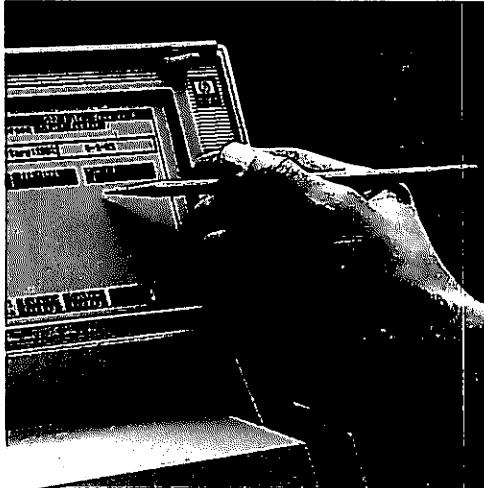


# HEWLETT-PACKARD

Technical  
Reference Manual



Portable PLUS

# **Portable PLUS**

---

## **Technical Reference Manual**

---



**HEWLETT  
PACKARD**

**Edition 1 August 1985**

## **Notice**

**Hewlett-Packard makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranty of merchantability and fitness for a particular purpose. Hewlett-Packard shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance, or use of this material.**

**Hewlett-Packard assumes no responsibility for the use or reliability of its software on equipment that is not furnished by Hewlett-Packard.**

**(c) Copyright 1985, Hewlett-Packard Company.**

This document contains proprietary information, which is protected by copyright. All rights are reserved. No part of this document may be photocopied, reproduced, or translated to another language without the prior written consent of Hewlett-Packard Company. The information contained in this document is subject to change without notice.

**Restricted Rights Legend.** Use, duplication, or disclosure by the Government is subject to restrictions as set forth in paragraph (b)(3)(B) of the Rights in Technical Data and Software clause in DAR 7-104.9(a).

**MS<sup>TM</sup>** is a U.S. trademark of Microsoft Corporation.

**1-2-3<sup>TM</sup>** and **Lotus<sup>TM</sup>** are U.S. trademarks of Lotus Development Corporation.

**Portable Computer Division  
1000 N.E. Circle Blvd.  
Corvallis, OR 97330, U.S.A.**

# Contents

---

Chapter		Page
<b>1</b>	<b>Overview</b>	
1.1	About This Manual .....	1-1
1.2	Options for Accessing the System .....	1-2
1.2.1	Accessing the Display .....	1-2
1.2.2	Accessing Communications Devices .....	1-3
1.3	References .....	1-4
<b>2</b>	<b>Electrical Design</b>	
2.1	Introduction .....	2-1
2.2	Memory Map .....	2-3
2.3	Operating Modes .....	2-4
2.3.1	Sleep Mode .....	2-5
2.3.2	Stop Mode .....	2-7
2.4	Mainframe Hardware .....	2-7
2.4.1	CPU .....	2-9
2.4.2	Clocking .....	2-9
2.4.3	Ready Circuit .....	2-9
2.4.4	PPU - Peripheral-Processor Unit .....	2-9
2.4.5	Keyboard Interface .....	2-10
2.4.6	Power Supply .....	2-10
2.4.7	Memory Board .....	2-10
2.4.8	Configuration EPROM .....	2-10
2.5	Serial Interface .....	2-13
2.6	HP-IL Interface .....	2-19
2.7	Recharger Interface .....	2-19
2.8	Video Connector .....	2-23
2.9	Modem Connector .....	2-25
2.10	Plug-In Ports .....	2-28
2.10.1	Generic Module Description .....	2-28
2.10.2	Electrical Specifications .....	2-28
2.10.3	Architectural Requirements .....	2-41

<b>3</b>	<b>Mechanical Design</b>	
3.1	Introduction .....	3-1
3.2	Mainframe .....	3-2
3.3	Modem .....	3-4
3.4	Plug-In Ports .....	3-6
<b>4</b>	<b>Resetting the Portable PLUS</b>	
4.1	Introduction .....	4-1
4.2	Reset Options .....	4-2
4.2.1	Reset via <b>(Shift)</b> <b>(CTRL)</b> <b>(Break)</b> .....	4-2
4.2.2	Reset via <b>(Shift)</b> <b>(CTRL)</b> <b>Extend char</b> <b>(Break)</b> .....	4-2
4.2.3	Reset via <b>(①)</b> .....	4-2
4.2.4	Reset via the Reset Button .....	4-3
4.3	Re-Boot Screen .....	4-3
4.3.1	Memory Lost Message .....	4-3
4.3.2	Standard Re-Boot Display .....	4-4
<b>5</b>	<b>BIOS Interrupts</b>	
5.1	Introduction .....	5-1
5.2	Print Screen Interrupt (Int 5h) .....	5-5
5.3	Video I/O Interrupt (Int 10h) .....	5-5
5.4	Equipment Check Interrupt (Int 11h) .....	5-12
5.5	Memory Interrupt (Int 12h) .....	5-13
5.6	Communications Interrupt (Int 14h) .....	5-13
5.7	Keyboard I/O Interrupt (Int 16h) .....	5-17
5.8	Print Byte Interrupt (Int 17h) .....	5-19
5.9	Reboot Interrupt (Int 19h) .....	5-20
5.10	Time Of Day Interrupt (Int 1Ah) .....	5-20
5.11	Keyboard Break Interrupt (Int 1Bh) .....	5-21
5.12	Timer Tick Interrupt (Int 1Ch) .....	5-22
5.13	Graphics Character Extensions (Int 1Fh) .....	5-23
5.14	Modem Transmit Interrupt (Int 40h) .....	5-24
5.15	Modem Ring/Carrier Interrupt (Int 42h) .....	5-24
5.16	Timer 2 Interrupt (Int 43h) .....	5-24
5.17	Plug-in 1 Interrupt (Int 44h) .....	5-25
5.18	Plug-in 2 Interrupt (Int 45h) .....	5-25
5.19	PPU Alarm Interrupt (Int 46h) .....	5-25
5.20	Death/Battery Cutoff Interrupt (Int 47h) .....	5-26
5.21	Keyboard Interrupt (Int 49h) .....	5-27
5.22	Serial Transmit Interrupt (Int 4Ah) .....	5-28
5.23	Serial Ring/Carrier Interrupt (Int 4Bh) .....	5-28
5.24	HP-IL IRQ Interrupt (Int 4Ch) .....	5-28

5.25	Low Battery Interrupt (Int 4Dh) . . . . .	5-29
5.26	Modem Input Interrupt (Int 4Eh) . . . . .	5-29
5.27	Serial Input Interrupt (Int 4Fh) . . . . .	5-30
5.28	System Services Interrupt (Int 50h) . . . . .	5-31
5.29	Modifier Key Interrupt (Int 52h) . . . . .	5-64
5.30	Print Key Interrupt (Int 53h) . . . . .	5-65
5.31	HP-IL Primitives Interrupt (Int 54h) . . . . .	5-65
5.32	Sleep Interrupt (Int 55h) . . . . .	5-80
5.33	Menu Key Interrupt (Int 56h) . . . . .	5-81
5.34	System Key Interrupt (Int 57h) . . . . .	5-82
5.35	Break Key Interrupt (Int 58h) . . . . .	5-83
5.36	Enable/Disable Ring Interrupt (Int 59h) . . . . .	5-84
5.37	AUX Expansion Interrupt (Int 5Dh) . . . . .	5-85
5.38	CON Expansion Interrupt (Int 5Eh) . . . . .	5-87
5.39	Fast Video Interrupt (Int 5Fh) . . . . .	5-94
5.39.1	Fast Alpha . . . . .	5-94
5.39.2	Fast Graphics . . . . .	5-113

## **6 Built-In Device Drivers**

6.1	Introduction . . . . .	6-1
6.1.1	Serial Operation . . . . .	6-2
6.1.2	Modem Operation . . . . .	6-7
6.2	AUX, COM1, COM2, COM3, and 82164A Devices . . . . .	6-9
6.3	NUL Device . . . . .	6-21
6.4	CLOCK Device . . . . .	6-22
6.5	LPT1, LPT2, LST, PLT, and PRN Devices . . . . .	6-22
6.6	CONsole Driver . . . . .	6-23
6.6.1	CONsole Control Sequences . . . . .	6-24
6.6.2	Keyboard Operation . . . . .	6-45
6.6.3	CONsole I/O Control Functions . . . . .	6-50

## **7 Low-Level Hardware Interface**

7.1	Introduction . . . . .	7-1
7.2	I/O Memory Map . . . . .	7-1
7.3	Multi-Controllers . . . . .	7-3
7.3.1	Keyboard Interface . . . . .	7-3
7.3.2	Interval Timer . . . . .	7-4
7.3.3	Serial Port . . . . .	7-5
7.3.4	Multi-Purpose Port . . . . .	7-7
7.3.5	Registers - Overview . . . . .	7-7
7.3.6	Registers - Keyboard Function . . . . .	7-9
7.3.7	Registers - Serial Port . . . . .	7-12

