 0 | 0 | 0 | 0 |
| 0D | Start Address Low | 0 | 0 | 0 | 0 |
| 0E | Cursor Location High | 0 | 0 | 0 | 0 |
| 0F | Cursor Location Low | 0 | 0 | 0 | 0 |

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.

Note that the values listed in the above two tables differ from the industry standard values used for other CGA devices. The user need not make special consideration when loading the CRTC but use the standard values as described in section 2.3.20. The auto switch NMI routine will adjust the CRTC values for the mode of operation as required.

#### 1.11.5.5 Set and Clear Light Pen Latch

A method of setting and clearing the light pen latch is provided as part of the IDA. Any read or write I/O access to port 03DBh clears the light pen latch and any read or write I/O to port 03DCh sets the light pen latch. The data read or written is ignored.

#### 1.11.5.6 Operation Control / Status Register

This register in a PPC unique register used to control and monitor the operational mode of the VDU/LCD gate array. It is an input/output port located at I/O address 3DEh.

The layout of the Operation Control/Status Register is as follows:

| Bit | Output Use | Input Use |
|---|---|---|
| 7 | Enable Auto-Switch NMI | NMI Enable Readback |
| 6 | CRTC Prohibit | CRTC Prohibit Readback |
| 5 | Not Used | PPC SW3 |
| 4 | Not Used | PPC SW2 |
| 3 | Not Used | PPC SW1 |
| 2 | Switch Out | Switch Out Readback |
| 1 | MDA/CGA Select | MDA/CGA Readback |
| 0 | VDU/LCD Select | VDU/LCD Readback |

Bit 0 (output) controls the selection of the video output. When bit 0 is set to "1" then video output is to the connector at the rear of the PPC. When Bit 0 is reset to "0" then the internal LCD is selected. Readback returns the information as written.

Bit 1 (output) controls the selection of the display type. When bit 1 is set to "1" then the PPC is setup as a monochrome device and video output, the display buffer, CRTC I/O addresses, and control registers are configured for monochrome. When bit 1 is reset to "0" then the PPC is configured as a colour device and the video output, display buffer, and I/O addresses are set for colour compatibility. Readback returns the information as written.

Bit 2 (output) controls whether the Internal Display Adapter is switched in or out. When bit 2 is set to "1" then the IDA video RAM cannot be read or written and IDA internal registers can be written but not read. The display will continue to be active. An external display adapter may be fitted and there will be no conflicts in either display RAM or display I/O addresses. When bit 2 is reset to "0" then the IDA is present and as configured by bits 0 and 1. In this case an external display adapter may be present but must not be in the same (colour/monochrome) mode else I/O address and regeneration buffer conflicts will occur. Readback returns the information as written.

Bit 3 is not used for output. Upon input Bit 3 returns the state of configuration switch 1 and returns "1" when the switch is ON and "0" when the switch is OFF.

Bit 4 is not used for output. Upon input Bit 4 returns the state of configuration switch 2 and returns "1" when the switch is ON and "0" when the switch is OFF.

Bit 5 is not used for output. Upon input Bit 5 returns the state of configuration switch 3 and returns "1" when the switch is ON and "0" when the switch is OFF.

Note that in practice the ROS uses configuration switches 1, 2 & 3 to set bits 0, 1 & 2 of the Operation Control Port so that the inputs on bits 3 through 5 are closely associated with the functions of bits 0 through 2.

Bit 6 (output) controls whether or not accesses to the CRTC will be prohibited and trapped by the auto switch NMI. Setting bit 6 to "1" allows writes to CRTC registers R0 - R11 to take place. Resetting bit 6 to "0" prohibits CPU access to the CRTC registers as part of the auto-switch mechanism. Readback returns the information as written.

Bit 7 (output) controls whether the auto-switch NMI is enabled or not. When bit 7 is set to "1" then a NMI will be generated when the CPU writes to a prohibited CRTC register. When bit 7 is "0" the NMI feature is turned off. Readback returns the information as written. Note that the NMI Mask Control register (section 1.6.3 at I/O address 0A0h) masks this and all other NMI inputs into the CPU.

#### 1.11.5.7 CRTC Index Readback

The CRTC Index Readback is a read only register located at I/O address 3DDh. It is used by the auto-switch NMI process to determine which CRTC register was last addressed. In addition the MS bits contain additional information about the NMI process.

The layout of the CRTC Index Readback register is as follows:

| Bit | Input Use |
|---|---|
| 7 | CGA Control Port Written. |
| 6 | Operation Control Port Written |
| 5 | Prohibited CRTC register written. |
| 4 - 0 | CGA Control Port Written. |

Bits 0 - 4 contain the last value written to the CRTC Index register.

Bit 5 is "1" when NMI was caused by an attempted write to a prohibited CRTC register.

Bit 6 is "1" when NMI was caused by a write to the Operation Control port.

Bit 7 is "1" when NMI was caused by a write to the CGA Control port.

Bits 5 - 7 are reset by the trailing edge of a read of the CRTC Index Readback port and are set when a NMI is generated by the VDU/LCD gate array.

#### 1.11.5.8 CRTC Data Readback

The CRTC Data Readback is a read only register located at I/O address 3DFh. It is used by the auto-switch NMI process to determine the value written to the last indexed CRTC register. Upon read of I/O address 3DFh the 8-bit returned value is the value as written.

## 1.11.6 Monochrome Mode Registers

When the PPC is in monochrome mode it supports the monochrome text software environment.

#### 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 | Hi-res mode (no effect - always set) |

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 0 on the standard MDA is Hi-res mode but since this bit must be set to achieve anything sensible it is redundant and wired permanently set.

#### 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). |

| Bit | Input Use |
|---|---|
| 0 | Horizontal Sync. |

Bit 0 = "0" when the Video signal is enabled. Bit 0 = "1" when the horizontal retrace is active.

Bit 3 = "1" 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 Monochrome CRTC Initialization.**

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. Note that this set of I/O addresses is partially decoded such that all even I/O addreses between 3B0 - 3B6 access the CRTC index register and all odd I/O addresses between 3B1 - 3B7 access the selected CRTC data register.

The CRTC registers names and initial values are as follows:

| Register Number (Hex) | Register Name | Text 80-Col VDU (Decimal) | Text 80-Col LCD (Decimal) |
|---|---|---|---|
| 00 | Horizontal Total (-1) | 97 | 83 |
| 01 | Horizontal Displayed | 80 | 80 |
| 02 | Horiz. Sync Position | 83 | 81 |
| 03 | Horiz. Sync Width | 15 | 1 |
| 04 | Vertical Total (-1) | 25 | 26 |
| 05 | Vertical Total Adjust | 6 | 0 |
| 06 | Vertical Displayed | 25 | 26 |
| 07 | Vertical Sync Position | 25 | 25 |
| 08 | Interlace | 2 | 2 |
| 09 | Maximum Scan | 13 | 7 |
| 0A | Cursor Start | 11 | 6 |
| 0B | Cursor End | 12 | 7 |
| 0C | Start Address High | 0 | 0 |
| 0D | Start Address Low | 0 | 0 |

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

Note that the values listed in the above table differ from the industry standard for other monochrome devices. The user need not make special consideration when loading the CRTC but use the standard values as described in section 2.3.20. The auto switch NMI will adjust the CRTC values for the mode of operation as required.

## 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 03F2h. 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 μs 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 have 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).

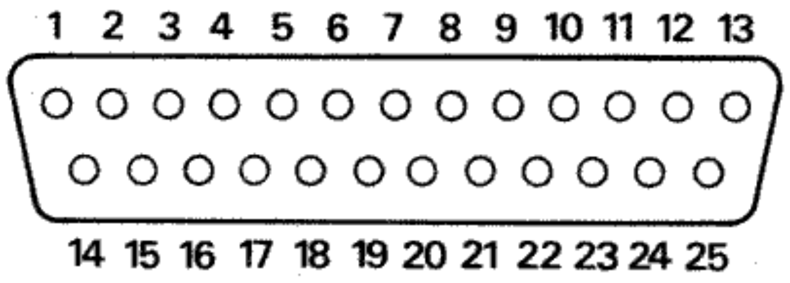
The electrical levels of signal lines on this interface conform with EIA (Electronics Industry Association) standard RS-232C (and the equivalent V.24 interface standard).

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.



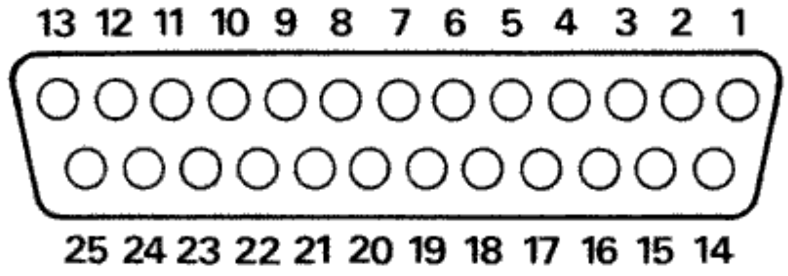
See Appendix 3 for additional details of serial signals and cable connections.

## 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 PPC.

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 4 contains the Amstrad PL-2 printer lead specification.

## 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 µs 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 µs 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). A number of keys (such as the dedicated keypad keys) send an extended keycode sequence which identify them uniquely as dedicated keys rather than the numeric keypad set. The typical extended keycode sequence consists of an extended keycode identifier, 224 (E0h), followed by the keycode.

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 PPC 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 2.3.4).

## 1.16 Modem Interface

The modem serial port is configured to I/O addresses 2F8h - 2FFh (COM2). It is based on the National INS8250 ACE (or UART), single channel device.

The clock frequency 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 IRQ3 input.

The Modem (Hayes) AT commadn interface and register set details are documented in the Amstrad PPC User Instructions.

### 1.16.1 Modem Connector.

The modem connector is a BT type 431A connector located in a small covered compartment to the right of the LCD. A telephone extension socket consisting of a BT type 603A socket is provided at the right hand rear of the case. When the modem is not in use the connection is straight through so that the telephone can be used in standard voice mode.

The modem connector pin assignments are as follows:

| Pin | Assignment |
|---|---|
| 1 | Not Connected. |
| 2 | Line A |
| 3 | Local Earth. |
| 4 | Dialing Shunt. |
| 5 | Line B. |
| 6 | Not Connected. |

Note that pins 3 and 4 are not connected to the modem.

The USA version modem does not use the extension lead arrangement as with the UK modem. Instead the USA modem has two BELL telephone connectors at the rear of the mdoem labelled LINE and PHONE. It is intended that the user supply a telephone extension lead and that the extension lead be connected between the wall connector and the modem LINE connector and the telephone which was connected to the wall socket be connected to the modem PHONE connection. The US telephone wiring uses a two wire circuit and the inner two pins of the BELL connector are the line A & B connections.

## 1.17 Light Pen Connector

The AMSTRAD PPC Supports a standard light pen interface via the 6845 CRTC. The Light Pen connector is located inside the PPC case on the left hand edge of the lower main board about 3 inches forward of the configuration DIP switches. It consisists of a 6-way berg strip and is labeled LIGHT PEN in large letters. Pin 1 is the forward most pin viewed from in front of the machine.

The pin assignment is as follows:

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

| Pin | Assignment |
|---|---|
| 6 | +12 Volts DC. |

## 1.18 Expansion Card Interface

At the rear of the PPC there are two large 'D' socket connectors labelled 'Expansion A-B' and these allow the connection of external devices to the expansion bus.

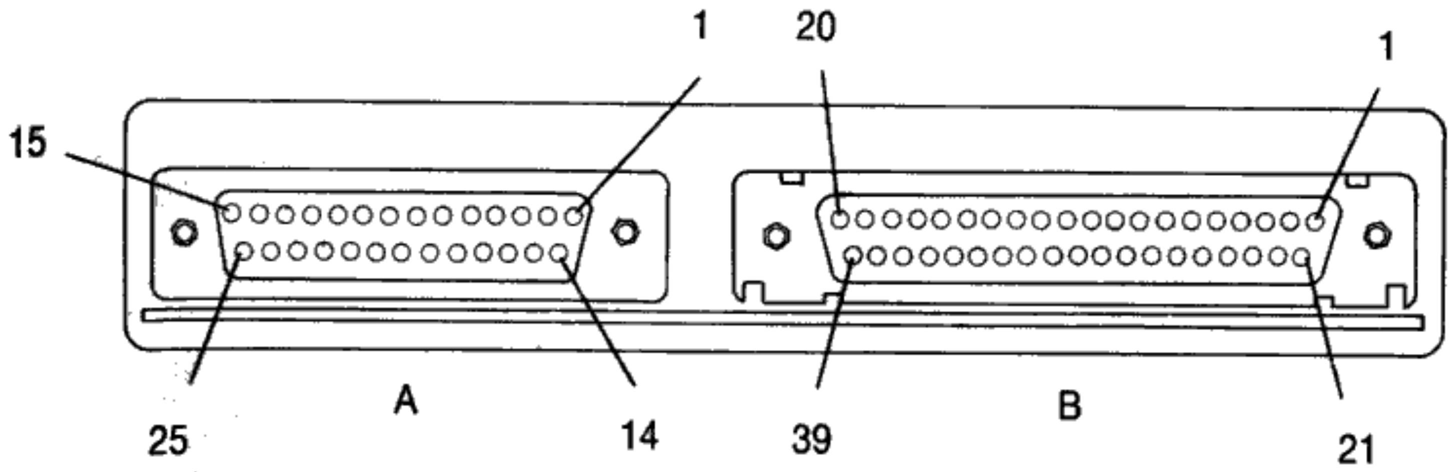
Expansion Connector A is a 25-way D-socket containing the following signals:

| Pin | Signal | In/Out |
|---|---|---|
| 1 | +5 Volts DC | In |
| 2 | T/C | Out |
| 3 | I/O & Memory Address Bit A19 | Out |
| 4 | I/O & Memory Address Bit A17 | Out |
| 5 | I/O & Memory Address Bit A15 | Out |
| 6 | I/O & Memory Address Bit A13 | Out |
| 7 | I/O & Memory Address Bit A11 | Out |
| 8 | I/O & Memory Address Bit A09 | Out |
| 9* | I/O & Memory Address Bit A07 | Out |
| 10* | I/O & Memory Address Bit A05 | Out |
| 11* | I/O & Memory Address Bit A03 | Out |
| 12* | I/O & Memory Address Bit A01 | Out |
| 13 | AEN - Address Enable | Out |
| 14 | Ground | In |
| 15 | -DACK0 | Out |
| 16 | I/O & Memory Address Bit A18 | Out |
| 17 | I/O & Memory Address Bit A16 | Out |
| 18 | I/O & Memory Address Bit A14 | Out |
| 19 | I/O & Memory Address Bit A12 | Out |
| 20 | I/O & Memory Address Bit A10 | Out |
| 21 | I/O & Memory Address Bit A08 | Out |
| 22* | I/O & Memory Address Bit A06 | Out |
| 23* | I/O & Memory Address Bit A04 | Out |
| 24* | I/O & Memory Address Bit A02 | Out |
| 25* | I/O & Memory Address Bit A00 | Out |

Expansion Connector B is a 37-way D-socket containing the following signals:

| Pin | Signal | In/Out |
|---|---|---|
| 1 | -20 Volts DC | -- |
| 2 | IRQ2 | In |
| 3 | IRQ4 | In |
| 4 | IRQ6 | In |
| 5 | I/O RDY | In |
| 6 | -DACK2 | Out |
| 7 | -I/O CHCK | In |

| Pin | Signal | In/Out |
|---|---|---|
| 8 | DREQ2 | In |
| 9* | CK14 (OSC) | Out |
| 10* | -MEMR (Memory Read) | Out |
| 11* | -IOR (I/O Read) | In/Out |
| 12* | ALE | Out |
| 13** | I/O & Memory Data Bit D7 | In/Out |
| 14** | I/O & Memory Data Bit D5 | In/Out |
| 15** | I/O & Memory Data Bit D3 | In/Out |
| 16** | I/O & Memory Data Bit D1 | In/Out |
| 17 | - 5 Volts DC | In |
| 18 | - 12 Volts DC | In |
| 19 | Ground | In |
| 20 | External Power (+12V) | In |
| 21 | IRQ3 | In |
| 22 | IRQ5 | In |
| 23 | IRQ7 | In |
| 24 | -DACK1 | Out |
| 25 | -DACK3 | Out |
| 26 | DREQ1 | In |
| 27 | DREQ3 | In |
| 28* | -MEMW (Memory Write) | Out |
| 29* | -IOW (I/O Write) | In/Out |
| 30* | RESET | Out |
| 31* | CK4 (CLOCK) | Out |
| 32** | I/O & Memory Data Bit D6 | In/Out |
| 33** | I/O & Memory Data Bit D4 | In/Out |
| 34** | I/O & Memory Data Bit D2 | In/Out |
| 35** | I/O & Memory Data Bit D0 | In/Out |
| 36 | + 12 Volts DC | In |
| 37 | + 5 Volts DC | In |



Expansion A-B

Erratum: The illustration shows sockets with too many pins. See annotated image.

Note that the power pins (+5v-20V, -5V-12V, +12V) are listed as inputs, this means the power is to be supplied to, rather than drawn from, these parts.

The signals marked with an asterisk (*) after the pin number are generated directly from the internal gate arrays and should be buffered using a suitable non-inverting buffer (74HC244). There are a total of 16 such signals.

The signals marked with two asterisks (**) after the pin number are the expansion data bus and are at cmos levels. They should be pulled up to TTL levels with 10K ohm resistors.

All remaining signals are TTL compatible and can support a maximum loading of six low-power schottky (LSTTL) loads.

When an expansion unit is fitted, the PPC must be powered via the expansion connectors. PPC power requirements are:

+12V ±5% 50mA
+5V ±5% 2.5A
-5V ±5% 50mA
-12V ±5% 50mA
-20V ±5% 30mA

In addition, the input signal External Power (pin B20) must be connected to +12 Volts to disable the interal switching regulator. Other sources of power (monitor, battery, AC adapter) should be disconnected.

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

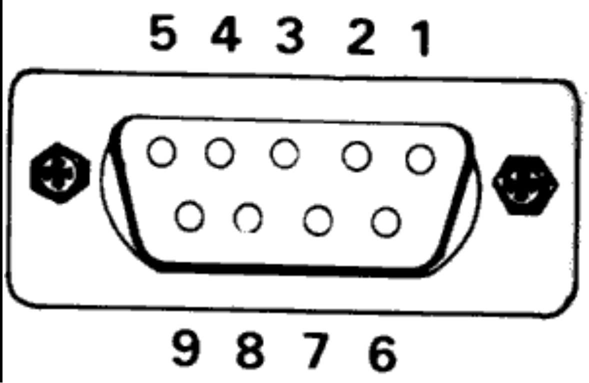
Additional engineering fetails of PPC circuitry can be obtained from the Amstrad PPC Service Manual which is availale as advised by Amstrad Customer Services.

**Note: The USA version of the PPC has the expansion connectors covered by a piece of metal to satisfy FCC requirements.**

## 1.19 Video Connector

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

| Pin | Assignment |
|---|---|
| 1 | Ground |
| 2 | Ground |
| 3 | Red (R) |
| 4 | Green (G) |
| 5 | Blue (B) |
| 6 | Intensity (I) |
| 7 | Mono Video (V) |
| 8 | Horizontal SYNC |



| Pin | Assignment |
|---|---|
| 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 selected video output mode and will be the RGBI version for colour mode or just Mono video and intensity when in monochrome mode.

## 1.20 Power Connectors

There are two sets of power connectors at the rear of the PPC, a small connector for 12 volt DC from the car cigarette lighter adapter or an AC adapter unit and a large 14-way Din connector for external power supplied by a standard Amstrad PC1640 MD, CD or ECD display.

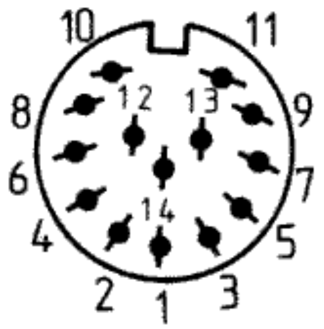
### 1.20.1 The Adapter Power Connector

The adapter power connector is to be used with either the car cigarette lighter adapter or the AC adapter unit, both of which are supplied with the PPC. The outer (sleeve) connection of the adapter power connector is ground and the centre pin in rated at +12V DC nominal ranging from 9.5 volts minimum to 17.5 volts maximum. The AC adapter output rating is 1.9 amps at 13 volts. The supply voltage for the USA/Canadian AC adapter (PPC640AC-A) must be 120 volts AC +/- 10%. The supply voltage for the UK AC adapter (PPC640AC-B) must be 240 volts AC +/- 10%. The supply voltage for the European AC adapter (PPC640AC-E) must be 220 volts AC +/- 10%. Only an Amstrad supplied AC adapter should be used with the PPC. The car cigarette lighter adapter can only be used in cars with a 12 volt, negative earth electrical system.

### 1.20.2 The Display Power Connector

The Display 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 PPC 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

## 1.21 PPC Switch Settings.

The PPC configuration switches are located on the left hand side of the PPC and control the Internal Display Adapter initial mode and the Default display mode to be used by the ROS and MS-DOS.