7.3.8	Registers - Interval Timer .....	7-16
7.3.9	Registers - Multi-Purpose Port .....	7-19
7.4	HP-IL Controller .....	7-22
7.5	Display Controller .....	7-26
7.5.1	Display RAM Mapping - Graphics Mode .....	7-26
7.5.2	Display RAM Mapping - Alpha Mode .....	7-29
7.5.3	Alpha Attribute Bits .....	7-33
7.5.4	Alpha Cursors .....	7-33
7.5.5	Registers .....	7-34
7.5.6	Softkey Menu Display .....	7-37
<b>8</b>	<b>Memory Management</b>	
8.1	Introduction .....	8-1
8.2	Edisc .....	8-3
8.3	ROM Disc .....	8-7
8.4	Summary of ROM Disc Access .....	8-14
<b>9</b>	<b>Plug-In ROM Design</b>	
9.1	Introduction .....	9-1
9.2	Plug-In ROM Format .....	9-2
9.3	ROM-Executable Code .....	9-6
9.4	ROM Boot Code .....	9-7
9.5	Constraints on Plug-In ROM Software .....	9-7
9.6	PAM Interface to Plug-In ROMs .....	9-8
<b>10</b>	<b>PAM - The Personal Application Manager</b>	
10.1	Power-Up Sequence .....	10-1
10.2	The PAM Environment .....	10-1
10.2.1	PAM and AUTOEXEC.BAT Files .....	10-2
10.2.2	PAM Internal State .....	10-3
10.3	PAM And Application Programs .....	10-4
10.3.1	Installing Applications in PAM .....	10-4
10.3.2	The "DOS Commands" Application .....	10-4
10.3.3	PAM Execution of a Program .....	10-5
10.4	The PAM Configurations .....	10-6
10.4.1	The System Configuration .....	10-6
10.4.2	Main Memory and Edisc .....	10-6
10.4.3	External Disc Drives .....	10-7
10.4.4	Disc Write Verify .....	10-7
10.4.5	Power-Save Mode .....	10-7
10.4.6	Determining a Reasonable Status Limit Value .....	10-8
10.4.7	Display Timeout .....	10-9

10.4.8	Cursor Type .....	10-10
10.4.9	Console Mode .....	10-10
10.4.10	Tone Duration .....	10-11
10.4.11	Plotter Interface .....	10-11
10.4.12	Printer Interface .....	10-11
10.4.13	Printer Mode .....	10-11
10.4.14	Printer Pitch, Line Spacing, and Skip Perforation .....	10-12
10.4.15	Datacom Interface .....	10-12
10.4.16	The Datacom Configuration .....	10-12
10.4.17	The Time and Date Configuration .....	10-15
10.5	PAM And Alarms .....	10-16
10.5.1	The PAM.ALM File .....	10-16
10.5.2	When an Alarm Occurs .....	10-16
10.6	Autoanswering: PAM and Ring Interrupts .....	10-17
10.7	The Battery Fuel Gauge .....	10-18
10.8	PAM Help Facility .....	10-19
10.8.1	Installing PAMHELP.COM .....	10-19
10.8.2	After PAMHELP.COM Is Installed .....	10-19
10.9	Bypassing PAM With COMMAND.COM .....	10-21
<b>11</b>	<b>Boot Sequence Options</b>	
11.1	Introduction .....	11-1
11.2	Boot Sequence .....	11-2
11.2.1	Built-In Diagnostics .....	11-2
11.2.2	Recover From Sleep .....	11-2
11.2.3	ROM Slot 7 Boot Code Before Changing RAM .....	11-2
11.2.4	Config EPROM Boot Code Before Changing RAM .....	11-3
11.2.5	ROM Slot 7 Boot Code After Some Initialization .....	11-3
11.2.6	Boot Code From the Config EPROM After Initialization .....	11-4
11.2.7	Boot Using the CONFIG.SYS on the ROM in ROM Slot 7 .....	11-4
11.2.8	Boot Using the CONFIG.SYS on the Default Drive .....	11-4
11.2.9	PAM Executes AUTOEXEC.BAT From ROM Slot 7 .....	11-5
11.2.10	PAM Executes AUTOEXEC.BAT From Drive A: .....	11-5
11.3	The CONFIG.SYS File .....	11-5
<b>12</b>	<b>Modem Interface</b>	
12.1	Overview .....	12-1
12.1.1	Command Mode and Data Mode .....	12-1
12.1.2	Commands .....	12-2
12.1.3	Baud Rate Selection .....	12-3
12.1.4	Transmission Settings .....	12-4
12.1.5	Auto-Answering .....	12-4

12.1.6	Default State of the Modem .....	12-5
12.2	Modem Commands .....	12-6
12.3	Dialing .....	12-12
12.4	Modem Responses .....	12-13
12.5	S-Register Description .....	12-15
12.6	Hayes Compatibility .....	12-20
12.7	Special Considerations for Programmatic Control .....	12-22
12.7.1	Modem Power-On Problem .....	12-22
12.7.2	Ignores Characters While Responding .....	12-22
12.7.3	Can't Dial Out While Receiving Ring .....	12-23
12.7.4	Spurious Extra Characters Generated .....	12-23
12.7.5	Spurious Interrupts at Power-Up .....	12-23
12.8	Directly Connecting Two Modems .....	12-23

## 13 Keyboards and Keycodes

### A Comparisons With Other Computers

A.1	Comparison With the HP 110 .....	A-1
A.2	Comparison With the HP 150 .....	A-2
A.3	Comparison With the IBM PC .....	A-3
A.3.1	Video Interrupt (Int 10h) .....	A-3
A.3.2	Equipment Check Interrupt (Int 11h) .....	A-5
A.3.3	Diskette/Disc Interrupt (Int 13h) .....	A-5
A.3.4	Communications Interrupt (Int 14h) .....	A-5
A.3.5	Cassette Interrupt (Int 15h) .....	A-6
A.3.6	Keyboard Interrupt (Int 16h) .....	A-6
A.3.7	Printer Interrupt (Int 17h) .....	A-6
A.3.8	Re-Boot Interrupt (Int 19h) .....	A-6
A.3.9	Time-of-Day Interrupt (Int 1Ah) .....	A-7
A.3.10	Keyboard Break Interrupt (Int 1Bh) .....	A-7

### B Schematic Diagrams

### C Assembler Listing for Configuration EPROM

### D Character Sets

### E Using TERM in Batch Files

<b>F</b>	<b>Mass Storage</b>	
F.1	Disc Drive Options .....	F-1
F.1.1	Built-In Disc Drives .....	F-1
F.1.2	HP 9114A HP-IL Disc Drive .....	F-1
F.1.3	HP-IB Disc Drives .....	F-2
F.1.4	Portable PLUS-Desktop Link .....	F-3
F.2	Media Compatibility .....	F-4
F.2.1	Reading Other Discs on the Portable PLUS .....	F-4
F.2.2	Reading Portable PLUS Discs on Other Computers .....	F-4
<b>G</b>	<b>Configuring Serial Printers</b>	
G.1	Introduction .....	G-1
G.2	The HP 2225D ThinkJet Printer .....	G-1
G.3	The HP 2601A Printer .....	G-3
G.4	The HP 2686A LaserJet Printer .....	G-5
G.5	The IDS-560 Impact Printer - The Paper Tiger .....	G-6
G.6	The NEC Spinwriter 3510 .....	G-8
G.7	The Xerox 610C1 Memorywriter .....	G-10
G.8	The Xerox 625C Memorywriter .....	G-12
<b>H</b>	<b>Portable PLUS-Desktop Link</b>	
H.1	Portable PLUS to HP 150 .....	H-2
H.2	Portable PLUS to IBM PC/XT .....	H-3
H.3	Portable PLUS to IBM AT .....	H-4
H.4	Portable PLUS to Portable (or Portable PLUS) .....	H-4
<b>I</b>	<b>Parts</b>	
<b>J</b>	<b>Escape Sequence Summary</b>	
<b>K</b>	<b>Software Module Configurations</b>	
K.1	Overview .....	K-1
K.2	Plug-In ROMs and EPROMS .....	K-3
K.3	Detailed Description .....	K-5
K.3.1	ROM/EPROM Organization Options .....	K-5
K.3.2	Jumper and Socket Labeling .....	K-6
K.3.3	Jumpers and Socket Groups .....	K-7
K.3.4	Configuration of the Small Group .....	K-7
K.3.5	Configuration of the Large Group .....	K-8

# Illustrations

---

Figure		Page
2-1	System Address Space .....	2-3
2-2	I/O Address Space .....	2-4
2-3	Portable PLUS Block Diagram .....	2-8
2-4	Configuration EPROM Addressing .....	2-12
2-5	Serial Connector .....	2-14
2-6	Modem Cable .....	2-18
2-7	Printer Cable .....	2-18
2-8	Automotive Recharger Schematic Diagram .....	2-22
2-9	Plug-In Connector .....	2-33
2-10	Plug-In Bus Read Cycle - Lower Memory .....	2-34
2-11	Plug-In Bus Write Cycle - Lower Memory .....	2-35
2-12	Plug-In Bus Read Cycle - I/O and Upper Memory .....	2-36
2-13	Plug-In Bus Write Cycle - I/O and Upper Memory .....	2-37
2-14	Plug-In Bus Interrupt Acknowledge Cycle .....	2-38
2-15	Plug-In Module Registers .....	2-43
3-1	Mainframe Dimensions .....	3-3
3-2	Modem PC Board .....	3-5
3-3	Plug-In PC Board .....	3-7
5-1	Interrupt 10h Attribute Byte .....	5-6
5-2	Fast Alpha Structures .....	5-95
5-3	Display Attribute Byte .....	5-96
5-4	Initial Font Load .....	5-97
5-5	Font Formats .....	5-98
6-1	Serial Interface Timing - Sheet 1 .....	6-4
6-2	Serial Interface Timing - Sheet 2 .....	6-5
6-3	Serial Interface Timing - Sheet 3 .....	6-6
6-4	Keyboard Scancodes .....	6-48
7-1	I/O Address Space .....	7-2
7-2	Display RAM Mapping - Graphics Mode .....	7-28
7-3	Display RAM Mapping - Alpha Mode .....	7-32
8-1	RAM Organization .....	8-2
8-2	ROM Disc .....	8-8

8-3	ROM Disc FAT .....	8-11
8-4	ROM Disc Root Directory .....	8-12
8-5	ROM Disc Fixed Subdirectory Files .....	8-13
8-6	ROM Disc Plug-In File Data .....	8-14
9-1	Plug-In ROM Format .....	9-3
13-1	English (U.S.) Keyboard .....	13-8
13-2	English (U.K.) Keyboard .....	13-11
13-3	French Keyboard .....	13-14
13-4	Belgian Keyboard .....	13-15
13-5	German Keyboard .....	13-18
13-6	Italian Keyboard .....	13-21
13-7	Dutch Keyboard .....	13-24
13-8	Swiss (German) Keyboard .....	13-27
13-9	Swiss (French) Keyboard .....	13-30
13-10	Danish Keyboard .....	13-33
13-11	Norwegian Keyboard .....	13-36
13-12	Swedish Keyboard .....	13-39
B-1	Motherboard PCA - Sheet 1 .....	B-2
B-2	Motherboard PCA - Sheet 2 .....	B-4
B-3	Motherboard PCA - Sheet 3 .....	B-6
B-4	Motherboard PCA - Sheet 4 .....	B-7
B-5	Memory Board PCA .....	B-8
B-6	Modem PCA .....	B-10
B-7	Software Drawer PCA .....	B-12
B-8	Memory Drawer PCA - Sheet 1 .....	B-14
B-9	Memory Drawer PCA - Sheet 2 .....	B-16
B-10	Memory Drawer Piggy-Back PCA .....	B-18
K-1	Pin Configuration for Plug-In ROM .....	K-4

# Tables

---

Table	Page
1-1 Options for Display Access .....	1-2
2-1 Serial Interface .....	2-14
2-2 Recharger DC Limits .....	2-20
2-3 Recharger Series Resistance Limits .....	2-21
2-4 Video Signals .....	2-24
2-5 Video Specifications .....	2-24
2-6 Modem Connector .....	2-25
2-7 Specifications for Modem Port .....	2-27
2-8 Signals for Plug-In Ports .....	2-29
2-9 Plug-In Bus Loading .....	2-39
2-10 Plug-In Power Loads .....	2-40
2-11 Voltage Levels for Plug-In Bus .....	2-40
2-12 Requirements for Plug-In Drivers .....	2-41
5-1 Hardware and BIOS Interrupts .....	5-2
5-2 Video I/O Interrupt 10h Functions .....	5-7
5-3 Communications Interrupt 14h Functions .....	5-14
5-4 Keyboard I/O Interrupt 16h Functions .....	5-17
5-5 Print Byte Interrupt 17h Functions .....	5-19
5-6 Time Of Day Interrupt 1Ah Functions .....	5-20
5-7 System Services Interrupt 50h Functions .....	5-31
5-8 System Services Int 50h Detailed Description .....	5-33
5-9 PPU Commands .....	5-54
5-10 HP-IL Primitives Interrupt 54h Functions .....	5-69
5-11 AUX Expansion Interrupt 5Dh Function .....	5-86
5-12 CON Expansion Interrupt 5Eh Functions .....	5-88
5-13 Fast Video Interrupt 5Fh Alpha Functions .....	5-99
5-14 Fast Video Interrupt 5Fh Graphics Functions .....	5-114
6-1 AUX I/O Control Commands .....	6-9
6-2 Control Characters .....	6-24
6-3 HP Two-Character Escape Sequences .....	6-26
6-4 HP Alpha Escape Sequences .....	6-31
6-5 HP Graphics Escape Sequences .....	6-36

6-6	ANSI Escape Sequences .....	6-41
6-7	CONsole Write I/O Control Functions .....	6-50
6-8	CONsole Read I/O Control Functions .....	6-51
7-1	Multi-Controller Registers .....	7-8
7-2	HP-IL Registers .....	7-22
7-3	Display Controller Registers .....	7-35
8-1	Edisc Sector 0h .....	8-5
8-2	ROM Disc Sector 0h .....	8-9
9-1	Plug-In ROM Sector 0h .....	9-4
10-1	PAM Internal State .....	10-3
10-2	PAMHELP.COM Parameters .....	10-20
12-1	Modem Commands .....	12-7
12-2	Modem Responses .....	12-14
12-3	Modem S-Registers .....	12-15
13-1	HP Mode Character Codes .....	13-3
13-2	Alternate Mode Character Codes .....	13-5
13-3	Common HP Mode Character Codes .....	13-6
13-4	Common Alternate Mode Character Codes .....	13-7
13-5	English (U.S.) HP Mode Character Codes .....	13-9
13-6	English (U.S.) Alternate Mode Character Codes .....	13-10
13-7	English (U.K.) HP Mode Character Codes .....	13-12
13-8	English (U.K.) Alternate Mode Character Codes .....	13-13
13-9	French/Belgian HP Mode Character Codes .....	13-16
13-10	French/Belgian Alternate Mode Character Codes .....	13-17
13-11	German HP Mode Character Codes .....	13-19
13-12	German Alternate Mode Character Codes .....	13-20
13-13	Italian HP Mode Character Codes .....	13-22
13-14	Italian Alternate Mode Character Codes .....	13-23
13-15	Dutch HP Mode Character Codes .....	13-25
13-16	Dutch Alternate Mode Character Codes .....	13-26
13-17	Swiss (German) HP Mode Character Codes .....	13-28
13-18	Swiss (German) Alternate Mode Character Codes .....	13-29
13-19	Swiss (French) HP Mode Character Codes .....	13-31
13-20	Swiss (French) Alternate Mode Character Codes .....	13-32
13-21	Danish HP Mode Character Codes .....	13-34
13-22	Danish Alternate Mode Character Codes .....	13-35
13-23	Norwegian HP Mode Character Codes .....	13-37
13-24	Norwegian Alternate Mode Character Codes .....	13-38
13-25	Swedish HP Mode Character Codes .....	13-40
13-26	Swedish Alternate Mode Character Codes .....	13-41
13-27	HP Mode Muted Character Codes .....	13-42
13-28	Alternate Mode Muted Character Codes .....	13-43

D-1	Roman8 Character Set .....	D-2
D-2	Line-Drawing/Math Character Sets .....	D-3
D-3	Alternate Character Set .....	D-4
D-4	Character Sets - Numeric Listing .....	D-5
G-1	ThinkJet Switch Settings .....	G-2
G-2	HP 2601A Switch Settings .....	G-4
G-3	IDS-560 Switch Settings .....	G-7
G-4	Spinwriter 3510 Switch Settings .....	G-9
G-5	6101C1 Memorywriter Option Settings .....	G-10
G-6	625C Memorywriter Option Settings .....	G-13
I-1	Portable PLUS Accessories .....	I-1
I-2	Custom HP Parts .....	I-3
I-3	Standard Parts .....	I-3
J-1	Control Characters .....	J-1
J-2	Two-Character Escape Sequence Summary .....	J-2
J-3	HP Alpha Escape Sequence Summary .....	J-3
J-4	HP Graphics Escape Sequence Summary .....	J-4
J-5	ANSI Escape Sequence Summary .....	J-5
K-1	Wire Jumper Connections for ROMs/EPROMs .....	K-2
K-2	Plug-In ROM Specifications .....	K-3
K-3	ROM/EPROM Organization Options .....	K-5

# 1

# Overview

---

## 1.1 About This Manual

This manual presents information that will help you to develop hardware and software that operate on the Portable PLUS computer. The manual contains information about:

### **Hardware**

- Electrical design.
- Mechanical design.

### **Software and Firmware:**

- Resetting the computer.
- BIOS interrupts.
- Built-in device drivers.
- Low-level hardware interface.
- Memory.
- Plug-in ROM design.
- PAM (Personal Applications Manager).
- Boot options.
- Modem interface.
- Keyboards and keycodes.

In addition, appendixes include additional reference information about the Portable PLUS.

Applications designed for the Portable PLUS may be designed to be compatible with other computers also. (Refer to appendix A for detailed information about compatibility.) The information in this manual will help you maximize compatibility.

## 1.2 Options for Accessing the System

The following topics describe various ways you can access certain features of the Portable PLUS. The method you choose will depend upon your particular application. The individual choices are described in different parts of this manual.

### 1.2.1 Accessing the Display

The Portable PLUS provides six ways to access the display. The methods are listed in table 1-1. They provide the programmer with options for satisfying the specific requirements of a program.