The Layout of the switches is as follows:

| Switch | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| Name | LCD/VDU | MDA/CGA | On/Off | DDM0 | DDM1 |

Switch 1 when ON selects that video be directed to the external video connector at the rear of the PPC. When switch 1 is OFF all display will be directeod to the internal LCD display unit.

Switch 2 when ON selects that the Internal Display Adapter operates as a monochrome device. When switch 2 is OFF then the IDA will operate as a colour device.

Switch 3 when ON turns the IDA off such that it is not perceived in the software environment. In this case an external display adapter must be fitted. When switch 3 is OFF then the IDA is switched on and it will operate as specified by switches 1 and 2.

Switches 4 and 5 are used by the ROS when setting up the default display mode (See section 1.8.2). The followung table gives the interpretation of the four settings.

| Switch 4 | Switch 5 | DDM Setting |
|---|---|---|
| OFF | OFF | External EGA installed |
| OFF | ON | CGA in 40 Column Mode |
| ON | OFF | CGA in 80 Column Mode |
| ON | ON | Monochrome Mode |

Note that the settings of switches 4 and 5 must be consistent with the IDA control switch 2. If switches 4 and 5 say Monochrome then switch 2 must either be ON or you have an external MDA or Hercules adapter installed in the expansion box. If neither is true the screen will go blank just after ROS initialization (Please Wait....). The same holds true that if switches 4 and 5 say CGA then switch 2 should be OFF or there's a colour display (CGA) adapter in the expansion box else the screen goes blank after ROS initialization.

Contents Index Section 2

# 2.0 Firmware

This section describes the Amstrad PPC Resident Operating System (ROS). It defines the interfaces to all the interrupt service routines provided by the Amstrad PPC 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 PPC 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 firmware should be made through the software interrupts disclosed in this manual. Application programs should 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 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 ROS. 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.

   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. Check the RTC.**

If the RTC registers are incorrect then it they are loaded with default values.

**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 3.

**6. Initialise the 8259 Programmable Interrupt Controller.**

Disable (mask) all interrupt levels. Note that levels 0, 1 and 6 are enabled (unmasked) later.

**7. Initialise the Write Status Registers.**

Write Status-1 is initialised with the number of drives fitted and the default Video Mode. This information is determined by checking if a second floppy drive is fitted to the FDC and by reading PPC switches 4 and 5. 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 | Reserved | Software |
| 22 | Keyboard I/O | Software |
| 23 | Printer I/O | Software |
| 24 | System Restart | Software |
| 25 | Disk Bootstrap | Software |
| 26 | System Clock and RTC I/O | Software |

| Interrupt | Purpose | Type |
|---|---|---|
| 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 2.2.13).

**12. 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.

**13. Display the ROS sign-on message.**

During power-up the ROS checks the RTC. After the sign-on message has been displayed, the ROS checks if the date is reasonable, that is, other than 1980. If so then the last startup and current date are considered valid and the time and date of last switch-on are displayed. If a date of 1980 is detected it is assumed that time and date are invalid and warning messages to "Please Set Time and Date" and "Fit new batteries" are output.

**14. 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.

**15. 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.

**17. System Clock initialization.**

After all tests have been run the ROS sets up the 32-bit system clock location in RAM (at 046Ch) by reading the RTC time and converts this to the number of system clock interrupts since midnight. (See section 2.3.3).

**18. 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 by the IDA to the LCD (regardless of the configuration switches), 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. System RAM test.
9. Programmable Interrupt Controller (8259) test.
10. Disk test.
11. 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 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.

### 2.2.11 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.

### 2.2.12 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.

### 2.2.13 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 ROS 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 happen.

### 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 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.

### 2.3.4 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. In the case of the new 101 / 102 key keyboard there are quite a few extra keys. There are extra [ALT] and [CTRL] keys as well as a dedicated cursor keypad. These keys basically return the exact same keycode as the originals except that preceding the keycode an extra one is sent to signify that the key pressed is one of the new keys.

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 Name | 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 |

| Key Code | Key Name | Normal | Num Lock | ALT | CTRL | SHIFT |
|---|---|---|---|---|---|---|
| 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 |
| 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 | CR | 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 |

| Key Code | Key Name | Normal | Num Lock | ALT | CTRL | SHIFT |
|---|---|---|---|---|---|---|
| 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 | 3920 | 3920 | 3920 | 3920 | N/A |
| * 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 |  | 4737 |
| 48 | Key Pad 8 | 4800 | Ignored | 8D00 |  | 4838 |
| 49 | Key Pad 9 | 4900 | Ignored | 8400 |  | 4939 |
| 4A | Key Pad - | 4A2D | Ignored | 8E00 |  | 4A2D |
| 4B | Key Pad 4 | 4B00 | Ignored | 7300 |  | 4B34 |
| 4C | Key Pad 5 | Ignored | Ignored | 8F00 |  | 4C35 |
| 4D | Key Pad 6 | 4D00 | Ignored | 7400 | Note 1 | 4D36 |
| 4E | Key Pad + | 4E2B | Ignored | 9000 |  | 4E2B |
| 4F | Key Pad 1 | 4F00 | Ignored | 7500 |  | 4F31 |
| 50 | Key Pad 2 | 5000 | Ignored | 9100 |  | 5032 |
| 51 | Key Pad 3 | 5100 | Ignored | 7600 |  | 5133 |
| * 52 | Key Pad 0 | 5200 | Ignored | Ignored |  | 5230 |
| 53 | Key Pad . | 5300 | Ignored | Ignored |  | 532E |
| 54 | Alt PrtScr | Ignored | Sys Req | Ignored | Ignored | N/A |
| 55 | Undefined | Ignored | Ignored | Ignored | Ignored | Ignored |
| + 56 | \ and \| | 565C | Ignored | Ignored | 567C | N/A |
| 57 | F11 | 8500 | 8B00 | 8900 | 8700 | N/A |
| 58 | F12 | 8600 | 8C00 | 8A00 | 8800 | N/A |
| 59 - 7F | Undefined | Ignored | Ignored | Ignored | Ignored | Ignored |
| Extra Keys and their tokens |  |  |  |  |  |  |
| E0,1C | Enter | E00D | A600 | E00A | E00D | N/A |
| E0,1D | Right CTRL | Ignored | Ignored | Ignored | Ignored | Ignored |
| E0,35 | Key Pad / | E02F | A400 | 9500 | E02F | N/A |
| E0,38 | Right ALT | Ignored | Ignored | Ignored | Ignored | N/A |

The following set of keys are the dedicated cursor keys. They produce different sequences of key codes depending on state of the [SHIFT] and [NUM LOCK] keys. In the base and [SHIFT + NUM LOCK] state the key sequence consists of the keycode expected from the corresponding key on the numeric keypad (with the [NUM LOCK] off) preceded by the extra key keycode (ie. E0h). With only [NUM LOCK] on the key sequence consists of the base sequence preceded by E0h,2Ah so there now are four keycodes in the sequence. The last variation is if the [SHIFT] key only is down. In this case the sequence as shown in the base case is preceded by E0h,AAh and again there are four keycodes in the sequence. All sequences of keycodes are treated in the same way by the ROS.

| | | | | | | |
|---|---|---|---|---|---|---|
| E0,46 | Prt Scrn | Ignored | Ignored | Ignored | Ignored | N/A |
| E0,47 | Home | 47E0 | 9700 | 77E0 | 47E0 | N/A |
| E0,48 | ↑ | 48E0 | 9800 | 8DE0 | 48E0 | N/A |
| E0,49 | Pg Up | 49E0 | 9900 | 77E0 | 49E0 | N/A |
| E0,4B | ← | 4BE0 | 9B00 | 73E0 | 4BE0 | N/A |
| E0,4D | → | 4DE0 | 9D00 | 74E0 | 4DE0 | N/A |
| E0,4F | End | 4FE0 | 9F00 | 75E0 | 4DE0 | N/A |
| E0,50 | → | 50E0 | A000 | 91E0 | 50E0 | N/A |
| E0,51 | Pg Up | 51E0 | A100 | 76E0 | 51E0 | N/A |
| E0,52 | Ins | 52E0 | A200 | 92E0 | 52E0 | N/A |
| E0,53 | Del | 53E0 | A300 | 93E0 | 53E0 | N/A |

The last set of keys are classed as miscellaneous.

E0,37
    CTRL or SHIFT Print Screen
E0,2A,E0,37
    Print Screen
E0,AA,E0,35
    Shift / on keypad
E1,1D,45
    Pause

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

The key code marked '+' is handled by ROS but cannot be produced by the US (101 key) version of the keyboard.

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.

Note 1: The Numeric keypad keys will produce a different token depending on the state of the [Num Lock] and [Shift] keys. With [Num Lock] on and [Shift] down then Num Lock is cancelled and the Normal column tokens are produced. With [Num Lock] off and [Shift] down then the tokens in the Num Lock column are produced.

Note 2: Not all tokens listed in this table will be returned by the non-extended Get Key Token software interrupt (Int 22); some are discarded and some are modified when returned to the calling software. Refer to section 2.3.11 for complete details of the rules used when dealing with the extended keycodes produced by the extended keyboard.

**2.3.4.1 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. [Pause].
   The ROS waits for another key to be pressed (except [PAUSE]), thus suspending any application that is running.
3. [Ctrl]+[Pause]: 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. [PrtScrn]: 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 install a null pointer here. Typing [Ctrl+PrtScrn] invokes printer echo mode whereby all keystrokes are echoed to the printer. This mode can be toggled off by the same keystroke combination. Printer echo mode can also be toggled using [Ctrl+P].
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. Note that pressing one of the keypad keys while a shift key is down inverts the state of the numlock and the token inserted into the buffer will be the opposite to that which Num Lock indicates. (ie. Num Lock light off + [Shift] gives one of the keypad numeric keys).
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. [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 programs (such as KEYB) 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.

## 2.3.5 Interrupt 14: Floppy Disk Controller (0Eh).

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.

## 2.3.6 Interrupt 16: Video I/O (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 2.3.20). It also provides facilities for scrolling the screen up or down, reading and writing pixels (graphics only) and reading the light pen.

CPU registers for 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 (0Ah).
- 11 - Write Colour Select Register (0Bh).
- 12 - Write Pixel (0Ch).
- 13 - Read Pixel (0Dh).
- 14 - Write Character in Teletype Emulation mode (0Eh).
- 15 - Get Video Parameters (0Fh).

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.

**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 28 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 colours.
7 - Alpha 25 Rows by 80 Columns Monochrome.

Exit:
All flags and registers preserved.

In mode 4 palette 0 may be selected by writing the colour select register using Video Int 16 Function 11. The definition of the palettes is contained in Section 1.11.2.1, Graphics Mode 1.

When mode 5 is selected it must be followed by a selection of palette zero (Colour 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.

If the MS bit of System RAM variable location 0:412 is set (80h) then when a mode change into text mode is setup the hardware LCD inverse bit will be set in the mode register and if the display is the internal LCD then the character output will be inverted as described in section 1.11.1.

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 Video mode select register.
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 (07 - white). The graphics mode fill is zeros.

6. If colour display then set up the Colour Select register (3D9h):

   Set the border colour to the default background colour.
   In graphics modes set intensified foreground colours.
   In mode 6 (Graphics 640 Mode) set white foreground colour.

7. For text modes select page zero.
8. Set the cursor size to start cursor from the video parameter table.
9. Enable video output.

**Int 16 Function 1: Set Cursor Size.**

This function is only relevant in alpha modes as the hardware 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 63.
- CL = Ending scan of cursor in range 0 to 31.

Exit:
- All flags and registers preserved.

Note that values greater than 31 turn the cursor off. This is because bit 5 in the start register is the 6845 cursor off bit.

**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 section 1.11.4 (BIOS Modes) for valid page numbers and page starting address details.

**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 section 1.11.4 for valid page numbers and page starting address details.

**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.

**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 section 1.11.4 for valid page numbers and page starting address details.

**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 colour.

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.

**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.

**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 2.3.20 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.

**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 2.3.20 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 colour (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.

**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 2.3.20 for additional details.

CPU registers are used as follows:

Entry:
> AH = 10 (0Ah).
> 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 colour (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.

**Int 16 Function 11: Write Colour Select Register.**

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

CPU registers are used as follows:

Entry:
> AH = 11 (0Bh).
> 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.

**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 (0Ch).
- 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 Colour for 320 graphics modes.

Exit:
- All flags and registers preserved.

The pixel colour 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).

**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 (0 to 199)
- CX = Pixel Column (0 to 639)

Exit:
- AL = Colour of the specified pixel.
- All flags and other registers preserved.

**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 (0Eh).
- 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 colour (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.

**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 (0Fh).

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.

## 2.3.7 Interrupt 17: System Configuration (11h).

This software interrupt returns the current system configuration status ad defined in RAM locations 0:410 and 0:411 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. |
| 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.

### 2.3.8 Interrupt 18: Memory Size (12h).

This software interrupt returns the system RAM size as held in system locations 0:413 and 0:414 hex.

CPU registers are used as follows:

Entry:
    No conditions.
Exit:
    AX = Number of 1K memory blocks fitted.
    All flags and other registers preserved.

### 2.3.9 Interrupt 19: Disk I/O (13h).

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.
- 8 - Read Drive parameters.
- 21 - Read Drive Type (15h).
- 22 - Read Disk Change Status (16h).

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 (10h).
- 32 - Floppy disk controller error (20h).
- 64 - Seek error (40h).
- 128 - Floppy disk controller timeout (Drive Not Ready) (80h).

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.

For all disk functions except 0 & 1 drive number (DL) is checked and if greater than maximum drive number the function is rejected and return status is set to 1 (AH=1) and carry is set.

Media changed error status is only applicable to operations which transfer information to and from the magnetic media. It is derived using the following procedure: Before performing the I/O operation the selected disk drive's change line status is checked. If false then the I/O operation carries on as normal. If, however the change line is active then the ROS attempts to clear the change line by issuing a step pulse. If the change status clears then there is media installed and the door is closed and the I/O operation terminates by reporting that change line was detected. (AH = 06). If however the change line remains true then either the door is open or there's no diskette installed (or both) and in this case a timeout error (AH = 128) is returned.

**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 disk 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 in 2.3.9.
- 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 2.3.20).

**Disk Function 8: Read Drive Parameters**

This function returns the floppy disk drive parameters.

CPU registers are used as follows:

Entry:
  AH = 8.
  DL = Drive Number (0 or 1).
Exit:
  AX = 0
  DH = Number of Heads - 1. (1)
  DL = Number of Drives installed. (1 or 2)
  CH = Number of Tracks - 1. (79)
  CL = Number of Sectors per Track. (9)
  ES:DI = 20-bit pointer to Disk Parameter Table for the drive.
  Flags corrupt and CY clear.
  All other registers preserved.

**Disk Function 21: Read Drive Type**

This function returns the floppy disk drive type.

CPU registers are used as follows:

Entry:
  AH = 21 (15h).
  DL = Drive Number (0 or 1).
Exit:
  AH = 2 (Diskette with change line.)
  Flags corrupt and CY clear.
  All other registers preserved.

**Disk Function 22: Read Disk Changeline.**

This function returns the drive changeline status.

CPU registers are used as follows:

Entry:
    AH = 22 (16h).
    DL = Drive Number (0 or 1).
Exit:
    AH = 0 if change line not active & CY clear.
    6 if change line active & CY set.
    Flags corrupt and CY as above
    All other registers preserved.

Note that this function does not clear the change line signal. An I/O function must be performed to clear change line active.

## 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 (0Bh).
16 - ECC error (10h).
17 - ECC data corrected (11h).
32 - Hard disk controller error (20h).
64 - Seek error (40h).
128 - Disk controller timeout (Drive Not Ready) (80h).
187 - Undefined error occurred (BBh).
255 - Read Status Failure (FFh).

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.

### 2.3.10 Interrupt 20: Serial I/O (14h).

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. |
| 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 VDU 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.

### 2.3.11 Interrupt 22: Keyboard I/O (16h).

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.
        5 - Insert token into keyboard buffer.
        16 - Get extended key token from the keyboard buffer (10h).
        17 - Return extended keyboard Buffer status (11h).
        18 - Return extended control states (12h).
Exit:
    If function selector out of range then
        AH = AH - 18.
    If function within range then
        All flags and registers as specified by function.

**Keyboard I/O Function 0: Get Key Token.**

Return the next non-extended key token from the key token buffer. If no non-extended key token is available then wait until a key token is available. If an extended key token is encountered during the token search it is discarded. The key token search is carried out as follows:

```
If high byte of the token equals 00
    then return the token. (Alt keypad token)
If high byte of the token is greater than 84h
    then discard the token.
If low byte of the token equals F0h
    then discard the token.
If low byte of the token equals E0h
    then set the low byte 00h and return entry.
```

All other entries are returned unchanged except for:

- E00Dh (Key Pad - Enter) translated to 1C0Dh (Enter).
- E02Fh (Key Pad - Slash) translated to 352Fh (Slash).

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.

This sub-function uses the same process as sub-function 0 to determine whether there is a non-extended key token in the keyboard buffer. Extended tokens are discarded in the key token search process.

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 0:417h).

CPU registers are used as follows:

Entry:
    AH = 2.
Exit:
    AL = Current shift states:

| Bit(s) | Function (Set if key active) |
|---|---|
| 7 | Ins Toggle |
| 6 | Caps Lock Set |
| 5 | Num Lock Set |
| 4 | Scroll Lock Set |
| 3 | Alt Key Down (either Left or Right) |
| 2 | Ctrl Key Down (either Left or Right) |
| 1 | Left Shift Down |
| 0 | Right Shift Down |

All flags and other registers preserved.

**Keyboard I/O Function 5: Insert token into keyboard buffer.**

Insert 16-bit key token into the keyboard buffer.

CPU registers are used as follows:

Entry:
    AH = 5.
    CX = Token to be inserted into the buffer.
Exit:
    AL = 0: Insertion Successful
    AL = 1: Keyboard Buffer Full - Token discarded.
    All flags and other registers preserved.

**Keyboard I/O Function 16: Get Extended Key Token.**

Return the next key token from the key token buffer, if buffer empty then waits till one is available.

The key token translation process is as follows:

If low byte of the token equals F0h
    then set the low byte 00h and return entry.

All other key tokens are returned unchanged.

CPU registers are used as follows:

Entry:

AH = 16 (10h).

Exit:

AX = Key Token.

All flags and other registers preserved.

**Keyboard I/O Function 17: Return Extended 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 actually removing it.

This sub-function uses the same process as sub-function 16.