Table 1-1. Options for Display Access

Access Method	Speed	Power	Usability	Graphics
CON output via Int 21h	1	3	Simple	Full
CON output via Int 50h	2	3	Simple	Full
Video I/O Int 10h	3	2	Moderate	Limited
Fast Video Int 4Fh	4	4	Moderate	Full
Fast Write via Int 50h	5	1	Simple	None
Direct hardware access	6	5	Difficult	Primitive

The preferred, most commonly encountered, and most portable method of sending data to the display is CON output via the standard MS-DOS service interrupt, Int 21h. (Refer to "References" below.) Display control can be accomplished via a fairly complete set of HP and ANSI escape sequences. A subset of standard HP graphics escape sequences provides access to all major graphics functions.

Output via Int 50h, the System Services interrupt, offers about a 2.5-times speed improvement over Int 21h, but only by sacrificing output redirection, simultaneous

printer hardcopy (via ^P printer on/off toggle), display start/stop control (via ^S/^Q), and portability to other MS-DOS computers. (Refer to "System Services Interrupt" in chapter 5.)

The Video I/O interrupt, Int 10h, provides a subset of the IBM PC Video I/O functions. (Refer to "Video I/O Interrupt" in chapter 5.) Its compatibility limitations are due mainly to the 480x200 size of the LCD panel and other hardware differences between the Portable PLUS and the IBM PC.

Fast Video (Int 5Fh) functions provide a level of display control similar to Int 10h, but with much more functionality and flexibility. (Refer to "Fast Video Interrupt" in chapter 5.) This set of routines provides very low-level, window-oriented control of display memory in alpha mode, and a relatively full set of graphics manipulation routines in graphics mode.

Int 50h Fast Write is a special service function that can be used to force short messages onto the display without interfering with any other part of the system. (Refer to "System Services Interrupt" in chapter 5.) This is used within the BIOS, for example, to display the "Low Battery!" warning. Fast Write is a very low-level function that simply forces a specified attribute and string of characters into display RAM with no special processing or safety checks, and should be used only in similar immediate-display situations.

As with most computers, there is the option of "going straight to hardware." Performing your own display control is potentially the fastest, most efficient way of getting the screen to do what you want it to do, although such programming can easily become quite complex, potentially dangerous, and can possibly interfere with the normal display operation if mixed with calls to other BIOS-resident display functions--unless you take certain precautions. For the sake of safety and simplicity, applications should refrain from directly accessing system hardware. (Refer to "Display Controller" in chapter 7.)

## 1.2.2 Accessing Communications Devices

The Portable PLUS can address three communications devices through the system BIOS:

- The serial (RS-232) port.
- A SmartModem-Compatible 1200-BPS modem.
- An HP 82164A HP-IL/RS-232-C Interface.

Each of these devices is supported by its own device driver, and data can be transferred through them using any of the MS-DOS standard device operations. Various device parameters can be configured through a standard set of IOCTL commands. (Refer to "AUX, COM1, COM2, COM3, and 82164A Devices" in chapter 6.) Also, the MS-DOS AUX device can be "redirected" to address any of the three individual devices.

In addition, the built-in serial port and the modem can be accessed through the IBM PC-compatible software interrupt 14h. (Refer to "Communications Interrupt" in chapter 5.) All IBM PC functions are supported, although some of the status information returned by this interrupt must be interpreted differently due to hardware incompatibilities. Interrupt 14h gives much better performance than the MS-DOS device calls, but it can be cumbersome to use from high-level languages, and it is not supported on all Hewlett-Packard products.

For applications that must get as close as possible to the hardware, interrupt 5Dh is provided to allow access to each character as it comes into the communications port. (Refer to "AUX Expansion Interrupt" in chapter 5.) This enables a program to achieve the effect of taking over the hardware interrupt, but will not need to duplicate the function of the BIOS.

---

## 1.3 References

Although this manual describes the Portable PLUS in detail, you may want to consult additional references for other information. Owner's manuals describe how to operate the system. Other references provide information about standards that are implemented by the Portable PLUS.

- Hewlett-Packard Company. *Using the Portable PLUS*. HP part number 45711-90002, (c)1985.
- Hewlett-Packard Company. *HP 82983A 300/1200 BPS Modem Owner's Manual*. HP part number 82983-90001, (c)1985.
- Hewlett-Packard Company. *The HP-IL Interface Specification*. HP part number 82166-90017, (c)1982.
- Hewlett-Packard Company. *The HP-IL Integrated Circuit User's Manual*. HP part number 82166-90016, (c)1982.

- Kane, Gerry, et al. *The HP-IL System: An Introductory Guide to the Hewlett-Packard Interface Loop*. Osborne/McGraw-Hill, Berkeley, California, (c) 1982.
- Hewlett-Packard Company. HP 45419C Programmer's Tool Kit, which contains:
  - Series 100 Programmer's Reference Manual: Microsoft MS-DOS Programmer's Reference Manual*.
  - Series 100 Macro Assembler Manual: Microsoft Macro Assembler Manual*.



## 2

# Electrical Design

---

### 2.1 Introduction

The Portable PLUS computer features a 25-line liquid-crystal display (LCD), a 76-key full-size (3/4 throw) keyboard, 128K bytes of built-in RAM, 16K bytes of display RAM, 8K bytes of built-in configuration EPROM (expandable to 16K bytes), and 192K bytes of built-in ROM. HP-IL and serial interfaces are built in. A 1200-baud direct-connect modem and an external video interface are optional.

The CPU is a CMOS 80C86 that runs at 5.33 MHz. RAM cycle time is 748 ns, with ROM and I/O cycle times being 935 ns minimum. A secondary processor, a peripheral processor unit (PPU), provides power supply and modem control, and also functions as a real-time clock.

Listed below are the main specifications for the Portable PLUS.

Size:	13 inches wide, 10 inches deep, 3 inches thick.
Weight:	8.9 pounds (with modem and two empty drawers).
LCD:	25 lines by 80 characters, alpha mode. 6 dots wide by 8 dots high font size. 200 dots high by 480 dots wide, bit-mapped graphics mode.
Keyboard:	Full size, 76 keys, 3/4 throw, embedded numeric pad.
Speaker:	Piezo-electric
CPU:	80C86, 16-bit CMOS processor, 5.33 MHz.
Memory:	128K bytes RAM. 192K bytes ROM. 16K bytes display RAM. 8K bytes configuration EPROM (16K bytes optional).
I/O:	HP-IL. Serial (RS-232-C). 1200-baud direct-connect modem (optional).
Battery:	6-volt, 2.5 Amp Hour, three-cell, sealed, lead-acid.
Power Consumption:	100-175 mA ON/awake mode (typical) 285 uA sleep mode (typical)
Environment:	Operating temperature: 0° to 50°C. Storage temperature: -25° to 55°C.* RFI: FCC class B, VDE class B.
Humidity:	5 to 95 percent relative humidity.

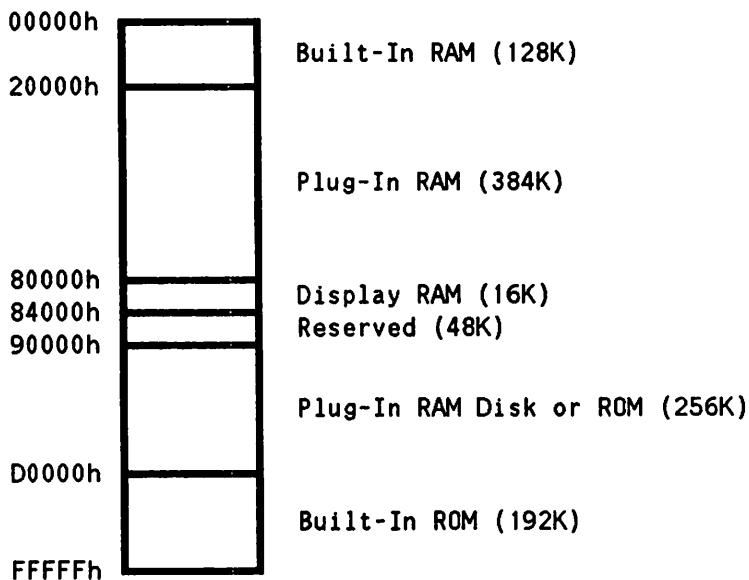
---

\* Exposure to temperature below -5C may cause temporary cosmetic blemishes in the display.

## 2.2 Memory Map

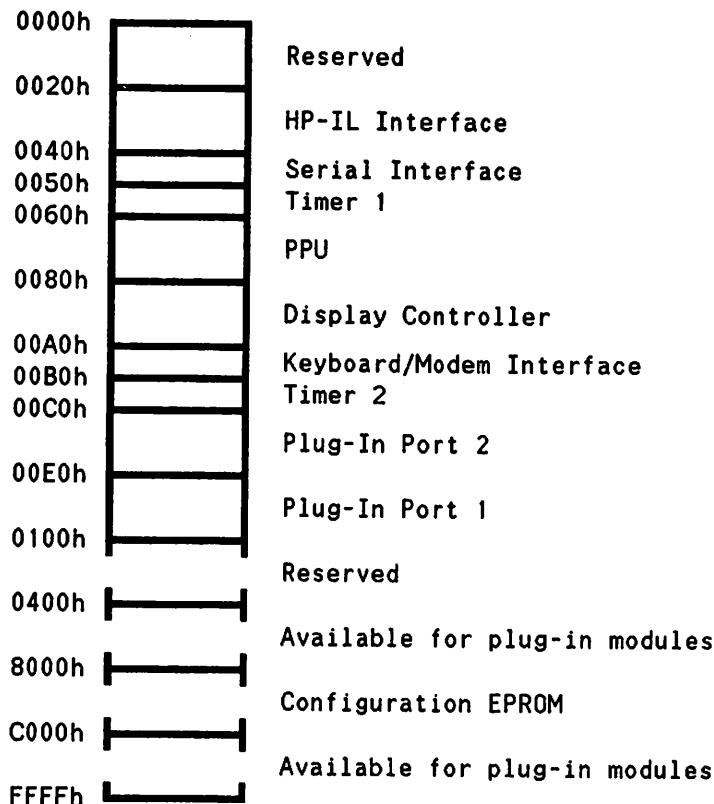
The 80C86 has a 1M-byte system memory address space and a 64K-byte I/O address space. High bytes have odd addresses; low bytes have even addresses. Memory space is allocated as shown in figure 2-1.

Figure 2-1. System Address Space



The mainframe uses I/O address space from 0000h to 03FFh, and from 8000h to BFFFh. Thus, the addresses from 0400h to 7FFFh and C000h to FFFFh are available for plug-in devices. I/O address space is represented in figure 2-2.

**Figure 2-2. I/O Address Space**



## 2.3 Operating Modes

The computer has several operating modes, which are controlled by a single chip micro-computer, known as the peripheral processor unit (PPU). The mainframe has two 5 volt supplies, known as VccS and VccDS. These supplies are switched on and off depending on the mode the mainframe is in.

- **Awake Mode:** Both 5 volt supplies are on. The display is turned on; the CPU is running or idle. RAM is preserved.
- **Sleep Mode:** VccS is off (the display is turned off; most circuits are powered down). VccDS is on but reduced to 3.25 volts nominal (RAM is preserved). The PPU remains in a low power state, monitoring system events. This mode is used to prolong battery life when the computer is not in use.
- **Stop Mode:** All internal power supplies are turned off. RAM memory is lost. All digital logic in the mainframe, plug-in cards and modem are turned off. This mode is *only* entered if a plug-in card is removed while the mainframe is in Awake Mode.

The following descriptions illustrate the system's behavior under various conditions.

### 2.3.1 Sleep Mode

**User initiated sleep mode to remove plug-in module:** System is awake; CPU running. User wants to change a plug-in drawer.

**Action Required:**

User must put system into its sleep mode (by pressing the "Off" softkey in PAM).

**System Behavior:**

PAM accepts the "Off" command and then issues a sleep command to the PPU.

User removes plug-in drawer. PPU senses the removal and waits until both plug-in drawers are present. During this wait, the PPU keeps updating the real time clock and the battery charge level.

When the user has plugged in both drawers, he must press the  key to wake the system up. When the system wakes up, power is applied to the mainframe and both plug-in drawers. The CPU is initially reset. When allowed to run, it reboots.



**Caution**

Any time a RAM plug-in module is removed its contents are lost. You must back up the electronic disc before removing or installing a RAM plug-in module.

**User initiated Sleep Mode to conserve battery:** System is awake; CPU running.  
User wants to put system to sleep in order to save power.

**Action Required:**

User presses the "Off" softkey in PAM.

**System Behavior:**

The PPU unpowers the CPU and the LCD display. Built-in RAM, display RAM, and plug-in RAM remain powered. The keyboard continues to be scanned.

System remains in sleep mode until one of the following occurs:

Any key is depressed.

The alarm time is reached.

A system interrupt is generated (for example, modem ring detected, serial ring detected, plug-in interrupt detected).

As the system wakes up, the CPU is initially reset. As it begins running, the BIOS determines that the system was in sleep mode (as opposed to a cold start) and restores the system to the state that existed before sleep mode was initiated.

**Timeout initiated Sleep Mode to conserve battery:** System is awake; CPU is running. The battery charger is not plugged in. The program running (MS-DOS, PAM, or an application) makes repeated calls to the keyboard driver's status without calling other I/O drivers. (This occurs when a program is waiting for keyboard input--refer to "Power-Save Mode" in chapter 10.)

**System Behavior:**

The BIOS monitors I/O driver "call" activity. If the keyboard driver's status is called often enough (with no calls to other I/O device drivers), after the timeout period has expired (set from PAM) the BIOS suspends operation of the current program, does some housekeeping, and then issues the sleep command to the PPU.

The PPU unpowers the CPU and the LCD display. Built-in RAM, display RAM, and plug-in RAM remain powered. The keyboard continues to be scanned.

System remains in sleep mode until one of the following occurs:

Any key is depressed.

The alarm time is reached.

A system interrupt is generated (for example, modem ring detected, serial ring detected, plug-in interrupt detected).

As the system wakes up, the CPU is initially reset. As it begins running, the BIOS determines that the system was in sleep mode (as opposed to a cold start) and restores the system to its previous state (the LCD displays the same information as before and the suspended program resumes where it left off).

### 2.3.2 Stop Mode

**Situation:** System is awake; CPU is running. User removes a plug-in drawer but forgets to put the computer in sleep mode.

#### System Behavior:

Removal of a plug-in drawer while the system is awake causes the system to enter stop mode. The power supply turns off completely, which turns off all mainframe digital logic and removes power to both plug-in ports. (All built-in RAM, LCD memory, and plug-in RAM data is lost. The real time clock is lost. The battery charge level is lost.)

When both plug-in ports are again occupied, the power supply for the PPU energizes and the PPU is reset.

The PPU waits until the **(O)** key is pressed before it wakes up the system (by applying power to the mainframe and both plug-in ports). The CPU is initially reset. When allowed to run, it reboots the BIOS (which reinitializes the RAM disk and the real-time clock).

The battery charge level initially reads 0 percent.

---

## 2.4 Mainframe Hardware

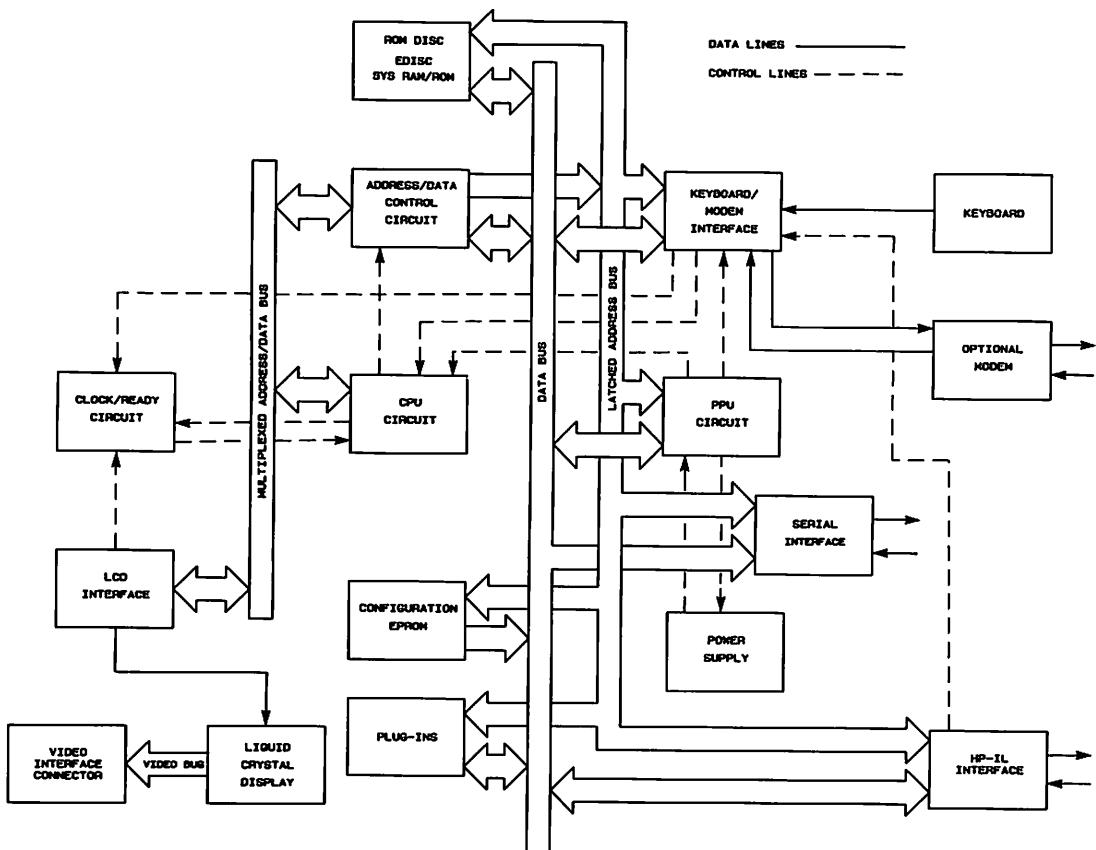
The mainframe (illustrated in figure 2-3) consists of the following assemblies:

- Motherboard (PCA), which contains the CPU and its associated circuitry, the peripheral-processor unit (PPU), 2 multi-purpose controllers, LCD controller, video interface, HP-IL interface, serial interface, the interface for the optional modem, and the power supply.
- Memory board (PCA), which contains built-in RAM, built-in ROM, the configuration EPROM, address decoding circuitry, and two plug-in ports.
- Keyboard assembly, which consists of 76 keyswitches (but no active circuitry).
- Liquid-crystal display module.
- Piezo-electric speaker.