CPU registers are used as follows:

```
Entry:
        AH = 17 (11h).
Exit:
        If key token buffer 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 18: Return Extended Shift States.**

Returns two bytes relating to the state of system ram locations 417h, 418h and 496h which contain keyboard status information.

CPU registers are used as follows:

```
Entry:
        AH = 18 (12h).
Exit:
        AX = Current extended shift states:
```

AH Register -

| Bit(s) | Function (Set if key active) |
|---|---|
| 7 | System Request (SysRq) Key |
| 6 | Caps Lock Active |
| 5 | Num Lock Active |
| 4 | Scroll Lock Active |
| 3 | Right Alt Key Down |
| 2 | Left Alt Key Down |
| 1 | Right Ctrl Key Down |

| Bit(s) | Function (Set if key active) |
|---|---|
| 0 | Left Ctrl Key Down |

AL Register -

| Bit(s) | Function (Set if key active) |
|---|---|
| 7 | Ins Toggle |
| 6 | Caps Lock Toggled |
| 5 | Num Lock Toggled |
| 4 | Scroll Lock Toggled |
| 3 | Alt Key Down (either Left or Right) |
| 2 | Ctrl Key Down (either Left or Right) |
| 1 | Left Shift Down |
| 0 | Right Shift Down |

### 2.3.12 Interrupt 23: Printer I/O (17h).

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.

## 2.3.13 Interrupt 24: System Restart (18h).

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 "Insert a SYSTEM disk into Drive A" and "Then press any key." 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.

### 2.3.14 Interrupt 25: Disk Bootstrap (19h).

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 07C00. 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).

CPU registers are used as follows:

Entry:
 No conditions.
Exit:
 To program loaded by Disk Bootstrap.

### 2.3.15 Interrupt 26: System Clock & Real Time Clock (1Ah).

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.

### 2.3.16 Interrupt 27: Keyboard Break Interrupt (1Bh).

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.

### 2.3.17 Interrupt 28: External Ticker Interrupt (1Ch).

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.

### 2.3.18 Interrupt 29: VDU Parameter Table (1Dh).

This interrupt vector location contains the 32-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 6845 CRTC registers (See section 1.11.5). When a new video mode is selected the table entry used for initialisation as follows:

| Table Entry | Video Mode |
|---|---|
| 0 | 0 - Alpha 25 by 40 Chars.<br>1 - Alpha 25 by 40 Chars. |
| 1 | 2 - Alpha 25 by 80 Chars.<br>3 - Alpha 25 by 80 Chars. |
| 2 | 4 - Graphics 320 by 200 Pixels, palettes 0 or 1.<br>5 - Graphics 320 by 200 Pixels, palette 2.<br>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 |

Note that this table reflects industry standard values for CGA and MDA type devices and that at this user level there is no distinction between using the PPC Internal Graphics Adapter and an external (expansion slot based) CGA or MDA. The PPC Auto Switch NMI process will adjust the values loaded into the CRTC to be appropriate to the LCD or external video mode being used. The user can therefore use this table directly without regard to the CRTC initialization values discussed in section 1.11.

### 2.3.19 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 |

### 2.3.20 Interrupt 31: VDU Matrix Table (1Fh).

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.

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 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).<br>Used as stack area only during initialisation. |
| 00400 | Logical Serial Device 0 Base I/O Address (Word).<br>Contains the base address of logical serial device 0.<br>Initally the System Asynchronous Serial port address. |
| 00402 | Logical Serial Device 1 Base I/O Address (Word).<br>Contains the base address of logical serial device 1.<br>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).<br>The base address of logical printer device 0.<br>Initally the System Parallel Printer port. |

| Location(s) | Usage |
|---|---|
| 0040A | Logical Printer Device 1 Base I/O Address (Word).<br>The base address of logical printer device 0.<br>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).<br>Initially points to the external monochrome VDU controller if both the external parallel 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).<br>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. |

| Location(s) | Usage |
|---|---|
| 00412 | Reserved (Byte). |
| 00413 | Total RAM Size (Word).<br>Initially set to the number of 1K User (System) RAM Blocks installed. |
| 00415 | Extra RAM Size (Word).<br>Initially set to the number of 1K User (System) RAM Blocks installed minus 64 |
| 00417 | Key Toggles and Key States (Byte).<br>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 Toggle Set |
| 6 | Caps Lock Active |
| 5 | Num Lock Active |
| 4 | Scroll Lock Active |
| 3 | Alt Key (Left or Right) Active |
| 2 | Ctrl Key (Left or Right) Active |
| 1 | Left Shift Key Down |
| 0 | Right Shift Key Down |

| Location(s) | Usage |
|---|---|
| 00418 | Keys down (Byte).<br>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) |
|---|---|

| Location(s) | Usage |
|---|---|
| | 7 Ins<br>6 Caps Lock<br>5 Num Lock<br>4 Scroll Lock<br>3 Pause<br>2 System Request<br>1 Left Alt<br>0 Left Ctrl |
| 00419 | Absolute Key Token Number (Byte).<br>When an absolute key token numbered is entered via ALT and the numeric key pad, this variable holds the current state of the token. |
| 0041A | Key Token Buffer Out Pointer (Word).<br>This variable holds the absolute offset to the next key token to be removed from the key token buffer.<br>Note that the ROS assumes that the buffer has a segment paragraph address of 0040h. |
| 0041C | Key Token Buffer In Pointer (Word).<br>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. |
| 0041E | Key Token Buffer (Buffer).<br>The Key Token Buffer is a 16 word circular buffer used to store up to 16 key tokens. |
| 0043E | Drive Restore Flag (Byte).<br>Each floppy disk drive has a restore flag associated with it (bit 0 for drive 0 and bit 1 for drive 1).<br>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.<br>Bit 7 is used for handling FDC hardware interrupts. |
| 0043F | Drive Motor Flag (Byte).<br>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. |
| 00440 | Drive Motor Timeout Counter (Byte).<br>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. |
| 00441 | Disk Status (Byte).<br>This byte holds the status returned by the last disk operation. (See section 2.3.11 Disk I/O Interrupt - Function 1.) |
| 00442 | FDC Results/HD Parameter Buffer (Buffer).<br>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. |
| 00449 | Current Video Mode (Byte).<br>The current VDU mode from the last Int 16 setmode call is stored here. |
| 0044A | Visible Display Columns (Word).<br>The number of visible character columns currently being displayed is stored here. |

| Location(s) | Usage |
|---|---|
| 0044C | Video Display Page Size (Word).<br>This word holds the amount of Video RAM used by the ROM BIOS to display one page as defined below:<br><table><tr><th>Mode(s)</th><th>Size</th></tr><tr><td>0 & 1</td><td>2048</td></tr><tr><td>2 & 3</td><td>4096</td></tr><tr><td>4 - 6</td><td>16384</td></tr><tr><td>7</td><td>4096</td></tr><tr><td>13</td><td>8192</td></tr><tr><td>14</td><td>16384</td></tr><tr><td>15 - 16</td><td>32768</td></tr></table> |
| 0044E | Display Page Start Offset (Word).<br>Contains the origin of the currently active video display page. |
| 00450 | Cursor Address Buffer (Buffer)<br>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).<br>This byte contains the current end scan number that was programmed into the CRT controller. |
| 00461 | Cursor Start Scan (Byte).<br>This byte contains the current start scan number that was programmed into the CRT controller. |
| 00462 | Active Display Page (Byte).<br>This byte contains the selected display page number. |
| 00463 | CRTC I/O Address (Word).<br>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).<br>This byte contains the current contents of the Video Mode Control Register. |
| 00466 | Current CGA Colour Select Register (Byte).<br>This byte contains a copy of the data loaded into CGA colour select register. |
| 0467-046B | Reserved |
| 0046C | System Clock (Long Word).<br>The 32 bit system clock count |
| 00470 | 24 Hour Flag (Byte).<br>When the system clock reaches 0001800B0h then it is cleared and this flag byte is set to 0FFh.<br>Note that reading the clock via interrupt 26 clears this flag. |
| 00471 | Break (Byte).<br>This byte is initially set to zero. Each time Break ([Ctrl]+[Nun 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).<br>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 |

| Location(s) | Usage |
|---|---|
| | does not repeat the RAM tests. Setting 1235h prevents full hardware reset. |
| 0474-0477 | Reserved for Hard Disk BIOS ROM. |
| 0478 - 47A | Logical Printer Device 0 - 2 Timeout Count (Buffer).<br>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). |
| 0047B | Reserved. |
| 047C - 047D | Logical Serial Device 0 - 1 Timeout Count (Buffer).<br>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). |
| 0047E | Reserved. |
| 00480 | Key Token Buffer Start Address (Word).<br>Offset pointer to the start of the key token buffer.<br>Note that the assumed buffer segment paragraph address is 0040h. |
| 00482 | Key Token Buffer End Address (Word).<br>Offset pointer to the start of the key token buffer. |
| 00484 | EGA Display Rows (Byte).<br>This location contains the number of character rows (less one) on the display screen. |
| 00485 | EGA Character Points (Byte).<br>This location contains the current character matrix length in bytes. |
| 00487 | EGA Status (Byte).<br>This location is used by an EGA's ROM BIOS to hold its current status information (Colour/Mono, Primary/Sec). |
| 00488 | EGA Switches (Byte).<br>This location contains the current switch settings for EGA control switches 1-4 in bits 0-3 (inverted) and the features switches in the MS bits. |
| 00489 | ROS NMI Flag (Byte).<br>This byte holds the ROS NMI anti-recursion flag. |
| 0048A | ROS NMI Vector (Long Word).<br>This 4-byte location holds the second level NMI vector. |
| 0048E | Low-Res Flag (Byte).<br>The ROS uses this byte to check for low-res mode.<br>Bit 0 = vduHlowres - Horizontal timing may mean low-res mode.<br>Bit 1 = vduVlowres - Vertical timing may mean low-res mode. |
| 0048F | Operation Control Reg (Byte).<br>This byte holds a copy of the operation control register. |
| 00496 | Keyboard type (Byte).<br>This byte holds the extended keyboard information |

| Bit | Meaning |
|---|---|
| 4 | Extended Keyboard Attached |
| 3 | Right Ctrl Down |
| 2 | Left Ctrl Down |
| 1 | Extended Code E0h last received |
| 0 | Extended Code E1h last received |

| Location(s) | Usage |
|---|---|
| 00500 | Print Screen Status (Byte). |

| Value | Meaning |
|---|---|
| 0 | Print Screen completed OK. |
| 1 | Print Screen in progress. |
| 255 | Print Screen abandoned due to timeout. |

## 2.5 Non-Volatile RAM (NVR)

The first 21 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 | Time and Date Checksum. | |
| 21-63 | 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 Time Last Used is checksummed and if the lower byte of the sum is not 0FFh 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 primary display device. Those locations without defaults (marked with '--') are not initialized.

## 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 PPC 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 (VRT) 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 PPC 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 inside the PPC case on the lower board. They are used by the ROS firmware to define the language option or diagnostic mode option as detailed below.

| Link Value | ROS Usage |
|---|---|
| 0 | English Language. |
| 1 | German Language. |
| 2 | French Language. |
| 3 | Spanish Language. |
| 4 | Danish Language. |
| 5 | Swedish Language. |
| 6 | Italian Language. |
| 7 | 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 IDA video RAM and other display adapters (See 1.11 for buffers) |
| C0000 - C3FFF | External EGA ROM BIOS area. |

| Location(s) | Usage |
|---|---|
| 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 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 |
|---|---|
| 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). |

| Bit(s) | Function |
|---|---|
| 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 |

## 3.4 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. |
| 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. |
| 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. |

## 3.5 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. |
| 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 PPC. 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. |

| Bit(s) | Function (Action ... { 1/0 }) |
|---|---|
| 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. |
| 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.

## 3.6 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.

## 3.7 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 | | 0E |
| | 50 Bytes User RAM | |
| 63 | | 3F |

### 3.7.1 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 |

| Address | Function | Range | Binary Data Mode | BCD Data Mode |
|---|---|---|---|---|
| 3 | Min Alarm | 0-59 | 00h-3Bh | 00h-59h |
| 4 | Hours<br>12-Hr Mode | 1-12 | 01h-0Ch (AM)<br>81h-8Ch (PM) | 01h-12h (AM)<br>81h-92h (PM) |
|  | 24-Hr Mode | 0-23 | 00h-17h | 00h-23h |
| 5 | Hrs Alarm<br>12-Hr Mode | 1-12 | 01h-0Ch (AM)<br>81h-8Ch (PM) | 01h-12h (AM)<br>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.

### 3.7.2 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: $IRQF = PF \cdot PIE + AF \cdot AIE + UF \cdot UIE$. Any time the IRQF bit is a "1", the $\overline{IRQ}$ 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.

## 3.8 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 |
|---|---|
| 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). |

| Bit(s) | Function |
|---|---|
| 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. |
| 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 }) |
|---|---|

| 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 |
|---|---|
| 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.

# APPENDIX 1 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.

## 1.1 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).

## 1.2 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.

## 1.3 Country Command

The country command is used to select the country dependent information as shown in appendix 2.

The syntax of the country command is:

```
COUNTRY=country code[,[<code page>][,[drive:]filename]]
```

The country code is shown in Appendix 2. Note that only the parameters in the table are affected by the country command. Other country dependent factors such as the language links, KEYB, and national variant disks, effect the total country dependent environment.

<code page> is the code page for the country (see appendix 2).

| | |
|---|---|
| Filename | is a file containing the country information. If no filename is specified the file COUNTRY.SYS is assumed and it must be available in the root directory. |

Note that the utility, Nlsfunc, can also be used to load country specific information.

## 1.4 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 PPC 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 PPC User Instructions for ANSI escape sequence reference information.

### 1.4.1 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).

### 1.4.2 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 [<disk size> [<sector size> [<entries>3 ]]] [/A]
```

Where:

<disk size> is disk size in kilobytes. Default is 64 and minimum is 16.
<sector size> is the sector size. The values 128, 256, 512, and 1024 are allowed. Default is 128.
<entries> 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.

### 1.4.3 DISPLAY.SYS

DISPLAY.SYS is an installable device driver which supports code page switching for the console device. A code page is an alternate set of 256 characters which can be used in graphics display modes and with EGA display adapters in text modes.

To install display.sys insert a command line of the following form in your config.sys file:

```
DEVICE=[DRIVE:][PATH]DISPLAY.SYS CON[:]=[type[,hwcp[,n,m]]]
```

The CON = parameters are as follows:

type -
The display adapter in use consisting of MONO, CGA, EGA or LCD. Note that LCD is not valid for the Amstrad PPC since this pertains only to the "Industry Standard" PC's LCD hardware.
hwcp -
The code page supported by the hardware as follows:
437 United States
850 Multilingual ** (Not Supported in PPC see LK6, LK7 below)
860 Portugal
863 French-Canadian
865 Norway
n -
The number of additional code pages that can be supported. This number is dependent on the hardware. MONO and CGA do not support other fonts, so n must be 0. For EGA, n can be 2.
m -

The number of sub-fonts that are supported by each code page.

Note that there are a set of font selector links within the PPC which are on the lower (main system) board located near the character generator ROM (IC135), and these links select one of four hardware (hwcp) fonts. The two links, labeled LK6 and LK7, are as follows:

| LK7 | LK6 | Selected Character set | hwcp number |
|---|---|---|---|
| Off | Off | English | 437 |
| Off | On | Norwegian | 865 |
| On | Off | Portugese | 860 |
| On | On | Greek | |

LK6 is the link closest to pin 28 of the character generator ROM (IC135) and LK7 is located adjacent to LK6 and is parallel to the axis of IC135.

Files EGA.CPI and LCD.CPI are additional code page files for use with the MODE command of the form:

```
MODE con: cp prepare=((cplist), drive:cpfile)
```

where:

cplist -
  Is 850 if the hardware code page (hwcp) is 437.
  If the hardware code page is not code page 437, cplist is 850 plus the hardware code page. For example, Portugese is 850, 860.

drive: -
  specifies the drive path where the display code page font file is loaded.

cpfile -
  is the name of the code page font file. Note that for the PPC only EGA.CPI is valid and that LCD.CPI only works on the "Industry Standard" PC's LCD hardware.

Note that this MODE command should be used in the AUTOEXEC.BAT file.

## 1.5 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).

## 1.6 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>

## 1.7 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.

## 1.8 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'.

## 1.9 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.

## 1.10 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 2: COUNTRY DEPENDENT INFORMATION FOR MS-DOS 3

| Country | Num | KbC | DcP | AcP | DtF | DtS | TmS | TmF | CSm | CFt | CSd | ThS | DeS | DIS |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Australia | 061 | US | 437 | 850 | 1 | - | : | 0 | $ | 0 | 2 | , | . | , |
| Belgium | 032 | BE | 437 | 850 | 1 | / | : | 1 | BEF | 2 | 2 | , | , | ; |
| Canada (Eng.) | 001 | US | 437 | 850 | 0 | - | : | 0 | $ | 0 | 2 | , | . | , |
| Canada (Fr.) | 002 | CF | 863 | 850 | 2 | - | : | 1 | $ | 3 | 2 | , | , | ; |
| Denmark | 045 | DK | 865 | 850 | 1 | - | . | 1 | kr | 2 | 2 | . | , | ; |
| Finland | 358 | SU | 437 | 850 | 1 | . | . | 1 | mk | 3 | 2 | , | , | ; |
| France | 033 | FR | 437 | 850 | 1 | / | : | 1 | F | 3 | 2 | , | , | ; |
| Germany | 049 | GR | 437 | 850 | 1 | . | . | 1 | DM | 2 | 2 | . | , | ; |
| Italy | 039 | IT | 437 | 850 | 1 | / | : | 1 | L | 0 | 0 | . | , | ; |
| Israel | 972 | -- | 437 | 850 | 1 | / | : | 1 | ö | 2 | 2 | , | . | , |
| Latin America | 003 | LA | 437 | 850 | 1 | / | : | 1 | $ | 3 | 2 | , | . | ; |
| Middle East | 785 | -- | 437 | 850 | 1 | / | : | 0 | ñ | 3 | 3 | . | , | ; |
| Netherlands | 031 | NL | 437 | 850 | 1 | - | : | 1 | ƒ | 2 | 2 | . | , | ; |
| Norway | 047 | NO | 865 | 850 | 1 | / | . | 1 | Kr | 2 | 2 | . | , | ; |
| Portugal | 351 | PO | 860 | 850 | 1 | / | : | 1 | $ | 4 | 2 | . | , | ; |
| Spain | 034 | SP | 437 | 850 | 1 | / | : | 1 | Pt | 3 | 2 | . | , | ; |
| Sweden | 046 | SV | 437 | 850 | 2 | - | . | 1 | SEK | 2 | 2 | . | , | ; |
| Switzerland (Fr.) | 041 | SF | 437 | 850 | 1 | . | . | 1 | Fr | 2 | 2 | , | . | , |
| Switzerland (Ger.) | 041 | SG | 437 | 850 | 1 | . | . | 1 | Fr | 2 | 2 | , | . | , |
| United Kingdom | 044 | UK | 437 | 850 | 1 | - | : | 1 | £ | 0 | 2 | , | . | , |
| United States | 001 | US | 437 | 850 | 0 | - | : | 0 | $ | 0 | 2 | , | . | , |

Table Columns:

| | |
|---|---|
| Num | = Country Number Code. |
| KbC | = KEYB Code. |
| DcP | = Default Code Page. |
| AcP | = Alternate Code Page. |
| 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. |
| DIS | = | Data List Separator. |

The items in this table are used in conjunction with the COUNTRY command and the Keyboard Utility (KEYB).

# APPENDIX 3: 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

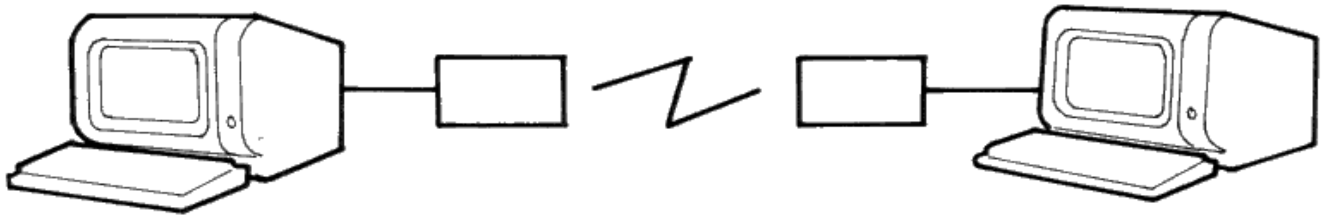


Figure 1.

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.

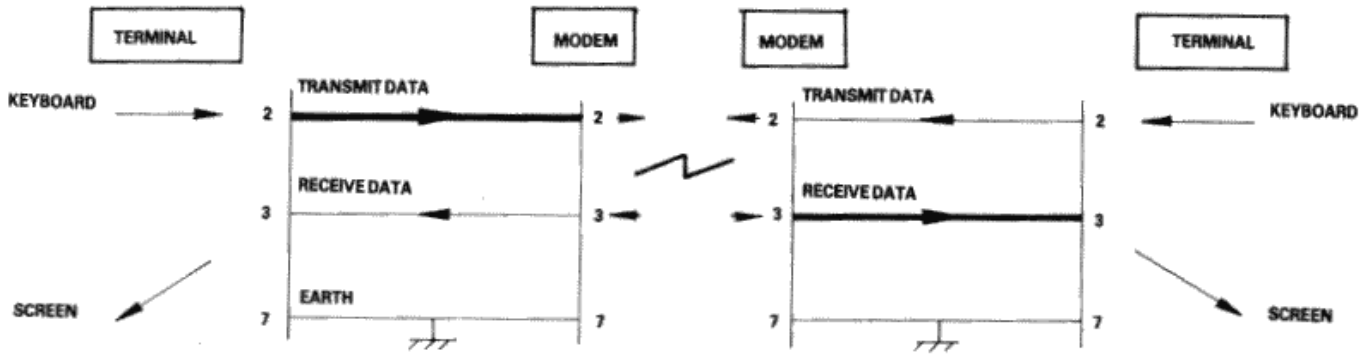


Figure 2.

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.

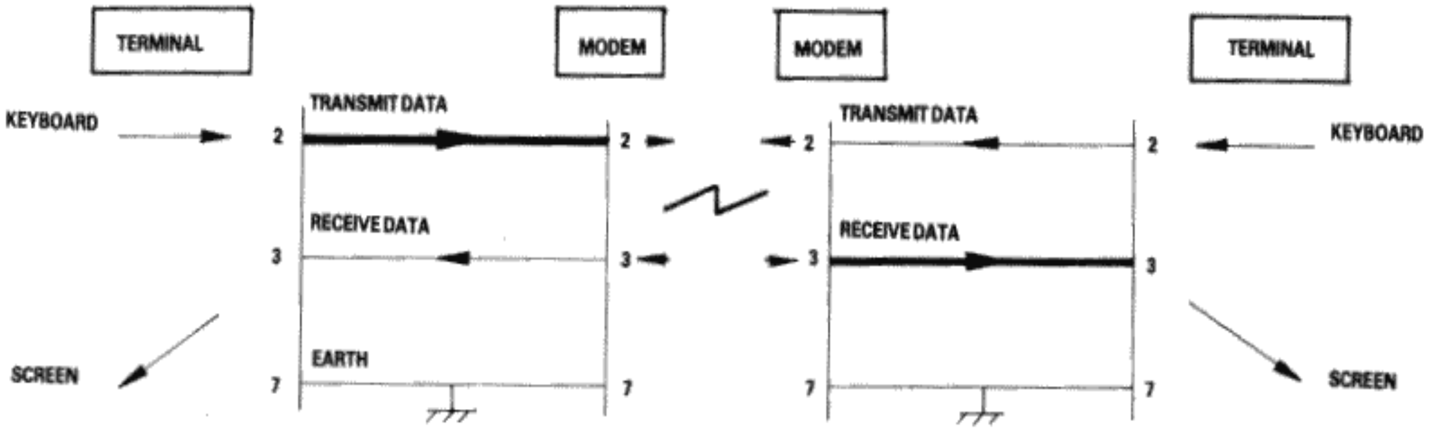


Figure 3

Naturally, the Amstrad PPC 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.