In addition, the mainframe has provisions for two types of optional hardware:

- 1200-baud modem, which contains the modem circuitry and its power supplies. It is installed internally in the mainframe.
- Plug-in module, which usually contains additional RAM or ROM. It is installed in a plug-in drawer, which is then inserted into one of two external plug-in ports.

**Figure 2-3. Portable PLUS Block Diagram**



The following sections describe the basic operation of the system components.

### 2.4.1 CPU

The 80C86 CPU communicates on a multiplexed address-and-data bus (20-bit addresses, 16-bit data). The 80C86 is strapped into minimum mode, and thus produces its own bus-control signals.

### 2.4.2 Clocking

All mainframe clocking is contained on the motherboard. The 16-MHz crystal and the oscillator circuit generate several clock frequencies: 5.33 MHz for the CPU, 2.67 MHz for the multi-controllers, and 16 MHz for the binary counter, which generates a 2-MHz clock for the HP-IL controller.

The LCD controller has its own 5-MHz oscillator. The PPU has its own 1-MHz crystal and built-in oscillator, which always operates--even while the system is in sleep mode. (The optional modem has its own oscillator circuit.)

### 2.4.3 Ready Circuit

The bus-cycle length can vary, depending on the address of the device being accessed. This is accomplished using the CPU's READY input.

The lower 512K bytes of system memory runs with no wait states (cycle time of 748 ns). The upper 512K bytes of system memory and all I/O addresses operate with one wait state minimum (cycle times of 935 ns minimum). These cycles are further extended when the (open-drain) READY line is pulled low.

### 2.4.4 PPU - Peripheral-Processor Unit

The peripheral processor unit (PPU) is a single-chip microcomputer of the 6805 family. It has 112 bytes of RAM and 2106 bytes of ROM. The PPU controls the power supplies, operating modes, and the beeper, and it provides the real-time clock. It runs even while the system is in sleep mode. The PPU can be accessed if needed via a system service (Refer to Int 50h in chapter 5.)

A one-byte data transfer between the CPU and the PPU takes about 2.3 ms to complete.

## 2.4.5 Keyboard Interface

The keyboard assembly contains 76 key mechanisms, but no electronic components. Hardware provides the key location; software maps the location into unique keycodes. The keyboard is organized into an eight-by-nine matrix, plus three additional function modifier keys. A matrix key connects a row line to a column line. The multi-controller alternately cycles between activating all column drivers and sampling the row lines, and activating all row drivers and sampling column lines.

The three function modifier keys each have an individual pullup resistor to the positive supply. A closed key pulls the line to ground.

## 2.4.6 Power Supply

The power supply is overseen by the PPU and provides power for the entire mainframe, the optional modem, and plug-in boards. Power supply conditions for each of the system operating modes are:

- Awake mode. VccS and VccDS are +5V ( $\pm 0.25V$ ).
- Sleep mode. VccDS is +3.25V ( $\pm 0.16V$ ), VccS floats.

In addition to the two 5 volt supplies there is a negative voltage supply used to bias the LCD display.

## 2.4.7 Memory Board

The memory board contains RAM, ROM, address decoding circuitry, the configuration EPROM, and two plug-in ports. (The plug-in ports are described separately in this chapter.)

## 2.4.8 Configuration EEPROM

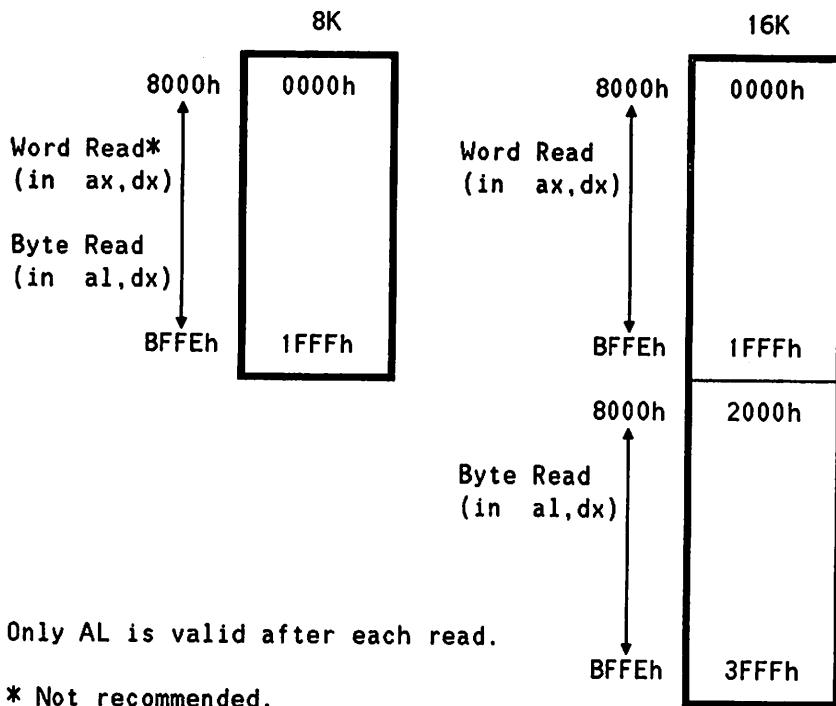
The configuration EEPROM resides in I/O memory at even addresses from 8000h to BFFFh. It is normally an 8Kx8 device (27C64), but it can be a 16Kx8 device (27C128). (A 32Kx8 EEPROM can be used, but only the upper 16K bytes are addressable.) The configuration EEPROM contains information which the BIOS uses to properly configure the system. The types of information it contains are:

- Product Number
- Serial Number
- Boot Information
- Country Specification
- Constants used by the BIOS
- Numeric Pad Map
- Font Loading Information
- Keyboard Matrix Maps
- System/Error Messages
- System Setup Information
- Option for Boot Code

Each supported language has a different version of the configuration EPROM. Appendix C contains a listing of the English (U.S.) version. It is possible to customize the main PAM screen and system/error messages by customizing the EPROM. Discussion of the option for boot code is in Chapter 11. Because only 8K of I/O memory is allocated to the configuration EPROM, special restrictions are required to use a 16K EPROM. For a given I/O address in the EPROM space, a "byte" access reads from a different EPROM location than a "word" access reads. For a 16K EPROM, a "word" access reads from its upper 8K (2000h through 3FFFh internal), but only the lower byte is valid; a "byte" access reads from its lower 8K (0000h through 1FFFh internal). For an 8K EPROM, both types of accesses read the same data (0000h through 1FFFh internal)--but for consistency only "byte" accesses should be made to an 8K EPROM. Figure 2-4 illustrates this.

Figure 2-4. Configuration EPROM Addressing

(Even Addresses Only)



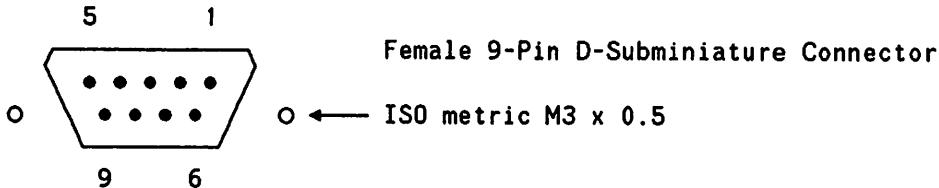
## 2.5 Serial Interface

The computer operates as a Data Terminal Equipment (DTE) on its serial interface. The interface complies with the following industry standards:

- EIA RS-232-C. Electrical specification (except that a 9-pin female connector is used instead of a 25-pin male connector).
- CCITT V.28. Electrical specification.
- CCITT V.24. Electrical specification (for the nine implemented lines).

Figure 2-5 shows the pin configuration for the nine-pin female serial connector. Table 2-1 lists the signals at the serial connector and relates the configuration to the EIA and CCITT standards.

**Figure 2-5. Serial Connector**



**Table 2-1. Serial Interface**

Pin Number	Signal	Equivalent RS-232-C Pin	RS-232-C Circuit Designator	V.24 Circuit Designator
1	Data Terminal Ready	20	CD	108/2
2	Transmitted Data Out	2	BA	103
3	Received Data In	3	BB	104
4	Request To Send	4	CA	105
5	Clear To Send	5	CB	106
6	Data Set Ready	6	CC	107
7	Ground Reference	7	AB	102
8	Received Line Signal Detect	8	CF	109
9	Ring Detect	22	CE	125

The serial interface function is shared by the multi-controller IC, the PPU, and the HP-IL IC. The multi-controller IC controls the frame format and receiver/transmitter status. The PPU controls power for the line drivers and controls the RTS and DTR output lines. The HP-IL IC maintains the status of the CTS and DSR input lines. The multi-controller is able to connect either the serial Rx/D line or its own serial output line to the receiver's serial input. Thus, the multi-controller is able to isolate the receiver from the serial Rx/D line. This is recommended during power-up or power-down sequences and when serial power is off.

The multi-controller is powered in sleep mode, but is reset as the system comes out of stop mode.