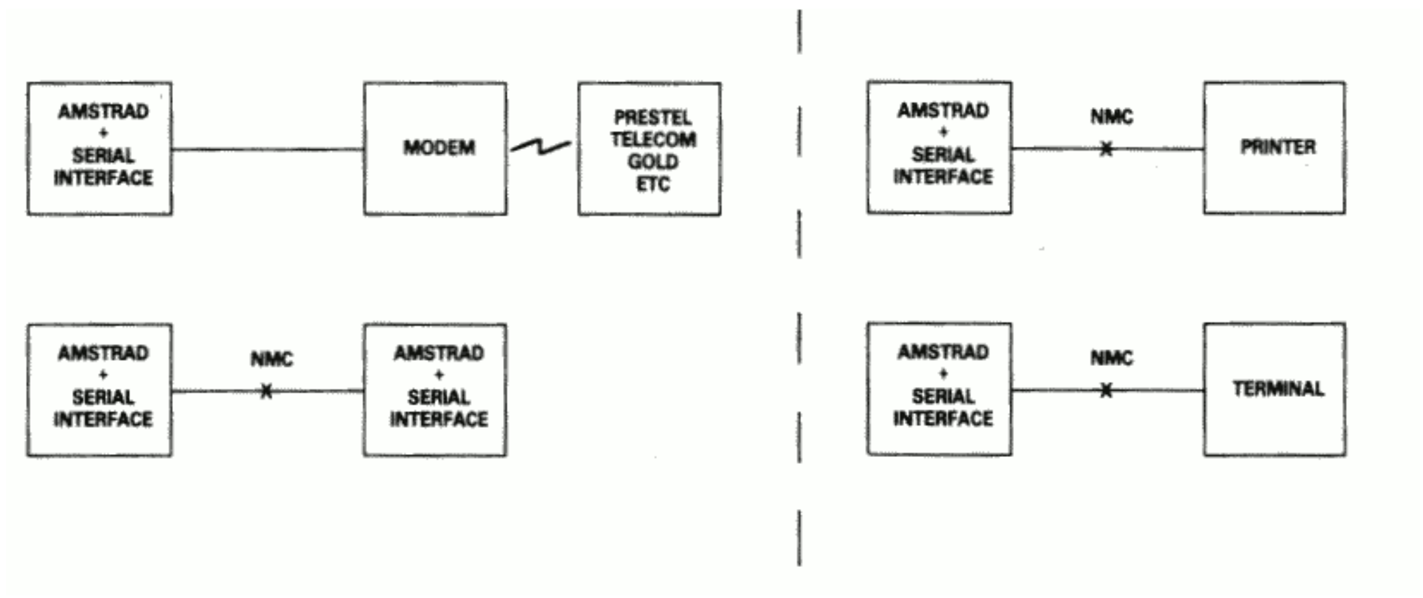

AMSTRAD + SERIAL INTERFACE
MODEM
PRESTEL TELECOM GOLD ETC
AMSTRAD + SERIAL INTERFACE
NMC
PRINTER
AMSTRAD + SERIAL INTERFACE
NMC
AMSTRAD + SERIAL INTERFACE
AMSTRAD + SERIAL INTERFACE
NMC
TERMINAL

Figure 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.

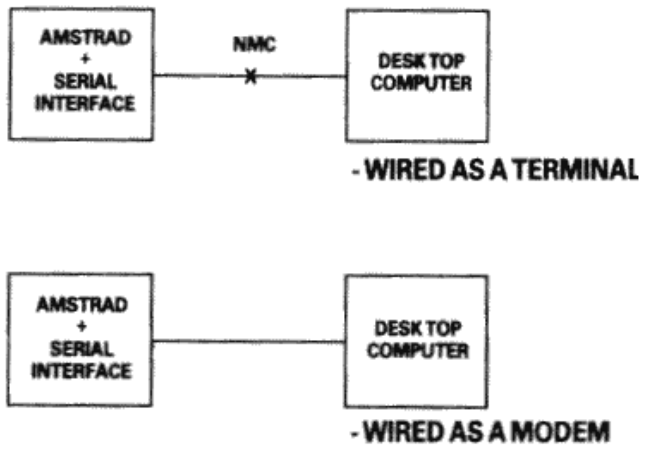


Figure 5.

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.

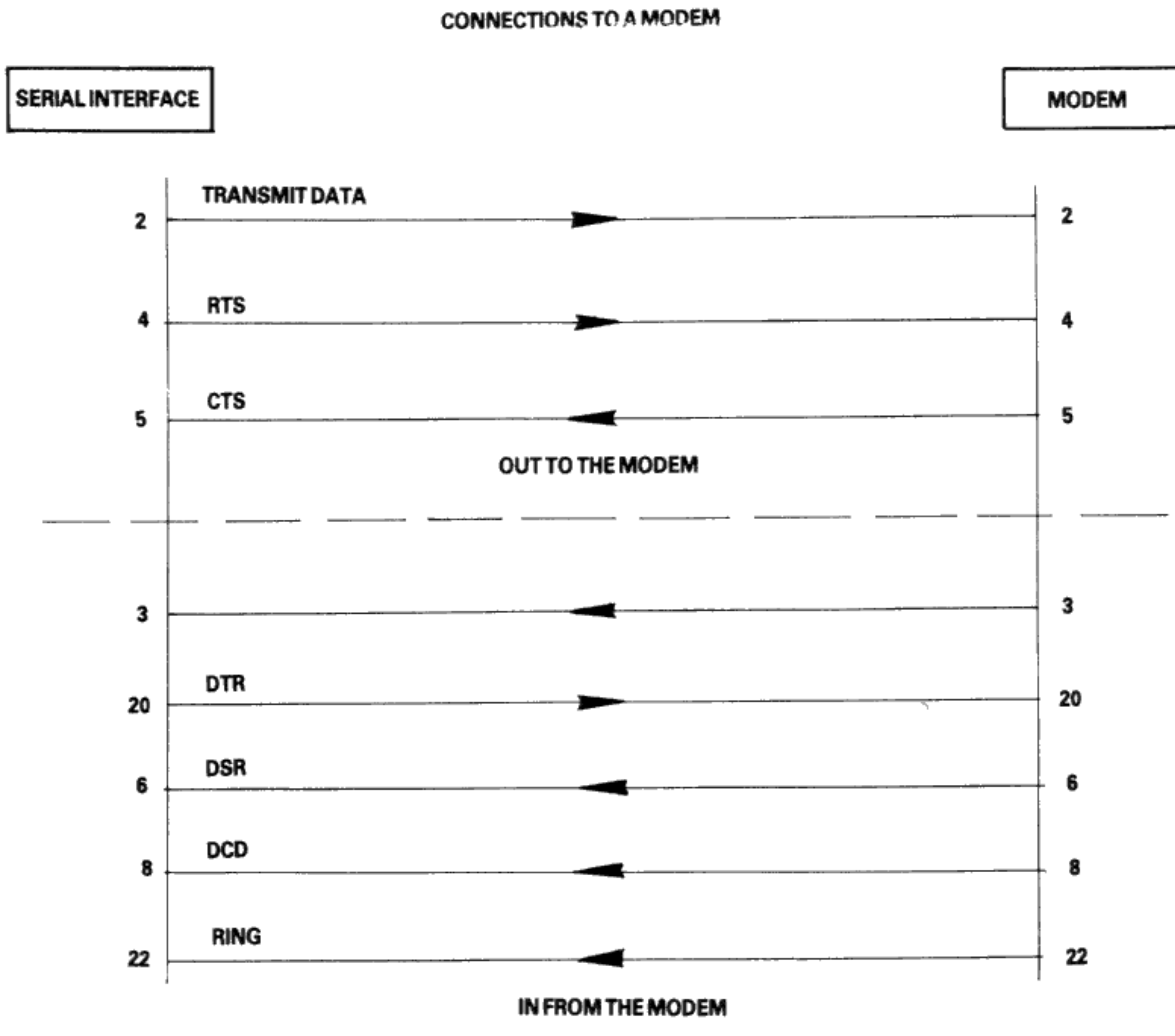


Figure 6.

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.

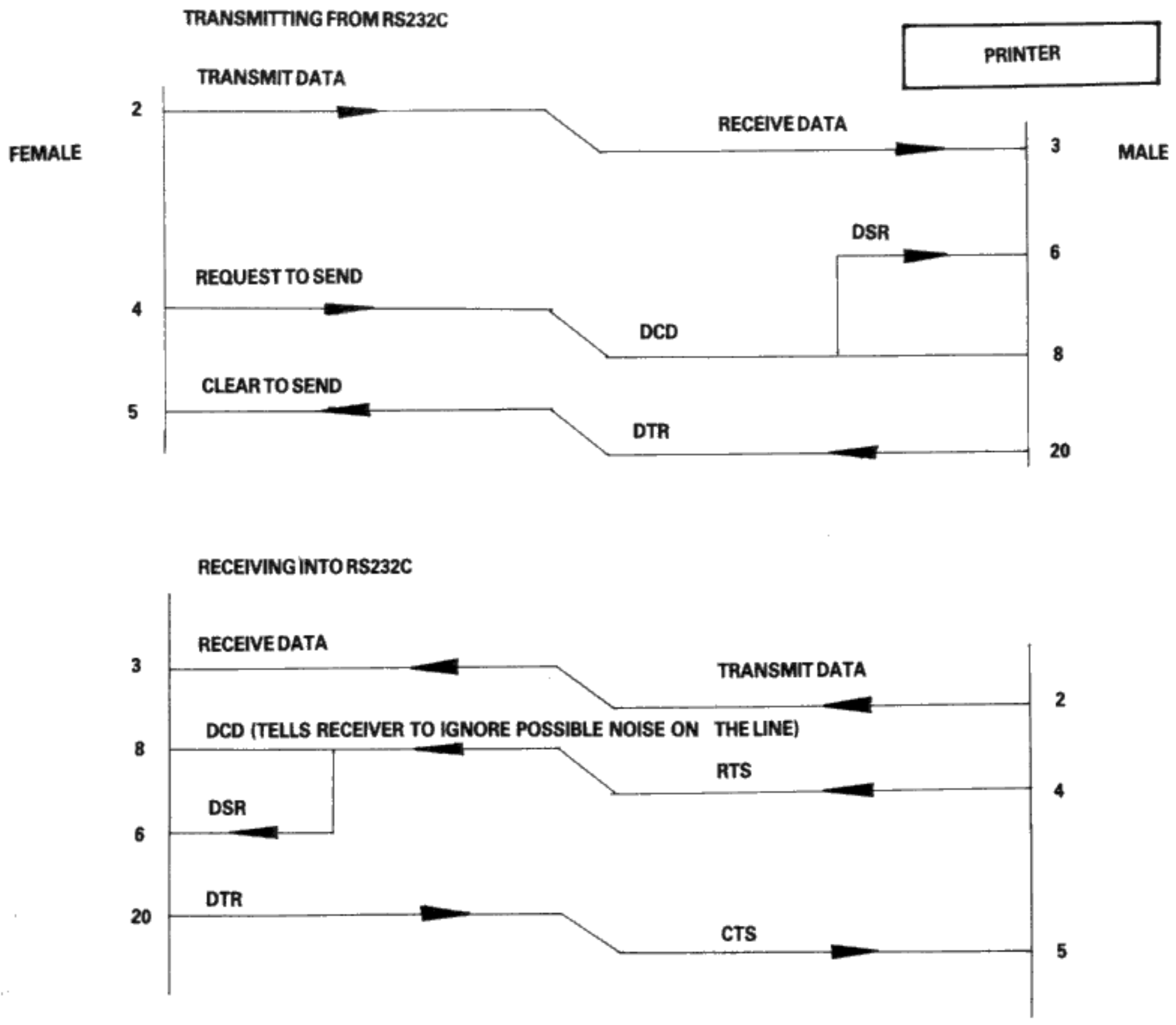
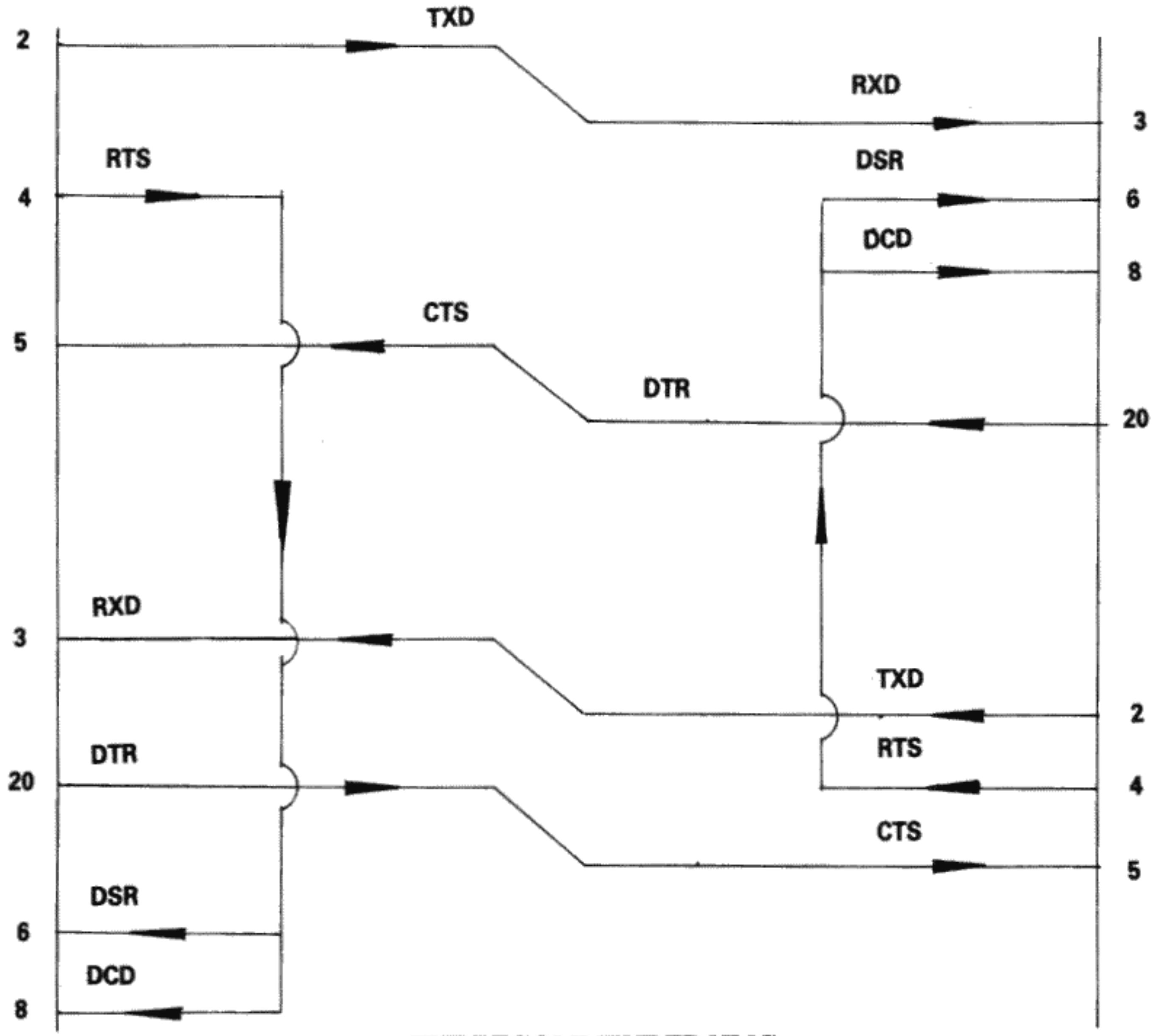


Figure 7.

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.



THE RECOMMENDED NMC

Figure 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

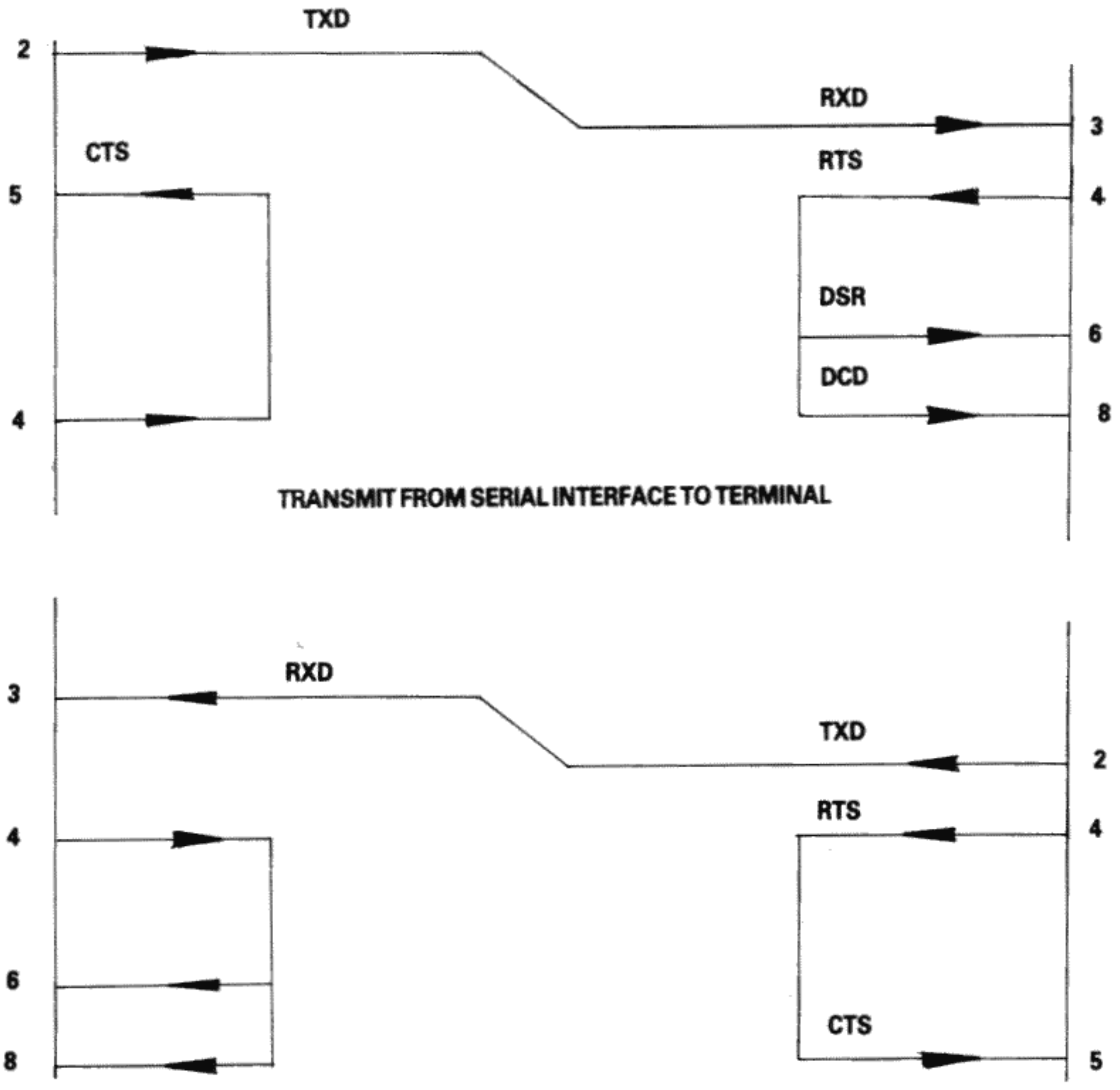


Figure 9

# APPENDIX 4: 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 |
|---|---|---|

| 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 |
| 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 5: PPC POWER USAGE

The PPC power usage from the batteries is as follows:

| | |
|---|---|
| 1. Power OFF (internal RTC running) | 1mA |
| 2. Power ON and RAM resident process, disk motors off | 620mA |
| 3. Power ON and RAM resident process, disk motors on | 870mA |

It is not recommended that the modem be used with the batteries otherwise the usable 1A rating of the batteries will be exceeded. As the batteries age their voltage output will drop and the current regulator will increase the current demand to maintain the internal electronics board power supply output. This process will foreshorten battery life considerably, as the batteries age.