**Output Electrical Characteristics.** The outputs are the TxD, DTR, and RTS signals.

2

The low level output voltage, Vol, for the TxD signal is considered the logic 1 state. For the DTR and RTS signals, it is considered the OFF state. Iol is the magnitude of the current provided by an output when driving the signal to Vol. All voltages are specified with respect to GND. The RS-232-C and CCITT Recommendation V.28 limits are:

Vol (Iol = 0 mA)	:	-25 V min.
Vol (3000 ohms < LOAD < 7000 ohms)	:	-15 V min. -5 V max.
Iol (output shorted to +15 V)	:	500 mA max.

The actual limits guaranteed by the Portable PLUS serial interface design are:

Vol (Iol = 0 mA)	:	-15 V min.
Vol (Iol = 2 mA)	:	-15 V min. -6.6 V max.
Iol (output shorted to +15 V)	:	2.5 mA min. 45 mA max.

The high level output voltage, Voh, for the TxD signal is considered the logic 0 state. For the DTR and RTS signals, it is considered the ON state. Ioh is the magnitude of the current provided by an output when driving the signal to Voh. The RS-232-C and CCITT Recommendation V.28 limits are:

Voh (Ioh = 0 mA)	:	+25 V max.
Voh (3000 ohms < LOAD < 7000 ohms)	:	+5 V min. +15 V max.
Ioh (output shorted to -15 V)	:	500 mA max.

The actual limits guaranteed by the serial interface design are:

Voh (Ioh = 0 mA)	:	+8 V max.
Voh (Ioh = 2 mA)	:	+5.2 V min. +8 V max.
Ioh (output shorted to -15 V)	:	10 mA min. 45 mA max.

**Miscellaneous Output Characteristics.** RS-232-C and CCITT Recommendation V.28 require the following characteristics of output signal drivers:

Transition time (between -3 and +3 V): 200 nsec min. 1.56 usec max.

Power-off impedance (+-2 V applied) : 300 ohms min.

The actual limits guaranteed by the serial interface design are:

Transition time (between -3 and +3 V): 200 nsec min. 1.50 usec max.

Power-off impedance (+-30 V applied) : 300 Kohms min.

**Input Electrical Characteristics.** The inputs are the RxD, DSR, CTS, RLSD, and RING signals. All voltages are specified with respect to GND.

The low level input voltage, Vil, for the RxD signal is considered the logic 1 state. For the DSR, CTS, RLSD, and RING signals, it is considered the OFF state. Vih is considered the logic 0 state for the RxD signal and the ON state for the DSR, CTS, RLSD, and RING signals. The RS-232-C and CCITT Recommendation V.28 require that a device properly interpret input signals that fall within the following voltage limits:

Vil (logic 1 state or OFF state) : -25 V min. -3 V max.

Vih (logic 0 state or ON state) : +3 V min. +25 V max.

However, the serial interface will properly interpret input signals which are within these larger ranges:

Vil (logic 1 state or OFF state) : -25 V min. +0.6 V max.

Vih (logic 0 state or ON state) : +3.0 V min. +25 V max.

When in SLEEP mode, the serial interface can respond to two of the input signal lines, RING and RLSD. These signals are properly interpreted when their voltages are within the following ranges:

Vil (OFF state) : -25 V min. +0.3 V max.

Vih (ON state) : +2.4 V min. +25 V max.

**LOAD and EI.** LOAD is DC resistance of an input signal line measured from that line to GND. EI is the magnitude of the open-circuit voltage that an input signal line generates. RS-232-C and CCITT Recommendation V.24 specify these quantities to be within the following limits:

LOAD (-25 V to +25 V applied) : 3000 ohms min. 7000 ohms max.

EI : 2 V max.

The actual limits guaranteed by the serial interface design are:

LOAD (-25 V to +25 V applied) : 4400 ohms min. 5000 ohms max.

EI : .05 V max.

**Cables.** Two cables are available for connecting the computer to serial devices: a modem cable (DTE to DCE) and a printer cable (DTE to DTE). A gender converter (HP 92222F) is available to convert each cable from male to female. Figure 2-6 describes the modem cable (HP 92221M).

**Figure 2-6. Modem Cable**

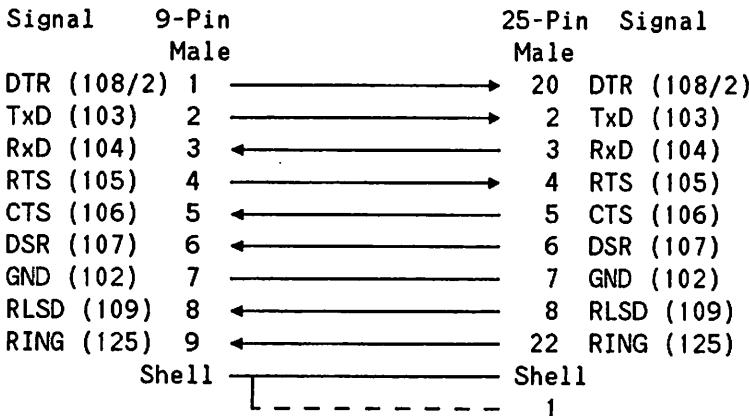
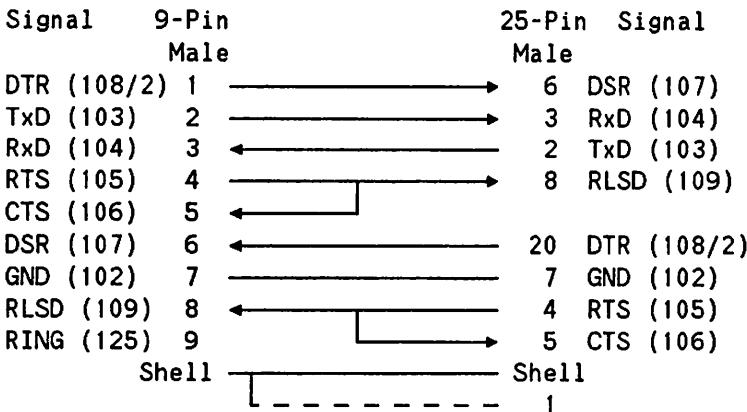


Figure 2-7 describes the printer cable (HP 92221P).

**Figure 2-7. Printer Cable**



## 2.6 HP-IL Interface

The HP-IL interface conforms to the Hewlett-Packard Interface Loop standard, as described in *The HP-IL Interface Specification* (HP part number 82166-90017). Standard "IN" and "OUT" HP-IL receptacles are provided on the I/O plate.

## 2.7 Recharger Interface

Power may be applied to operate the computer and charge the internal battery pack through a two pin jack (labelled "RCH") which is located on the rear panel. The RCH connector is interfaced with the mainframe's internal battery charger circuitry. Both computer operation and battery charging occur simultaneously when power is applied as long as the power that is applied at the RCH input is more than the power that the computer is using. If this is not the case, battery drain continues, but at a reduced rate.

**Battery Charger Operation.** The battery charger circuit was designed to work with a specific group of AC adapters made by Hewlett-Packard. (The U.S. model is the HP-82059D.) These adapters provide current limiting; therefore the battery charger circuit within the computer is designed without current limiting. Power that is applied to the RCH input must therefore be adequately limited to ensure the survival of the computer's circuitry. Excessive voltage can cause the DC rectifier to be damaged. Excessive current can cause the battery fuse to blow. Excessive power applied when the battery is nearly fully charged can damage the battery charger's voltage regulator.

Current from the recharger has two possible paths, the computer circuits, and the battery. When the battery charger regulator is providing more current than the computer circuits are using, excess current flows into the battery, charging it. Otherwise, current flows out of the battery to satisfy the computer's needs.

The voltage applied to a fully charged battery by the internal voltage regulator is selected to give optimum battery life. This optimum voltage (called "float" voltage) varies with temperature. The battery charger regulator is designed to maintain the proper float voltage over a temperature range of -10 to +55 C.

**Battery Percentage Indicator.** The computer maintains a Battery Fuel Gauge (main PAM screen) which operates during battery charging. This indicator assumes a certain minimum charge current from the regulator when an AC adapter is connected. If less than this current is provided, the battery percentage indication may show a higher percentage of charge than the battery actually has. It is intended that the battery percentage indication always under-estimate the remaining battery charge, rather than over-estimate it.

**DC Requirements of the RCH Port.** DC power may be applied to the RCH input in the form of a DC voltage source with a series output resistance. The limitations on the voltage source and series resistance are, in general, functions of temperature. This is due to the varying response of the battery charger voltage regulator to temperature, as required by the battery. The limits are listed in Table 2-2. The maximum input current limitations are given for reference only.

**Table 2-2. Recharger DC Limits**

Temp (°C)	Maximum Input Voltage (Volts)	Maximum Input Current (Amps)	Minimum Input Voltage @ 0 Current (Volts)
-10	21	0.5	9.06
0	21	0.5	8.93
10	21	0.5	8.83
20	21	0.5	8.72
25	21	0.5	8.66
30	21	0.5	8.59
40	21	0.5	8.46
50	21	0.5	8.32
60	21	0.5	8.19

The limitations on series resistance given below guarantee that the maximum input currents are not exceeded. The series resistance limitations are functions of the value of the DC voltage source used to supply power to the RCH input. They are also functions of battery charger circuit operating parameters which are not detailed here. Table 2-3 specifies the minimum and maximum limitations of the series resistance for a given DC voltage source (open circuit voltage). Limitations are given for two operating temperature ranges--a restricted office temperature range and the full operating temperature range of the Portable PLUS.