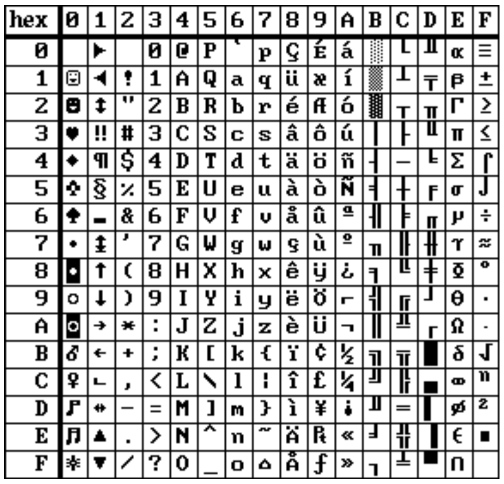
The PPC power usage from the AC adapter is as follows:

| | |
|---|---|
| 1. Power OFF (batteries supplying RTC) | 100mA |

2. Power ON and ROM resident process, disk motors off 740mA
3. Power ON and RAM resident process, disk motors off 850mA
4. Power ON and RAM resident process, disk motors on 1100mA
5. Power ON and RAM resident process, modem active 1220mA
6. Power ON and RAM resident process, modem & disk active 1470mA

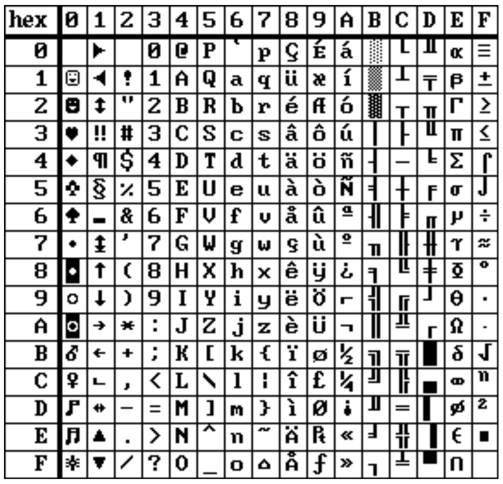
The power usage during power off condition is due to a bleed resister in the adapter circuit, used to stablize the adapter so that it will not supply open circuit voltage to the computer. The voltage from the AC adapter with just bleed current is approximately 15 volts. The minimum voltage with RAM resident process, modem and disk both active is approximately 13.5 volts.

# APPENDIX 6: ROM CHARACTER SET



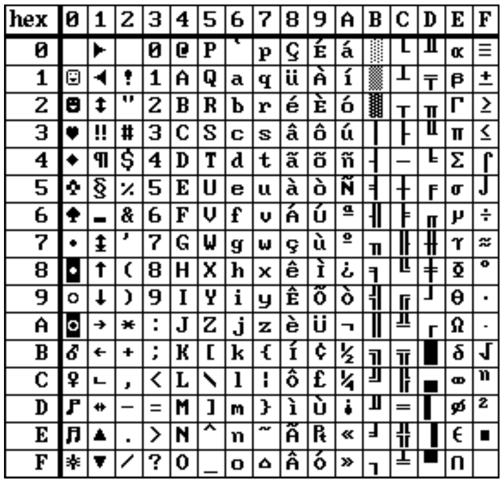
| hex | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 |  | ► |  | 0 | @ | P | ` | p | Ç | É | á | ░ | └ | ╨ | α | ≡ |
| 1 | ☺ | ◄ | ! | 1 | A | Q | a | q | ü | æ | í | ▒ | ┴ | ╤ | β | ± |
| 2 | ☻ | ↕ | " | 2 | B | R | b | r | é | Æ | ó | ▓ | ┬ | ╥ | Γ | ≥ |
| 3 | ♥ | ‼ | # | 3 | C | S | c | s | â | ô | ú | │ | ├ | ╙ | π | ≤ |
| 4 | ♦ | ¶ | $ | 4 | D | T | d | t | ä | ö | ñ | ┤ | ─ | ╘ | Σ | ⌠ |
| 5 | ♣ | § | % | 5 | E | U | e | u | à | ò | Ñ | ╡ | ┼ | ╒ | σ | ⌡ |
| 6 | ♠ | ▬ | & | 6 | F | V | f | v | å | û | ª | ╢ | ╞ | ╓ | µ | ÷ |
| 7 | • | ↨ | ' | 7 | G | W | g | w | ç | ù | º | ╖ | ╟ | ╫ | τ | ≈ |
| 8 | ◘ | ↑ | ( | 8 | H | X | h | x | ê | ÿ | ¿ | ╕ | ╚ | ╪ | Φ | ° |
| 9 | ○ | ↓ | ) | 9 | I | Y | i | y | ë | Ö | ⌐ | ╣ | ╔ | ┘ | Θ | ∙ |
| A | ◙ | → | * | : | J | Z | j | z | è | Ü | ¬ | ║ | ╩ | ┌ | Ω | · |
| B | ♂ | ← | + | ; | K | [ | k | { | ï | ¢ | ½ | ╗ | ╦ | █ | δ | √ |
| C | ♀ | ∟ | , | < | L | \ | l | ¦ | î | £ | ¼ | ╝ | ╠ | ▄ | ∞ | ⁿ |
| D | ♪ | ↔ | - | = | M | ] | m | } | ì | ¥ | ¡ | ╜ | ═ | ▌ | ø | ² |
| E | ♫ | ▲ | . | > | N | ^ | n | ~ | Ä | ₧ | « | ╛ | ╬ | ▐ | ε | ■ |
| F | ☼ | ▼ | / | ? | O | _ | o | ⌂ | Å | ƒ | » | ┐ | ╧ | ▀ | ∩ |  |

**ENGLISH character set**



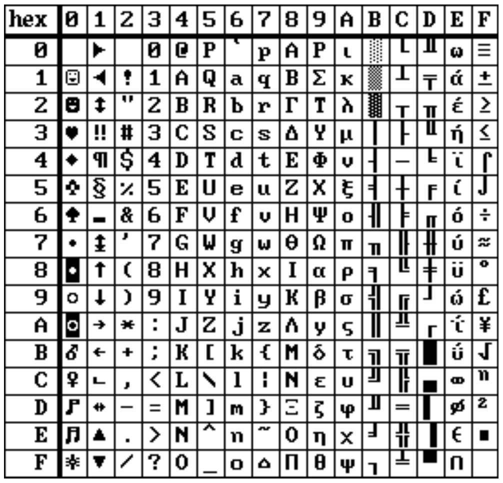
| hex | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 |  | ► |  | 0 | @ | P | ` | p | Ç | É | á | ░ | └ | ╨ | α | ≡ |
| 1 | ☺ | ◄ | ! | 1 | A | Q | a | q | ü | æ | í | ▒ | ┴ | ╤ | β | ± |
| 2 | ☻ | ↕ | " | 2 | B | R | b | r | é | Æ | ó | ▓ | ┬ | ╥ | Γ | ≥ |
| 3 | ♥ | ‼ | # | 3 | C | S | c | s | â | ô | ú | │ | ├ | ╙ | π | ≤ |
| 4 | ♦ | ¶ | $ | 4 | D | T | d | t | ä | ö | ñ | ┤ | ─ | ╘ | Σ | ⌠ |
| 5 | ♣ | § | % | 5 | E | U | e | u | à | ò | Ñ | ╡ | ┼ | ╒ | σ | ⌡ |
| 6 | ♠ | ▬ | & | 6 | F | V | f | v | å | û | ª | ╢ | ╞ | ╓ | µ | ÷ |
| 7 | • | ↨ | ' | 7 | G | W | g | w | ç | ù | º | ╖ | ╟ | ╫ | τ | ≈ |
| 8 | ◘ | ↑ | ( | 8 | H | X | h | x | ê | ÿ | ¿ | ╕ | ╚ | ╪ | Φ | ° |
| 9 | ○ | ↓ | ) | 9 | I | Y | i | y | ë | Ö | ⌐ | ╣ | ╔ | ┘ | Θ | ∙ |
| A | ◙ | → | * | : | J | Z | j | z | è | Ü | ¬ | ║ | ╩ | ┌ | Ω | · |
| B | ♂ | ← | + | ; | K | [ | k | { | ï | ø | ½ | ╗ | ╦ | █ | δ | √ |
| C | ♀ | ∟ | , | < | L | \ | l | ¦ | î | £ | ¼ | ╝ | ╠ | ▄ | ∞ | ⁿ |
| D | ♪ | ↔ | - | = | M | ] | m | } | ì | Ø | ¡ | ╜ | ═ | ▌ | ø | ² |
| E | ♫ | ▲ | . | > | N | ^ | n | ~ | Ä | ₧ | « | ╛ | ╬ | ▐ | ε | ■ |
| F | ☼ | ▼ | / | ? | O | _ | o | ⌂ | Å | ƒ | » | ┐ | ╧ | ▀ | ∩ |  |

**NORWEGIAN character set**



| hex | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 |  | ► |  | 0 | @ | P | ` | p | Ç | É | á | ░ | └ | ╨ | α | ≡ |
| 1 | ☺ | ◄ | ! | 1 | A | Q | a | q | ü | À | í | ▒ | ┴ | ╤ | β | ± |
| 2 | ☻ | ↕ | " | 2 | B | R | b | r | é | È | ó | ▓ | ┬ | ╥ | Γ | ≥ |
| 3 | ♥ | ‼ | # | 3 | C | S | c | s | â | ô | ú | │ | ├ | ╙ | π | ≤ |
| 4 | ♦ | ¶ | $ | 4 | D | T | d | t | ã | õ | ñ | ┤ | ─ | ╘ | Σ | ⌠ |
| 5 | ♣ | § | % | 5 | E | U | e | u | à | ò | Ñ | ╡ | ┼ | ╒ | σ | ⌡ |
| 6 | ♠ | ▬ | & | 6 | F | V | f | v | Á | Ú | ª | ╢ | ╞ | ╓ | µ | ÷ |
| 7 | • | ↨ | ' | 7 | G | W | g | w | ç | ù | º | ╖ | ╟ | ╫ | τ | ≈ |
| 8 | ◘ | ↑ | ( | 8 | H | X | h | x | ê | Ì | ¿ | ╕ | ╚ | ╪ | Φ | ° |
| 9 | ○ | ↓ | ) | 9 | I | Y | i | y | Ê | Õ | Ò | ╣ | ╔ | ┘ | Θ | ∙ |
| A | ◙ | → | * | : | J | Z | j | z | è | Ü | ¬ | ║ | ╩ | ┌ | Ω | · |
| B | ♂ | ← | + | ; | K | [ | k | { | Í | ¢ | ½ | ╗ | ╦ | █ | δ | √ |
| C | ♀ | ∟ | , | < | L | \ | l | ¦ | Ô | £ | ¼ | ╝ | ╠ | ▄ | ∞ | ⁿ |
| D | ♪ | ↔ | - | = | M | ] | m | } | ì | Ù | ¡ | ╜ | ═ | ▌ | ø | ² |
| E | ♫ | ▲ | . | > | N | ^ | n | ~ | Ã | ₧ | « | ╛ | ╬ | ▐ | ε | ■ |
| F | ☼ | ▼ | / | ? | O | _ | o | ⌂ | Â | Ó | » | ┐ | ╧ | ▀ | ∩ |  |

**PORTUGESE character set**

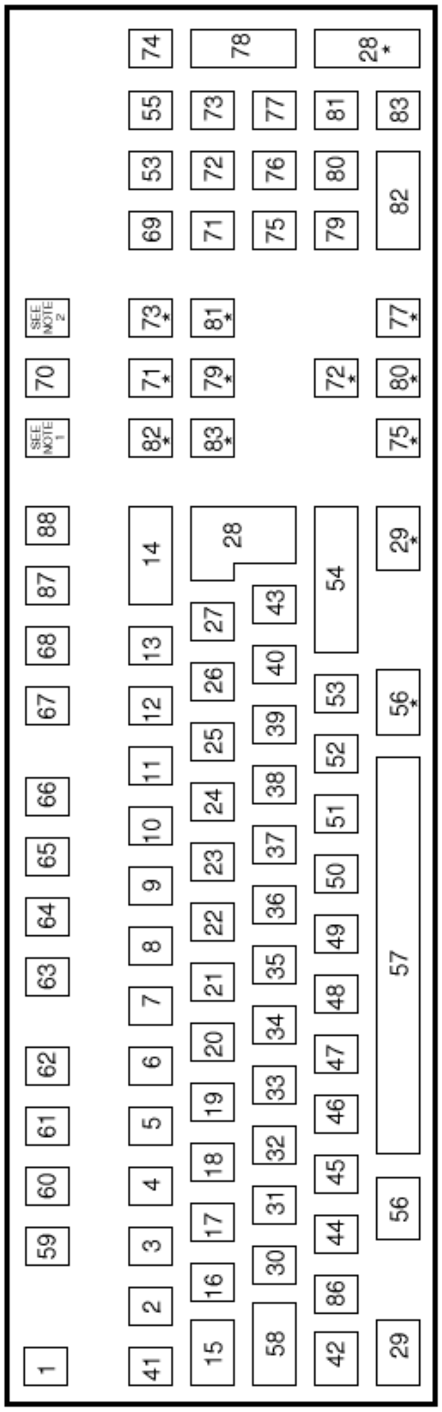


| hex | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 |  | ► |  | 0 | @ | P | ` | p | Α | Ρ | ι | ░ | └ | ╨ | ω | ≡ |
| 1 | ☺ | ◄ | ! | 1 | A | Q | a | q | Β | Σ | κ | ▒ | ┴ | ╤ | ά | ± |
| 2 | ☻ | ↕ | " | 2 | B | R | b | r | Γ | Τ | λ | ▓ | ┬ | ╥ | έ | ≥ |
| 3 | ♥ | ‼ | # | 3 | C | S | c | s | Δ | Υ | μ | │ | ├ | ╙ | ή | ≤ |
| 4 | ♦ | ¶ | $ | 4 | D | T | d | t | Ε | Φ | ν | ┤ | ─ | ╘ | ϊ | ⌠ |
| 5 | ♣ | § | % | 5 | E | U | e | u | Ζ | Χ | ξ | ╡ | ┼ | ╒ | ί | ⌡ |
| 6 | ♠ | ▬ | & | 6 | F | V | f | v | Η | Ψ | ο | ╢ | ╞ | ╓ | ό | ÷ |
| 7 | • | ↨ | ' | 7 | G | W | g | w | Θ | Ω | π | ╖ | ╟ | ╫ | ύ | ≈ |
| 8 | ◘ | ↑ | ( | 8 | H | X | h | x | Ι | α | ρ | ╕ | ╚ | ╪ | ϋ | ° |
| 9 | ○ | ↓ | ) | 9 | I | Y | i | y | Κ | β | σ | ╣ | ╔ | ┘ | ώ | £ |
| A | ◙ | → | * | : | J | Z | j | z | Λ | γ | ς | ║ | ╩ | ┌ | ΐ | ¥ |
| B | ♂ | ← | + | ; | K | [ | k | { | Μ | δ | τ | ╗ | ╦ | █ | ΰ | √ |
| C | ♀ | ∟ | , | < | L | \ | l | ¦ | Ν | ε | υ | ╝ | ╠ | ▄ | ∞ | ⁿ |
| D | ♪ | ↔ | - | = | M | ] | m | } | Ξ | ζ | φ | ╜ | ═ | ▌ | ø | ² |
| E | ♫ | ▲ | . | > | N | ^ | n | ~ | Ο | η | χ | ╛ | ╬ | ▐ | ε | ■ |
| F | ☼ | ▼ | / | ? | O | _ | o | ⌂ | Π | θ | ψ | ┐ | ╧ | ▀ | ∩ |  |

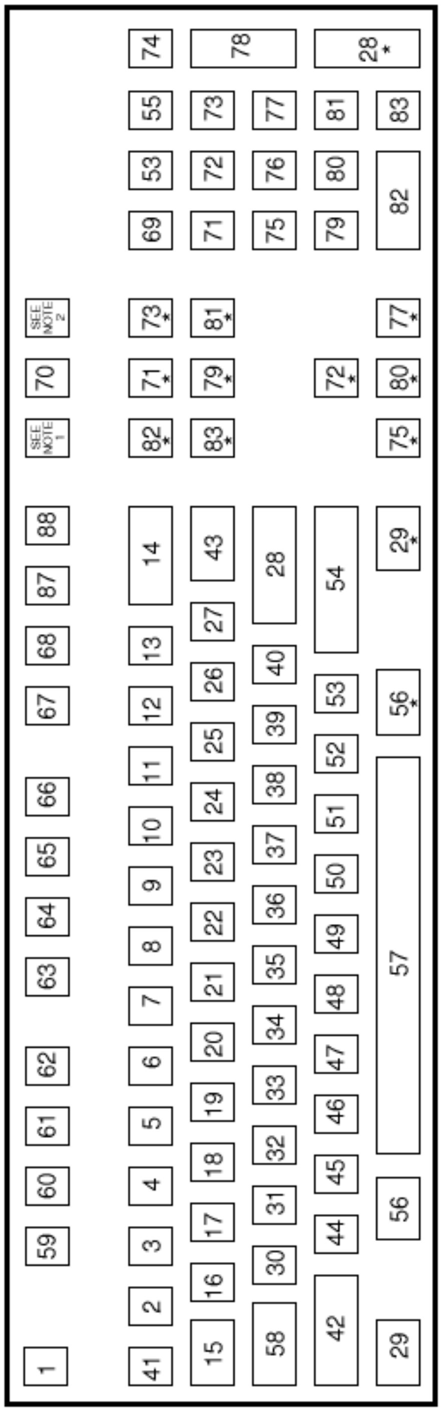
**GREEK character set**

NOTE: Character sets are selected using the character set selection links described on page 141.

# APPENDIX 7: KEYBOARD KEYCODES



**UK keyboard**



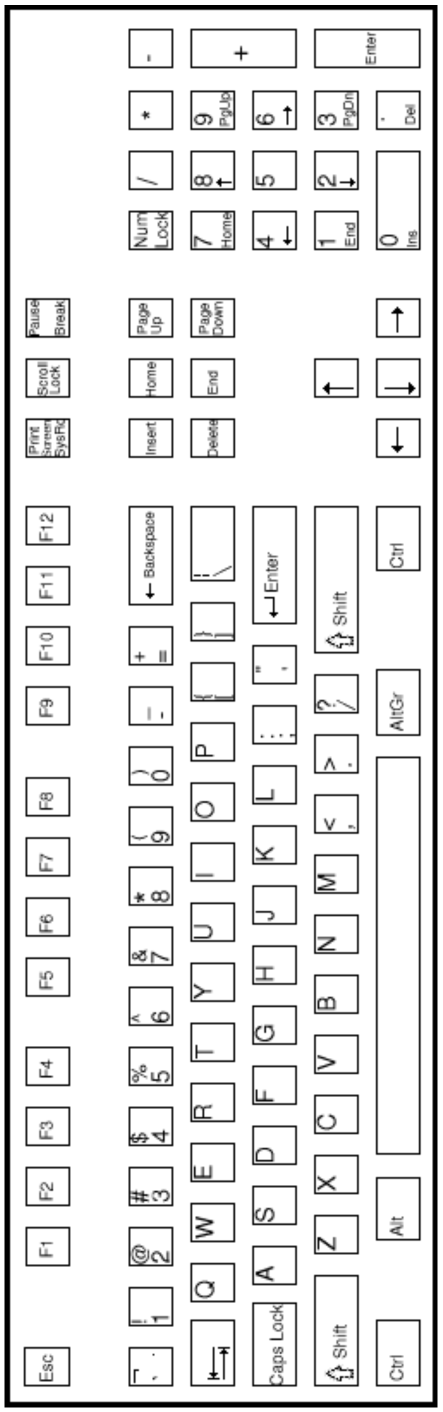
**USA keyboard**

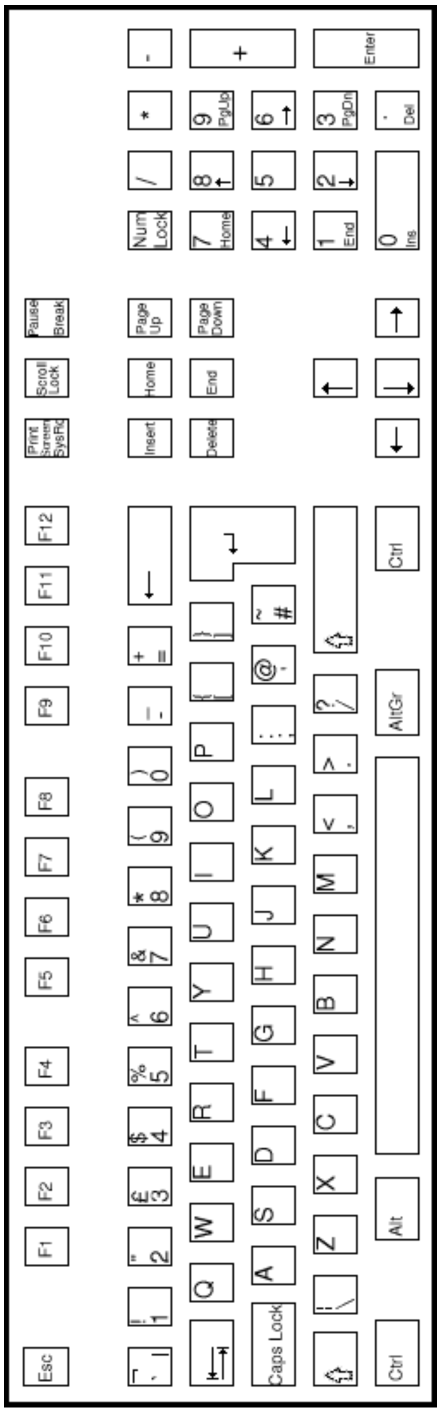
* keys designated by an asterix are extended keys and give extended key sequences as documented in section 2.3.4.

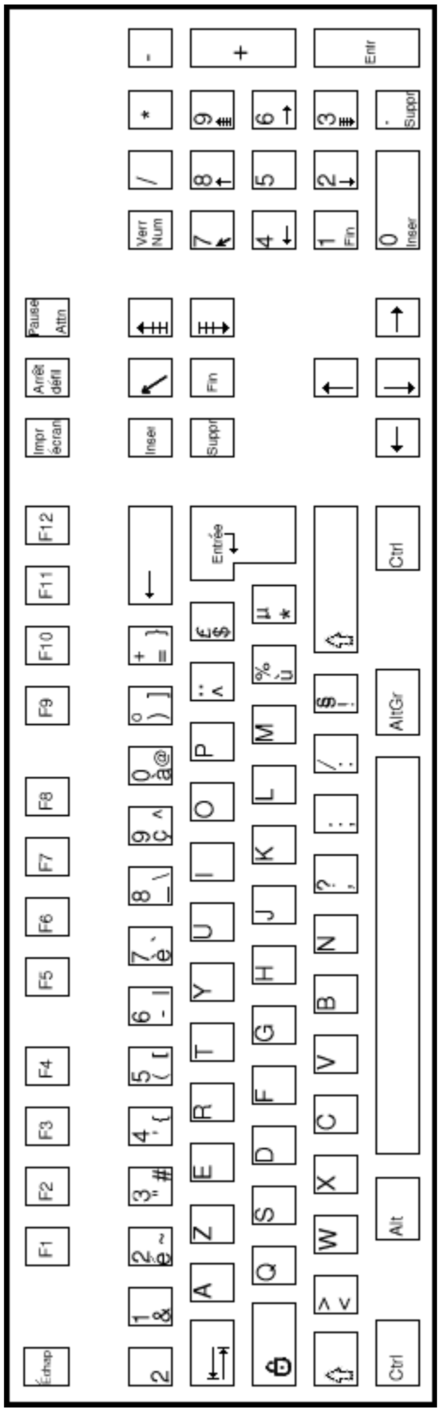
Note 1: This key, the Print Screen/Sys req key, gives the keycode sequence E0h, 2Ah, E0h, 37h when typed.

Note 2: This key, the Pause/Break key, gives the keycode sequence E1h, 1Dh, 45h when typed.

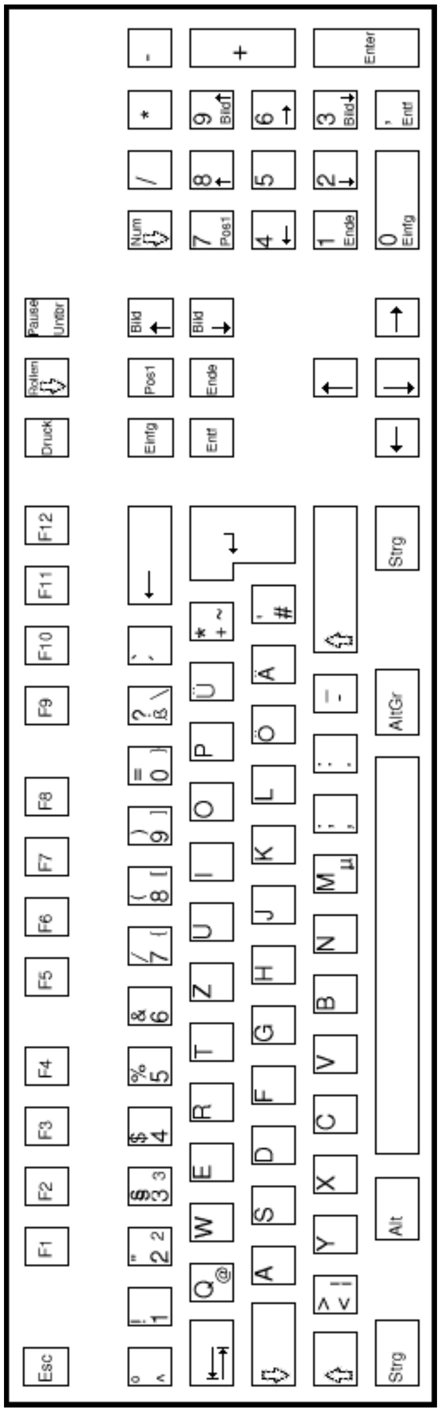
# APPENDIX 8: KEYBOARD LAYOUTS


**USA keyboard**

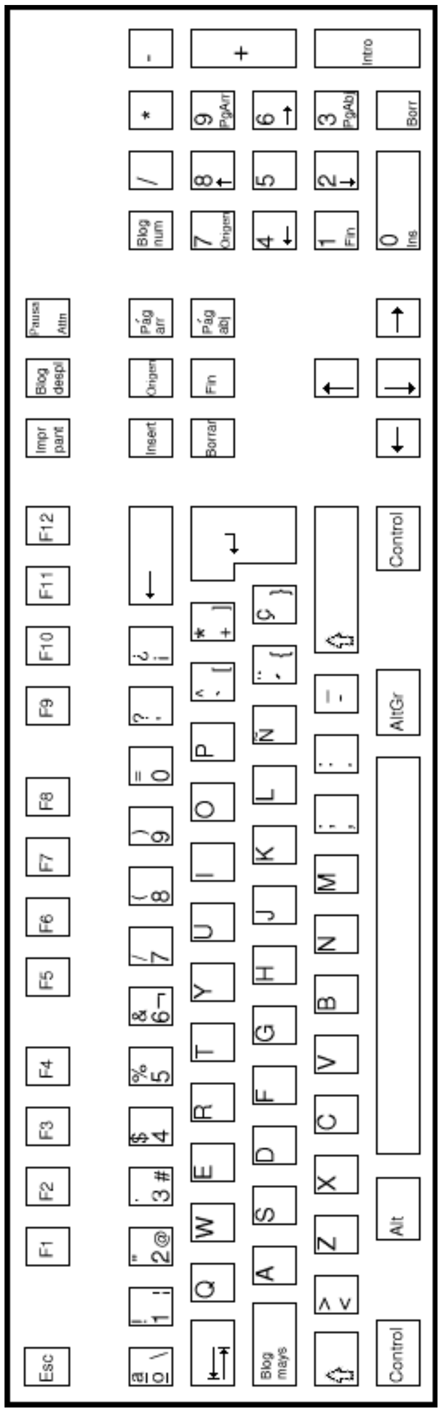

**UK keyboard**

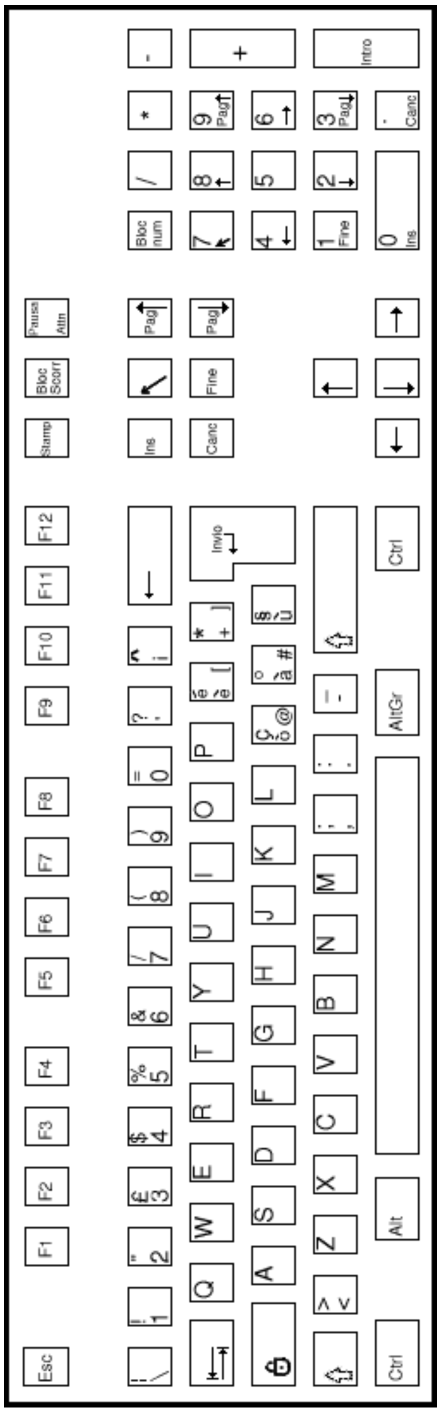


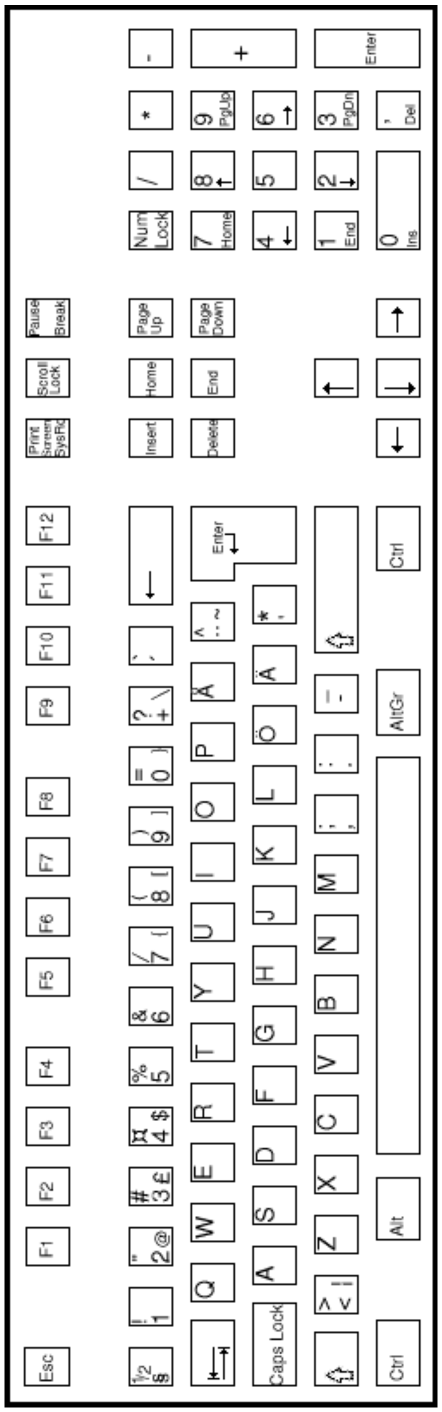
FRENCH keyboard

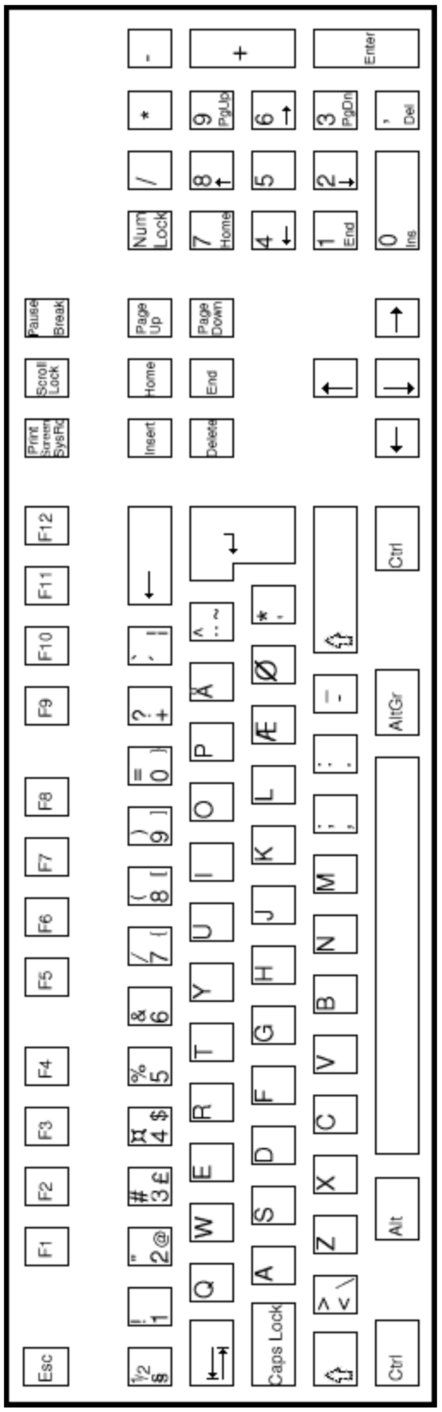


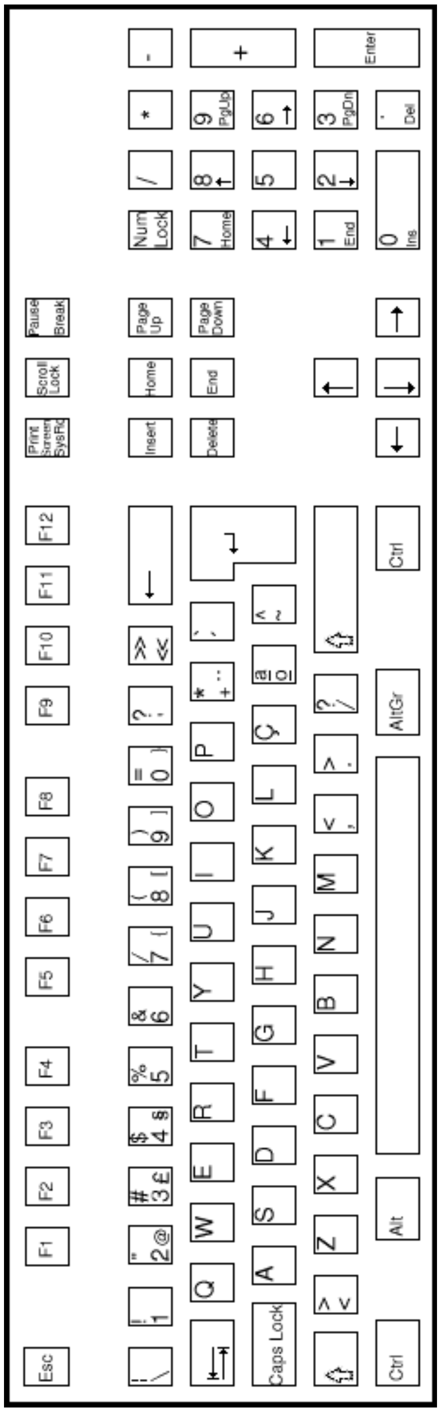
GERMAN keyboard

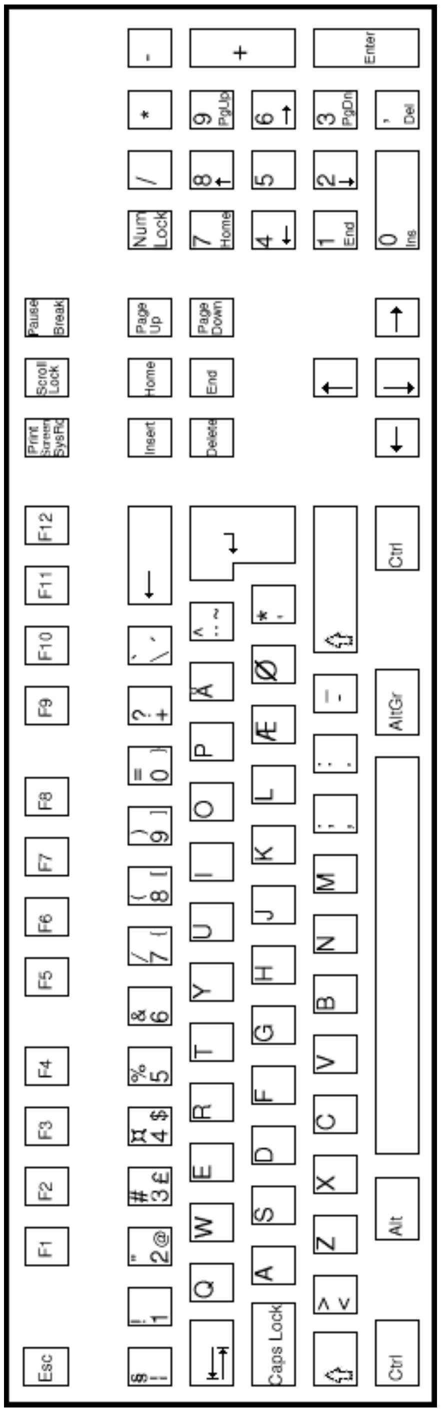

**SPANISH keyboard**


**ITALIAN keyboard**


SWEDISH keyboard


DANISH keyboard


**PORTUGESE keyboard**


**NORWEGIAN keyboard**

# APPENDIX 9: THE LINKER PROGRAM (MS-LINK)

## 9.1 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.

## 9.2 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:

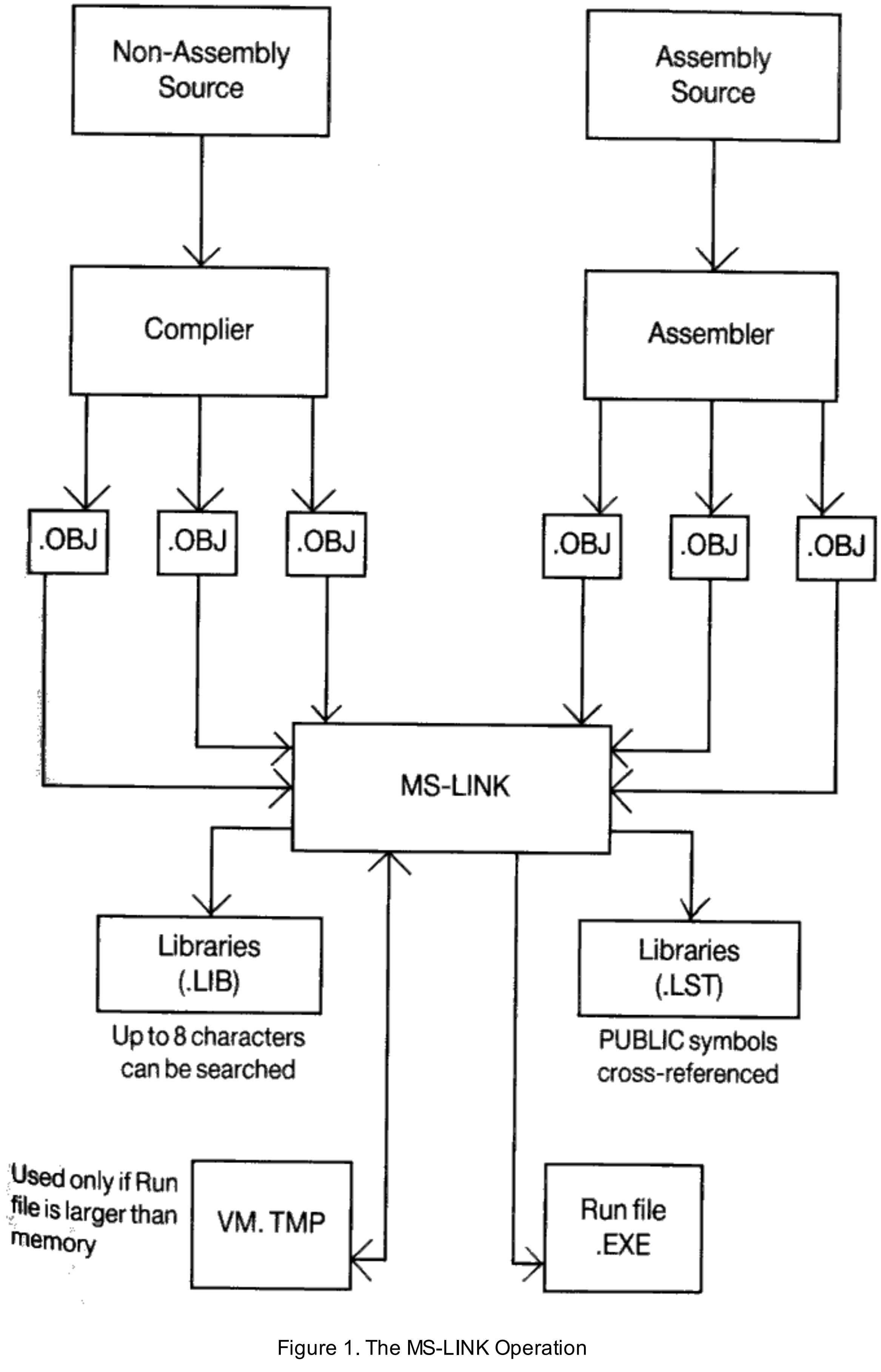
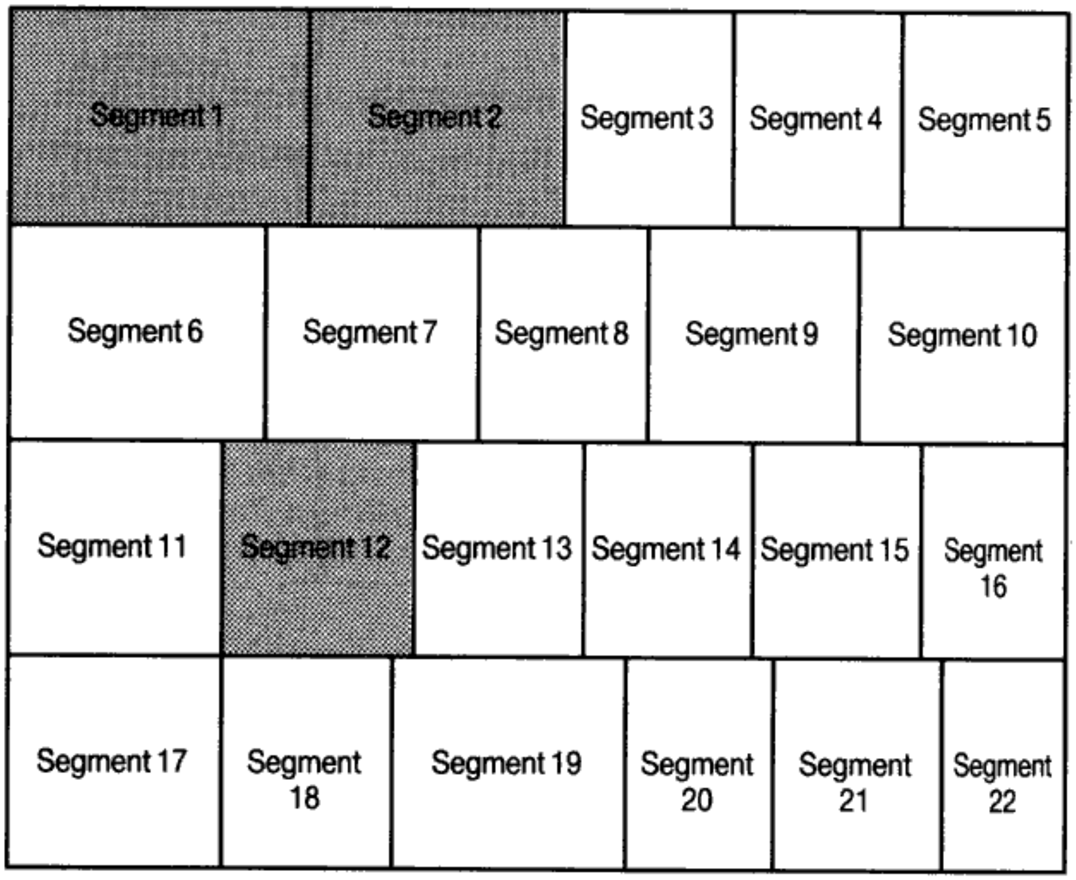


Figure 1. The MS-LINK Operation

## 9.3 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.

## 9.4 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.

### 9.4.1 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.

### 9.4.2 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)

### 9.4.3 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.

## 9.5 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>

### 9.5.1 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.)

### 9.5.2 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.

### 9.5.3 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.

## 9.6 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.

## 9.7 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).

## 9.8 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.

## 9.9 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
```

## 9.10 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 10: 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 11: THE DEBUG UTILITY PROGRAM (DEBUG)

## 11.1 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.

## 11.2 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> ]]
```

### 11.2.1 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.

### 11.2.2 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.

## 11.3 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 11.1 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. |

PARAMETER DEFINITION

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
```

The limit for <range> is 10000 hex. To specify a value of <10000> hex within four digits, type 00 (or 0).

**<list>**

A series of <byte> values or of <string>s. <list> must be the last parameter on the command line.

Example:

```
fcs:100 42 45 52 54 41
```

**<string>**

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:

```
"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 "WORD PTR" or "BYTE PTR". Acceptable abbreviations are "WO" and "BY". 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)
```

DXDSPC
SPESF

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.

## 11.5 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 |
| 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 12: 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 13: 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 14: 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 15: 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 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.

# APPENDIX 16: THE FDISK UTILITY PROGRAM

The FDISK utility configures hard disks for MS-DOS.

A hard disk is divided into separate areas, called partitions. There will always need to be at least one partition for MS-DOS. In addition, DOS 3.3 allows extra partitions to be allocated which can be handled as though they were separate hard disk drives (referenced as though they were drives d:, e: etc.) In addition, FDISK will recognise and report the presence of non-DOS partitions on a hard disk. Partitions of this sort are installed by other operating systems such as XENIX.

Syntax:
FDISK

The FDISK command displays a series of menus to help you partition your hard disk for MS-DOS. With the FDISK command you can:

- Create a primary MS-DOS partition.
- Create an extended MS-DOS partition.
- Change the active partition.
- Delete an MS-DOS partition.
- Display partition data.
- Select the next fixed disk for partitioning on a system with multiple fixed disks.

WARNING Reconfiguring your disk with FDISK destroys all existing files. Be sure to have a backup of all files on your hard disk before you create a MS-DOS partition with FDISK.

After typing "FDISK" the main menu is displayed:

```
Fixed Disk Setup Program Version 3.30
(C)Copyright Microsoft Corp. 1987

FDISK Options

Current Fixed Disk Drive: 1

Choose one of the following:

    1. Create DOS Partition
    2. Change Active Partition
    3. Delete DOS Partition
    4. Display Partition Data
    5. Select Next Fixed Disk Drive

Enter choice: [1]

Press [ESC] to return to DOS
```

You will see that each option has a number associated with it, and there is a box containing a number labelled:

Enter choice:

This box contains the current choice. Initially it contains the number 1. This is the option that the computer thinks you are most likely to require, and is called the default. If you press a number, you will see it appear in the box. When you have the option you want, press the [RETURN] key.

Each option is described in detail in the sections that follow.

Note that menu item 5, enclosed in curly brackets `{ }`, is optional and will only appear when one or more additional physical hard disk drives are present.

## 16.1 Creating a DOS partition

When you choose the first main menu option, and if your hard disk is not yet completely partitioned, FDISK displays a screen like the following:

```
Create DOS Partition

Current Fixed Disk Drive:  1

      1. Create Primary DOS partition
      2. Create Extended DOS partition
      3. Create Logical DOS Drive(s) in
      the Extended DOS partition

Enter choice: [1]

Press ESC to return to FDISK options
```

Note that if no extended partitions exist, the third option (in curly brackets `{ }`) is not displayed.

Selection 1: Create primary DOS partition

You must create a primary MS-DOS partition before you can create any extended MS-DOS partition on your hard disk. In most cases you will need only one MS-DOS partition for your entire disk. To create a primary partition press the [RETURN] key to accept the default selection (1).

The Create Primary DOS Partition menu appears next:

```
Create Primary DOS Partition

Current Fixed Disk Drive:  1

Do you wish to use the maximum size
for a DOS partition and make the DOS
partition active (Y/N).........? [Y]
```

Press ESC to return to FDISK Options

If you use the entire hard disk for MS-DOS, you will use the FDISK program only once to create the primary MS-DOS partition. If you want to use the entire disk (up to 32 megabytes) fro MS-DOS press the [RETURN] key to accept the default selection (Y).

FDISK now displays the following screen:

```
System will now restart

Insert DOS diskette in drive A:
Press any key when ready ...
```

Put your MS-DOS disk in drive A and press any key to restart MS-DOS.

Once MS-DOS has restarted you will need to format your hard disk to that MS-DOS can use it. If you want to start (bootstrap) MS-DOS from your hard disk remember to use the /S switch with the format command. The format command for a hard disk in drive C would be as follows:

```
FORMAT c: /s
```

## Creating more than one DOS partition

When you chose the Create Primary MS-DOS partition menu selection, you had the option of using the maximum size (available on your hard disk) for the DOS partition. In order to create more than one DOS partition you must create a primary partition which is less than the maximum size. To do this type [N] (for No) in response to the first Create Primary Dos Partition menu. When you press [RETURN] with the N displayed in the choice box (and after about 4 seconds hesitation) FDISK displays the following screen:

```
Create Primary DOS Partition

Current Fixed Disk Drive:  1

Total disk space is 614 cylinders.
Maximum space available for partition
is 614 cylinders.
```

```
        Enter partition size..........:[614]


        No partitions defined

        Press ESC to return to FDISK Options
```

The space available on your disk is measured in cylinders, also called tracks (and the number shown, 614, is just a typical number for illustration purposes). This menu shows the total number of cylinders available for a hard disk partition, and prompts you to enter the size for size of your primary partition. For illustration purposes, let us say that we want an extended partition of 200 cylinders (giving approximately 2/3rds to the primary partition and approximately 1/3rd to the extended partition). In this case (using our 614 cylinder disk) you would enter 414 as the primary partition size and press the [RETURN] key. The following confirmation screen would now appear:

```
        Create Primary DOS Partition

        Current Fixed Disk Drive:  1

        Partition Status   Type  Start  End Size
         C: 1           PRI DOS      0  413  414

        Primary DOS partition created

        Press ESC to return to FDISK Options
```

At this point the only thing you can do with FDISK is to press the [Esc] key and you will be returned to the original (top level) menu. But there's one extra line on the screen below the choice line:

```
        WARNING! No partitions marked active
```

This is because you have created a partition less than the maximum and FDISK has not automatically made it active so this will have to be done later using main menu, selection 2. The active partition is the one that the bootstrap program will expect to find MS-DOS on so that it can startup MS-DOS.

If the hard disk has already been partitioned, the above screens for allocating the primary partition would not have been displayed. Instead FDISK displays a screen such as:

```
        Create Primary DOS Partition

        Current Fixed Disk Drive:  1


        Primary DOS partition already exists.

        Press ESC to return to FDISK Options
```

And (as expected) pressing [Esc] takes you back to the main menu. No other key has any effect.

## Selection 2: Create Extended DOS partition

Once you have created a primary MS-DOS partition, you can create extended MS-DOS partitions on your hard disk. In most cases you will need only one MS-DOS partition for your entire disk unless the total capacity of your hard disk is greater than 32 megabytes or you have some other special need for using multiple partitions. In order to designate one or more logical drives, use menu selection 2 in the Create DOS partition menu. When you press [2] and the [RETURN] key (after about 4 seconds hesitation) FDISK displays a screen something like this (using the disk size and primary allocation numbers of the previous illustration):

```
        Create Extended DOS Partition

        Current Fixed Disk Drive:  1
```

```
Partition Status   Type  Start  End Size
 C: 1          PRI DOS      0  413  414


Total disk space is 614 cylinders.
Maximum space available for partition
is 200 cylinders.

Enter partition size..........:[200]

Press ESC to return to FDISK Options
```

This menu is very similar to the primary creation menu and it gives you the current status of the disk partition(s) in cylinders. Following the previous illustration you could press [RETURN] to allocate the total (default) of 200 cylinders. If, for instance, 150 were entered you would leave 50 cylinders not allocated to MS-DOS, and since FDISK only allows one primary partition and one extended partition, the unused space would be wasted unless some other operating system could use the unused space. (There is actally room for four partitions on the bootstrap sector of your hard disk).

Following the previous illustration assume that you select the default (200) and press the [RETURN] key. The following confirmation screen is displayed by FDISK:

```
Create Primary DOS Partition

Current Fixed Disk Drive:  1

Partition Status   Type  Start  End Size
 C: 1          PRI DOS      0  413  414
    2          EXT DOS    414  613  200

Extended DOS partition created

Press ESC to return to FDISK Options
```

FDISK will not accept any key but [Esc] and when pressed you do not exactly return to the (top level) FDISK Options menu as expected. Instead, FDISK knows you will want to allocate logical drives within the extended partition and automatically enters the Create DOS partition option 3, Create logical drive(s) in the extended DOS partition.

When an extended DOS partition already exists and selection 2 is entered, FDISK displays a screen very similar to the screen above with the message: Extended DOS partition already exists. The only thing you can do in this instance is press the [Esc] key.

## Selection 3: Create Logical DOS Drive(s) in the Extended DOS partition

Once you have created an extended MS-DOS partition, you can create logical MS-DOS drives within the extended MS-DOS partition; and once configured and formatted, MS-DOS3.3 will recognize them as though they were real, physical, hard disk drives. Older versions of MS-DOS will not recognize the extended partition as a native DOS partition and will not support this capability. Considering transportability (which is not normally a major concern with hard disks), extended partitions and logical drives are MS-DOS 3.3 unique and older DOS versions (or CP/M) cannot access information in these areas.

When Create DOS Partition menu option 3 is selected, the following style screen is presented by FDISK (assuming the 200 cylinder allocation of the previous illustration):

```
Create Logical DOS Drive(s)

No logical drives defined

Total partition size is 200 cylinders.
Maximum space available for logical
drive is 200 cylinders.
```

```
        Enter logical drive size..........[200]

        Press ESC to return to FDISK Options
```

For illustration purposes assume you want two logical drives of 100 cylinders each. To do this just type the number 100 and press [RETURN] and FDISK hesitates momentarily while doing the allocation, and presents the following screen:

```
        Create Logical DOS Drive(s)

        Drv Start End  Size
         D:  414   513  100

        Total partition size is 200 cylinders.
        Maximum space available for logical
        drive is 100 cylinders.

        Enter logical drive size..........[100]

        Logical DOS drive created, drive letters
        changed or added

        Press ESC to return to FDISK Options
```

You will note that the next logical drive size will default to 100 cylinders, so just press [RETURN] and you will get the above screen with the second drive's parameters displayed (E: 514 613 100). Note that when the system is next bootstrapped there will be drives c:,d: and e: available under DOS 3.3. Pressing [Esc] at this point returns to the top level FDISK Options menu.

Remember that any logical drives allocated must be formatted using the format command prior to their usage and this should be done immediately after the primary drive is formatted.

## 16.2 Changing the active partition

This main menu selection causes FDISK to display information about all the partitions on your hard disk. It shows you the size, position, type and whether active or not, for each partition. There is only one active partition on a hard disk, and that is the one the computer uses when it is first turned on. All the other partitions are inactive. If you have only one partition, it has to be active.

If you had more than one partition the information would be displayed on the screen something like this:

```
        Change Active Partition

        Current Fixed Disk Drive:  1

        Partition Status   Type  Start  End Size
         C: 1           PRI DOS      0  413  414
            2           EXT DOS    414  613  200

        Total disk space is  614  cylinders

        Enter the number of the partition you
        want to make active...............: [1]

        Press ESC to return to FDISK options
```

Type the number of the partition you want to make active, and press the [RETURN] key. The computer will choose the currently active partition as the default, to reduce the risk of accidents.

If there were partitions created by other operating systems (such as XENIX) there would be extra partitions displayed with the type non-DOS displayed. These partitions cannot be made active by FDISK.

If, as is likely, you have just one partition covering the entire hard disk then FDISK will inform you that

```
        Partition 1 is already active

        Press ESC to return to FDISK options
```

In that case, you have to press [Esc] to return to the main menu. No other key will have any effect.

## 16.3 Deleting the DOS partition

When main menu, option 3 is chosen, FDISK displays a screen showing a menu such as the following:

```
        Delete DOS Partition

        Current Fixed Disk Drive:  1

        Choose one of the following:

              1. Delete Primary DOS partition
              2. Delete Extended DOS partition
              3. Delete Logical DOS Drive(s) in
                 the Extended DOS partition

        Enter choice: [ ]

        Press ESC to return to DOS
```

FDISK puts up no default in the choice box and you must enter a number otherwise FDISK will not know what to do. Note that selection number 3 (displayed in brackets `{ }`) is not displayed unless there are logical drives which can be deleted. There is also an order of deletion which FDISK enforces and this is from the lowest level up, that is, first logical drives, then extended partitions and finally the primary partition. Any attempt to delete in another order will evoke the notification that the deletion cannot be done with the lower element in existence. Note also that FDISK will not delete a non-DOS partition since these are to be managed by the program which initially placed them on the disk.

### Deleting the Primary DOS Partition

When delete menu option 1 iss chosen, there can be no extended partition (and no logical drives) and if this is true, FDISK displays a screen showing information on each partition on your hard disk, and ask you whether or not you want to delete the primary DOS partition.

BEWARE: If you answer Y to that question, the primary DOS partition and all the data that it contained will be lost for ever. For that reason, FDISK selects N as the default.

The screen will look something like this:

```
        Delete Primary DOS Partition

        Current Fixed Disk Drive:  1

        Partition Status   Type  Start  End Size
         C: 1        A  PRI DOS      0  413  414

        Warning! Data in the Primary DOS
        partition will be lost. Do you wish
        to continue......................? [N]

        Press ESC to return to FDISK options
```

If you don't want to delete your DOS partition, press either the [RETURN] key or to return to the main menu. Otherwise type Y and press the [RETURN] key. The partition will be deleted, and you will be returned to the main menu after being shown the confirmation page.

### Deleting the Extended DOS Partition

When delete menu, option 2 is chosen, there can be no logical drives and if this is true, FDISK displays a screen showing information on each partition on your hard disk, and asks you whether or not you want to delete the Extended DOS partition.

The screen will look something like this:

```
Delete Extended DOS Partition

Current Fixed Disk Drive:  1

Partition Status   Type  Start  End Size
 C: 1       A  PRI DOS      0  413  414
    2          EXT DOS    414  613  200

Warning! Data in the Extended DOS
partition will be lost. Do you wish
to continue.......................? [N]

Press ESC to return to FDISK options
```

If you do not want to delete your extended DOS partition, press either the [RETURN] key or [Esc] to return to the main menu. Otherwise type Y and press the [RETURN] key. The partition will be deleted, and you will be returned to the main menu after being shown the confirmation page.

### Deleting Logical DOS Drive(s)

When delete menu option 3 is chosen, FDISK displays a screen showing information on each logical drive on your hard disk, and asks you which one you want to delete. After the selection you'll be asked to verify your selection.

BEWARE: If you answer Y to that question, the selected logical DOS drive and all the data that it contained will be lost for ever. For that reason, FDISK selects N as the default.

The screen will look something like this:

```
Delete Logical DOS Drive

Drv Start End  Size
D:  414   513  100
E:  514   613  100

Total partition size is 200 cylinders.

Warning! Data in the logical DOS drive
will be lost. What drive do you wish
to delete........................? [D]

Are you sure.....................? [N]

Press ESC to return to FDISK Options
```

If you do not want to delete your logical DOS drive, press either the [RETURN] key or [Esc] to return to the main menu. Otherwise type Y and press the [RETURN] key. The logical drive will be deleted, and if there are more logical drives (such as shown above) you will be recycled through the deletion procedure. Once there are no more logical drives, you will be returned to the main menu after being shown the confirmation page.

## 16.4 Displaying partition data

Choosing the fourth main menu option will display a screen showing information about all the partitions on the disk. It will look something like this:

```
Display Partition Information

Current Fixed Disk Drive:  1

Partition Status   Type  Start  End Size
 C: 1        A  PRI DOS      0  413  414
    2           EXT DOS    414  613  200

Total disk space is 614 cylinders.

The extended DOS partition contains
logical DOS drives. Do you want to
display logical drive information?  [Y]

Press ESC to return to FDISK Options
```

Note that the extra message (contained in brackets `{ }`) will only be displayed when there are logical drives present. If you press Y and the [RETURN] key the screen wiuld look something line this:

```
Display Partition Information

Drv Start End  Size
 D: 414    513  100
 E: 514    613  100

Press ESC to return to FDISK Options
```

And (as expected) pressing [Esc] returns you to the main menu.

## 16.5 Selecting the Next Fixed Disk Drive.

This main menu option is not displayed unless you have more than one hard disk drive on your system. When you select main menu item 5 and press the [RETURN] key, the main menu reappears and the message:

```
Current Fixed Disk Drive:  n
```

below the options title line shows the newly selected drive number. All other menus which show the currently fixed drive number will also be effected. Selecting menu item 5 again advances to the next physical drive or if the current drive was the last one then the next drive is the first drive (ie. the next drive chain is circular).

# APPENDIX 17: THE BACKUP COMMAND.

The BACKUP command backs-up one or more files from one disk to another.

Syntax:
    BACKUP [drive 1:][\][path\][filename.filetype]
    [drive2:][/S][/M][/A][/F][/D:date][/T:time]
    [/L:[[drive[path]filename]

drive1 is the disk that you want to back-up.
drive2 is the target drive onto which the files are backed up.

The BACKUP command can back-up files on disks of different media types (hard disks and floppy disks). BACKUP can also back-up files from one floppy disk to another, even if the disks have a different number of sides or sectors.

You will need to format the disk which is going to store the copies. But it does not have to be formatted in the same way as the source disk. For example, it can have a different number of sectors. However, if you use DISKCOPY to make a back-up copy of a disk, both disks must be formatted in the same way. The copies are normally stored in the root directory of the back-up disk.

The OPTION SWITCHES control:

- Whether existing back-up files on the disk are overwritten by the new copies
- Whether sub-directories are backed-up or not
- If back-ups are made of files that haven't been changed since they were last backed-up
- If back-ups are made of files created before a certain date
- Keeping a record of the back-up process

An exit code is set by BACKUP to record whether the back-up was completed successfully. This can be used in an IF command to determine the next command.

OPTION SWITCHES:

/S
: Backs-up subdirectories.

/M
: Will only back-up those files which have the Archive attribute set. This generally means that the file has been changed since it was last backed-up.

/A
: Adds the files to be backed-up to those already on the back-up disk. It does not erase old files on the back-up disk. This switch will not be accepted if files exist that were backed-up using BACKUP from MS-DOS versions 3.2 or earlier.

/F
: Causes the target disk to be formatted if it is not already formatted. For this switch to function the MS-DOS format command must be accessible by the current path.

/D:date
: Backs-up only those files that were changed on or after the given date. (Date is in the form dd-mm-yy).

/T:time
: Will only back-up those files that were changed at, or after the given time. (Time is in the form hh[:mm[:ss]]).

/L:filename
: Makes a BACKUP.LOG entry in the specified file. If you do not specify the filename, BACKUP places a file called BACKUP.LOG in the root directory of the disk which contains the files being backed-up.

If the BACKUP.LOG file already exists, BACKUP appends the current entry to the file.

The log file uses the following format:

The first line lists the time and date of the back-up. A line for each backed-up file lists the filename and number of the back-up disk on which the file resides.

The back-up log file can be used when restoring a particular file from a floppy disk, but you must specify which disk to restore so that the RESTORE command does not have to search for files. BACKUP displays the name of each file as it is backed-up.

Notes:

If there are more files than can be backed-up to a single floppy disk, BACKUP directs you to insert additional floppies as required. You should label and number each back-up disk consecutively to help you restore the files properly with the RESTORE command.

If you're sharing files, MS-DOS will only allow you to back-up the files to which you have access.

You won't be able to restore these files using the PC-DOS version of the RESTORE command, nor with the old (MS-DOS 3.2 or earlier) versions of the RESTORE command.

Unless you use the /A switch, BACKUP erases the old files on a back-up disk before adding the new files to it.

You should not use the BACKUP command if the drive you are backing-up has been joined or substituted with the

ASSIGN, JOIN or SUBST commands. If you do, you may not be able to restore the files with the RESTORE command.

Exit Codes:

BACKUP returns the following exit codes:

0
: Normal completion.

1
: No files found to backup.

2
: Some files not backed-up because of sharing conflicts (networked systems only).

3
: Terminated by user.

4
: Terminated due to error.

These exit codes are used with the batch processing, IF ERRORLEVEL command, for appropriate responses to the various codes.

Examples:

To back-up all the files in the current directory on Drive C, onto the disk in Drive A, and overwriting any files that are already there, use the command line:

```
BACKUP C: A:
```

(assuming that the external command BACKUP is stored in the default directory or in a directory that MS-DOS automatically searches)

The back-up copies are all stored in the Root directory on Drive A.

To back-up all the files in the DIR1 directory on Drive C (a subdirectory of the Root directory) onto the disk in Drive A, adding to the files already in the Root directory on Drive A, use the command line:

```
BACKUP C: DIR1 A:/A
```

To back-up all the files on the hard disk - Drive C (ie. not just the files in the Root directory but those in all the subdirectories as well) onto Drive A, use the command line:

```
BACKUP C: A:/S
```

These files are all stored in the Root directory on Drive A. It is most likely that this back-up operation will require more than one diskette to complete the process. BACKUP will prompt you for additional disketts as required. It is a good idea to have a batch of newly formatted diskettes at hand. Each successive diskette of a series should be labelled sequentially (1, 2, 3, ...) with the back-up date and other details for easy reference when restoring.

Use the /L option in your command line to keep a record of the files which are copied onto the back-up disk, and the date the copies were made. For example, to record that you backed-up the files in the current directory on Drive C onto the disk in Drive A, you might use the command line:

```
BACKUP C: A:/L
```

The record would then be kept on the back-up disk in a file called BACKUP.LOG. If, however, you wanted to give this file a special name - for example, RECORDS.LOG - you should change the command line to:

```
BACKUP C: A:/LRECORDS.LOG
```

# APPENDIX 18 THE RESTORE COMMAND.

The RESTORE command restores files that were backed up using the BACKUP command.

Syntax:

```
RESTORE drive 1: [drive2:][ ][path ][filename.filetype]
[/S][/P][/B:date][/A:date][/E:time][/L:time][/M][/N]
```

drive1 contains the backed-up files.
drive2 is the target drive.

The RESTORE command copies files from backup disks onto working disks (eg. your hard disk), usually replacing the existing versions of the same file on your working disk.

The option switches control:

- IF ONLY files that have been changed since they were last backed-up are restored.
- IF ONLY files that have been changed since a particular date and/or time are restored.
- IF ONLY files that have been deleted from the working disk (ie. files of which there is no existing version on this disk) are restored.
- IF HIDDEN and READ-ONLY files are only restored with your knowledge.

At the end of the process, RESTORE sets an exit code to record how it finished. This can be used in an IF command.

Option Switches:

/S
: Restores subdirectories also.

/P
: Prompt for permission to restore any files that are READ-ONLY or that have changed since the last backup.

/M
: Restores only those files modified since the last backup.

/N
: Restores only those files that no longer exist on the target disk.

/A:date
: Restores only those files last modified on or after the date.

/L:time
: Restores only those files last modified at or after the time.

/B:date
: Restores only those files last modified on or before the date.

/E:time
: Restores only those files last modified at or before the time.

Notes:

Dates are in the form: dd-mm-yy

Times are in the form: hh[:mm[:ss]]

RESTORE cannot restore the system files, IO.SYS and MSDOS.SYS. Use the SYS command to restore these files.

The MS-DOS 3.3 RESTORE command will restore files backed up with earlier MS-DOS versions as well as the MS-DOS 3.3 BACKUP command.

Once MS-DOS has restored the file, use the DIR or TYPE command to make sure that the file was restored properly.

EXIT CODES:

Upon completion, RESTORE returns one of the following exit codes:

0
: Normal completion.

1
: No files were found to restore.

3

Terminated by user.

4 Terminated due to error.

These exit codes are used with the batch processing, IF ERRORLEVEL command, for appropriate responses to the various codes.

Examples:

To restore all the files matching the filename template MY*.TXT to the DIR1 directory (a subdirectory of the Root directory) on your hard disk (the default drive) from the backup disk in Drive A, use the command line:

```
RESTORE A: DIR1 MY*.TXT
```

(this assumes that the external command RESTORE is either in the default directory or in a directory on the search path).

If you want to restore only those files which no longer exist on your working disk, make your command line:

```
RESTORE A: DIR1 MY*.TXT/N
```

To restore all the files from the backup disk in Drive A onto your hard disk (the default drive) and to restore the structure of directories as well, use the command line:

```
RESTORE A: *.*/S
```

This will replace every file on your hard disk - including files that are marked READ ONLY. If you don't necessarily want all files to be replaced, you should use the /P option. Your command line then becomes:

```
RESTORE A: *.*/S/P
```

Then, when a file which has changed since the last backup or a READ-ONLY file is about to be replaced, you will see a message like:

```
restore filename.filetype (Y/N)?
```

Type N if you don't want this file replaced.

# INDEX

1234h 52, 101
24 Hour Flag 90, 101
32.768 Khz Oscillator 15
160 x 100 Graphics 21
320 x 200 Graphics Mode 22
640 x 200 Graphics Mode 23
8087-2 Numeric Data Processor 1, 72

Absolute Key Token 62, 98
ACE (8250) Registers 107
Active Video Display page 100
AMSTRAD PC 640K (Vv.i) message 104
Asynchronous Communications Element test 54
Attribute Byte (Color) 20
Attribute Byte (Monochrome) 24

Baud Rate Selector 80
Break Byte 100
BREAK Command 137
BUFFERS Command 138

Central Processing Unit (CPU) 1
Centronics Compatible Port 15
Centronics Interface Connector 37
CGA Color Select Register 26
CGA CRTC Display Addressing 27
CGA Mode 6845 CRTC Emulation 67
CGA Mode Control Register 25
CGA Status Register 27
Character set select 141
Colour Alpha Display 19
Colour Graphics Display 21
COMMAND.COM 179
Counter 1 initialisation 10
Country Dependent Information 144
CRC Error 73
CRTC Initialisation Data 94
Ctrl + [Keys]: Key Actions 61
Current Video Color Select byte 100
Current Video Mode Byte 99
Current Video Mode Control Byte 100
Cursor Address Buffer 100
Cursor Start/End Scan Byte 100

DEBUG Utility Program 180
Default Display Mode (DDM) 12, 46, 71
DEVICE Command 138
Diagnostic Mode 18, 106
Direct memory Access (DMA) 5
  Controller test 54
  Controller (8237A-4) Registers 110
  Initialization 7
  Over 64K boundary (Disk I/O Error) 72, 77
  Overrun Error 73
  Page Registers 6
Disk Bootstrap Interrupt 89
Disk Controller Error 72, 78
Disk I/O Interrupt 72, 77
Disk Parameter Table 95
Disk Write Protect Error 72
Drive Motor Flag 99
Drive Motor Timeout Counter 99
Drive Not Ready
Drive Restore Flag
Drive Status Byte 99
DRIVER.SYS 139
DRIVPARM Command 141
Dynamic RAM Refresh 9

ECC Error 78
End of Track 95
Error: External ROM checksum incorrect 105
Error: Faulty ... 105
EXE (Run File) 164, 168
EXE2BIN Utility Program 202
EXIT Command 204
Expansion Bus I/O Channels 5
Expansion Interface Connectors 41
External Cluster Controllers 5
External Ticker Interrupt 93
Extra RAM size 97

Fatal ROS Messages 105
FCBS Command 142
FDC (uPD765A) registers 121
FDC Command Codes 124
FDC Hardware Conditions 34
FDC Results Buffer 99
FILES Command 142
Filler Byte 95
Firmware 47
Floppy Disk Controller 34
Floppy Disk I/O Interrupt 72
Floppy Disk Interface test 55

Font Selection Links 141
Format Bad Track 77
Format Drive 77
Format Track 76, 77

Gap Length 95
Get Cursor Address 65
Get Key Token (Keyboard I/O Interrupt) 83
Get Light Pen Address 66
Get RTC Date 91
Get RTC Time 91
Get System Clock 90
Get Video Parameters 71

Hard Disk Call parameters and registers 77
Hard Disk Drive Count 101
Hard Disk I/O Interrupt 72
Hard Disk ROM 2, 77
Head Load Delay 95
Head Settling Delay 95
High Resolution (640 x 350) Graphics Mode 23
Horizontal Total 94

Initialization Stack 96
Initialize disk sub-system 73, 77
Initialize printer port 88
Initialize Serial Port 80
Insert a System disk into drive A 89, 104
Installed RAM Size 13
Internal Display Adapter 18
Interrupt Controller Initialisation 8
Interrupt Levels 8, 114
Interrupt Vector Initialisation 49
Interrupt Vectors 49
  Interrupt 2: Parity Error (NMI) 55
  Interrupt 5: Print Screen 56
  Interrupt 8: System Clock Interrupt 56
  Interrupt 9: Keyboard Interrupt 57
  Interrupt 14 (0Eh): Floppy Disk Controller 62
  Interrupt 16 (10h): Video I/O 62
  Interrupt 17 (11h): System Configuration 71
  Interrupt 18 (12h): Memory Size 72
  Interrupt 19 (13h): Disk I/O 72
  Interrupt 20 (14h): Serial I/O 79
  Interrupt 22 (16h): Keyboard I/O 83
  Interrupt 23 (17h): Printer I/O 87
  Interrupt 24 (18h): System Restart 88
  Interrupt 25 (19h): Disk Bootstrap 89
  Interrupt 26 (1Ah): System Clock and RTC 89
  Interrupt 27 (1Bh): Keyboard Break Interrupt 93
  Interrupt 28 (1Ch): External Ticker Interrupt 93

  Interrupt 29 (1Dh): Video Parameter Table 93
  Interrupt 30 (1Eh): Disk Parameter Table 95
  Interrupt 31 (1Fh): Video Matrix Table 95
Introduction 1
Invalid FDC Opcodes 136
IRGB Color Selection 19, 20

Keycodes 39, 57, 158
Key States 86, 97
Key Toggles 86, 97
Key Token Buffer End Address 102
Key Token Buffer Start Address 102
Key Token Buffer 98
Key Tokens 57 - 61
Keyboard and Key Codes 158
Keyboard Break Interrupt 93
Keyboard Buffer status 84, 86
Keyboard I/O Interrupt 83
Keyboard Interface 38
Keyboard Interface test 55
Keyboard Interrupt 57
Keyboard Layouts 159
  Danish 162
  French 160
  German 160
  Norwegian 163
  Portugese 163
  Spanish 161
  Swedish 162
  UK 159
  USA 159
Keyboard to Main Board Interface 38
Kill RTC Alarm 92

Language Links 106
LASTDRIVE Command 143
LIB (Library Files) 167
Light Pen Connector 40
Linker Program 164
Logical Printer Device Base I/O Address 96
Logical Printer Device Timeout Buffer 101
Logical Serial Device Base I/O Address 96
Logical Serial Device Timeout Buffer 101
Low Resolution (320 x 200) Graphics 21

Main Board I/O Channels 3
Main Board to Keyboard Interface 38
MAP (Link Map File) 167
MDA CRTC Initialisation 32
MDA Mode Register 31
MDA Status Register 32

Medium Resolution (320 x 200) Graphics Mode 22
Memory Layout 2
Memory Size Interrupt 72
Missing Address mark 73, 78
Mode Control Register 25
Monochrome Alpha Display 23
Motor off timeout 95
Motor on Delay 95
MS-DOS System Configuration 137
MS-LINK 164
Maths Co-processor (NDP) 1, 72

NMI Disable 11
NMI Mask Control 9
Non-Fatal ROS Messages 104
Non-Volatile RAM (NVR) 103, 117
Null Modem Cable 151
Number of Printers Attached 72
Number of Serial Interfaces Attached 72

OBJ (Object File) 167
Optional Games Adapter 72

Paper Out 87
Parallel Printer Interface 36
Parallel Printer Port 15
Parity Error (NMI) 55
Parity Error Disable 11
PCC Power Usage 155
PIC (8259A-2) Command Words 112
PIT (8253) Registers 115
Pixel to Bit Mapping (320 Res) 22
Pixel to Bit Mapping (640 Res) 23
Please fit new batteries 104
Please Wait ..... 52, 104
Port A - Status-1 Input/Keyboard Code 12
Port B - System Control 11
Port C - Status-2 Input 13
Power Connector 45
Power-Up Initialisation and Self Test 47
Power-Up Self Tests 52
Power-Up Test Methods 53
Power-Up Test Procedure 52
Print Screen Status Byte 130
Print Screen 56
Printer Acknowledge 87
Printer Connector 37
Printer Control Latch 16
Printer Data Latch 16
Printer I/O Interrupt 87
Printer Idle 87

Printer Lead Wiring 153
Printer Parallel Port test 54
Printer Selected 87
Printer Status Channel 17
Processor Memory Usage 2, 106
Programmable Interrupt Controller test 55
Programmable Interval Timer test 54
Programmable Interval Timers 9
Programmable Peripheral Interface test 54
RAM0 - RAM4 139
RAMDRIVE.SYS 13
Read Character and Attributes 67
Read Pixel (Video Int 16) 70
Read Sectors 74, 78
Read Serial Port 81
Real Time Clock test 54
Real Time Clock 15
Record Not Found (Disk I/O Error) 73
RECOVER Utility Program 205
Reference Information 106
Resident Operating System (ROS) ROM 2
Return Disk I/O Status 74, 78
Return printer port status 87
RGB Colour 19
ROM Character Set
  English, Norwegian 156
  Portugese, Greek 156
ROM Firmware Interrupts 50
ROS Checksum Test 53
ROS Interrupt 16 (10h): Video I/O 62
ROS Messages 104
RS232 Connections 145
RS232C Asynchronous Serial Port 35
RTC (HD146818) Registers 116

Scroll Screen Down 67
Scroll Screen Up 66
Second Floppy Disk Drive 14
Sector Size 95
Seek Error 73, 78
Send character to the printer port 87
Serial Baud Rate Settings 80
Serial Channel Interface 35
Serial Channel Pin Arrangement 36
Serial Clock and Serial Data 38
Serial Connector 36
Serial I/O Interrupt 79
Serial Parity Settings 80
Serial Port Status 82
Serial Stop Bit Settings 80

Set Cursor Address 65
Set Cursor Size 65
Set Display Page 66
Set RTC alarm 92
Set RTC Date 91
Set RTC Time 90
Set System Clock 90
Set Video Mode 63
SHARE Utility Program 206
SHELL Command 143
Size of Ram Disk 140
Soft Reset 61
Speaker Drive & Modulate 11
Special Key Actions 61
STACKS Command 143
Status-1 Input/Keyboard Code 12
Status-2 Input 13
Step rate 95
System Clock and RTC Interrupt 89
System Clock Interrupt 56
System Clock Long Word 90, 101
System Commands Processor 179
System Configuration Interrupt 71
System Configuration Word 96
System Control Port 11
System Interrupts 7
System Memory 2
System RAM test 54
System RAM Variables 96
System Reset Flag 53, 101
System Reset 14
System Restart Interrupt 88
System Status and Control 10

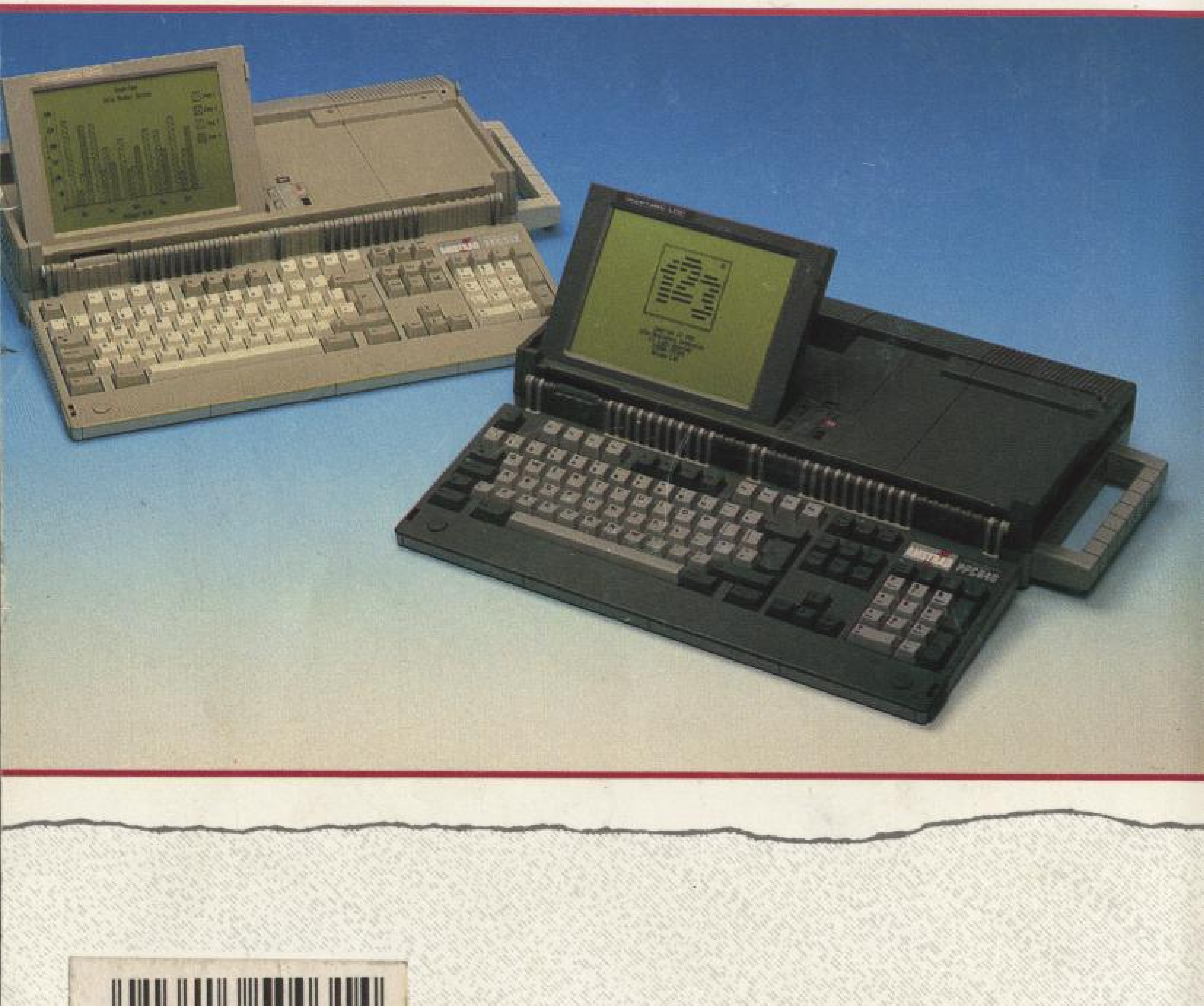
Then press any key. 89, 104
Time and Date of last usage 103, 104
Time and Date parameters 103, 116
Timer Configuration 10
Total RAM Size 97

Verify Sectors 75, 77
Video Connector 44
Video Display Buffer Size Word 99
Video Display Buffer Start Address 99
Video I/O Address Word 100
Video I/O Interrupt 62
Video Matrix Table 95
Video Output Pins 44
Video Parameter Table 95
Visible Video Columns Byte 99

VM.TMP 168

Wait States 1
Write Character and Attributes 68
Write Character Only 80
Write Color Palette 69
Write in TTY emulation mode 70
Write Pixel 70
Write Sectors 75, 78
Write Serial Port 80
Write System Status 14

Appendices Site Index Back Cover



5 013413 500314

**SOFT 50031**