Table 2-3. Recharger Series Resistance Limits

Voltage (Volts)	Office Temperature Range (15 to 35°C)		Full Temperature Range (-10 to 60°C)	
	Minimum Resistance (Ohms)	Maximum Resistance (Ohms)	Minimum Resistance (Ohms)	Maximum Resistance (Ohms)
9 *	5.2	6.0	---	---
10 *	7.2	8.0	7.6	8.0
11 *	9.2	10.0	9.6	10.0
12	11.2	12.7	11.6	12.0
13	13.2	16.3	13.6	15.2
14	15.2	19.9	15.6	18.8
15	17.2	23.4	17.6	22.3
16	19.2	27.0	19.6	25.9
17	21.2	30.6	23.7	29.5
18	23.2	34.1	28.7	33.0
19	25.2	37.7	34.2	36.6
20	27.5	41.3	40.2	41.0

\* Battery charging with these supply voltages is possible but not recommended. It may result in PAM's Battery Fuel Gauge over-estimating the remaining capacity of the battery.

**AC requirements of the RCH Port.** AC power may be applied to the RCH input in the form of an AC voltage source with a series output resistance. The source and resistance should match the characteristics of the Hewlett-Packard AC adapters which are designed to be used with the computer.

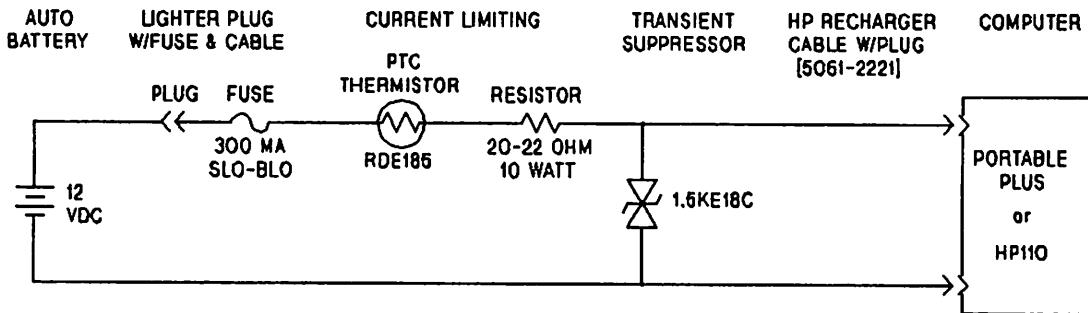
Output Voltage (open circuit) :  $11.6 \pm 0.2$  volts rms

Output Resistance :  $11.0 \pm 0.5$  ohms

Frequency : 47.5 Hz minimum, 440 Hz maximum

**Automotive Recharger.** Conceptually, the circuit shown in figure 2-8 fulfills the requirements for a 12 Vdc automobile recharger. This circuit has not been thoroughly tested and Hewlett-Packard assumes no responsibility for its use. Other circuit topologies are clearly possible.

**Figure 2-8. Automotive Recharger Schematic Diagram**



- The automobile's voltage regulator must maintain a dc level between 11 and 16 volts to provide effective and safe power.
- An automobile's 12 Vdc source can experience a short-duration transient of hundreds of volts, or a high energy transient of up to 80 volts which may not decay for several seconds. Thus, transient voltage suppression must be provided to protect the computer. In addition, the HP recharger cable includes miniature back-to-back 27-volt Zener diodes in its plug--they may open or short when overstressed. (These Zeners prevent a high voltage static discharge when the plug is inserted or removed.)

The transient suppressor should protect both the computer and the recharger plug's Zener diodes. Placing the suppressor on the resistor's output side allows it to survive the voltage transients it is designed to suppress. (The 1.5KE18C is manufactured by General Semiconductor and Motorola.)

- The lighter plug must be fused to protect against a gross short. The fuse size (300 mA) and type (slo-blo) should be chosen to survive high energy transients while still blowing for a sustained high current condition. Note that a current ranging from 300 mA (indefinitely long duration) to 1200 mA (very short duration) may be required to blow a 300 mA fuse.

- The thermistor/resistor combination must be *bonded* in intimate thermal contact; a silicone heatsink compound is a good choice. The resistor limits the current during normal operation and is large enough to sustain a direct short at the output--for which the thermistor should then trip.

The thermistor's resistance is negligible when cool. Internal self heating and external heating from the resistor combine in a high current condition (about 500 mA) to "trip" the thermistor into a high resistance state. The thermistor will maintain this low current state until power is removed. (The RDE185 is manufactured by Raychem Corporation, Menlo Park, CA.)

- Although the HP recharger cable will plug into a number of HP computers, calculators, and peripherals, a dc circuit designed for the Portable PLUS and HP110 may *not* provide effective or safe power for other devices. (The recharger cable is manufactured by Hewlett-Packard. Refer to appendix I for more information.)
- A package containing the thermistor, resistor and transient suppressor must be designed to avoid overheating or melting while still sustaining a current not quite large enough to blow the fuse. The package must also protect the user from possible contact with any wire or component, and should provide adequate cable strain relief.

---

## 2.8 Video Connector

The video interface connector is located inside the battery compartment. Six of the signals that drive the internal LCD are present at the video interface connector. These signals can be used by an external driver to generate a video display. Table 2-4 describes the video signals.

**Table 2-4. Video Signals**

<b>Pin</b>	<b>Name</b>	<b>Description</b>	<b>Frequency</b>
7	CL2	Dot Clock	1.25 MHz
5	FLM	Frame Clock	52 Hz
2	DI1	Upper-Left Quadrant Data	625 KHz max.
1	DI2	Upper-Right Quadrant Data	625 KHz max.
4	DI3	Lower-Left Quadrant Data	625 KHz max.
3	DI4	Lower-Right Quadrant Data	625 KHz max.
6,8	GND	Ground Reference	--

Note: Pin 1 is the left-most pin as you face the rear of the product.

Table 2-5 lists the voltage and timing specifications for the video signals.

**Table 2-5. Video Specifications**

<b>Parameter</b>	<b>Specification</b>
All: high output voltage	3.80 volts min.
All: low output voltage	0.95 volts max.
CL2: high pulselwidth	225 ns min.
CL2: risetime	160 ns max.
CL2: falltime	115 ns max.
FLM: high setup to CL2 fall	280 ns min.
FLM: high hold from CL2 rise	370 ns min.
DI1-DI4: Data setup to CL2 fall	210 ns min.
DI1-DI4: Data hold from CL2 fall	270 ns min.

## 2.9 Modem Connector

The modem connector, located on the motherboard, provides an internal interface designed for the optional modem. The signals provided at the modem connector are described in table 2-6. All data and control signals are CMOS-compatible. The AUX Device Driver handles the low level modem control.



### Note

An asterisk in a signal name (\*) indicates a negative-true signal (active low).

**Table 2-6. Modem Connector**

Pin	Signal	Description	Direction
1	MRESET*	Modem Reset. A low voltage should reset the modem. This line will be driven low before MODEMON goes low, and will remain low 50 ms after MODEMON goes high.	→ Modem
2	MRING*	Modem Ring. Falling edge indicates a ring signal on the phone line. This line should function when the modem is either on or off. Mainframe has 47K pullup resistor to VccDS on this line.	← Modem
3,5,7,11	GND	Ground Reference.	
4	MSOUT	Modem Serial Out. Transmitted data line from mainframe to modem. Mark is 5V.	→ Modem

**Table 2-6. Modem Connector (Continued)**

6	<b>MSIN</b>	Modem Serial In. Received data line from modem to mainframe. Mainframe has 47K pullup to VccDS on this line. Mark is OV.	← Modem
8	<b>MCARRIER</b>	Modem Carrier. Falling edge indicates a loss of carrier on the phone line. Required to function only when the modem is on. Mainframe has 47K pullup to VccDS on this line.	← Modem
9	<b>VBAT</b>	Battery. Unregulated battery positive supply line (fused on motherboard). 5.6 to 7.5 Vdc.	→ Modem
10	<b>MODEMON</b>	Modem On. A high voltage on this line should turn on the modem power supply. When this line is low, the modem should reduce its power consumption to standby (microamp) levels.	→ Modem
12	<b>MRCM</b>	Modem Return to Command Mode. A high voltage on this line for 100 ms (or longer) returns the modem to command mode.	→ Modem

Table 2-7 lists the specifications for a circuit connected to the modem connector.

**Table 2-7. Specifications for Modem Port**

Signal	Parameter	Min	Max	DC Load
<b>Input:</b>				
MSOUT	Vil	0V	0.4V	Iol ≤ 1.6 mA
	Vih	4.25V	5.25V	Ioh ≤ -150 uA
MRESET*	Vii	0V	0.1V	Iol ≤ 10 uA
	Vil	0V	0.4V	Iol ≤ 800 uA
	Vih	2.4V	5.25V	Ioh ≤ -2 mA
	Vih	4.65V	5.25V	Ioh ≤ -10 uA
MODEMON	Vil	0V	0.1V	Iol ≤ 10 uA
	Vil	0V	0.4V	Iol ≤ 800 uA
	Vih	2.4V	5.25V	Ioh ≤ -8 mA
	Vih	4.65V	5.25V	Ioh ≤ -10 uA
MRCM	Vil	0V	0.1V	Iol ≤ 10 uA
	Vil	0V	0.4V	Iol ≤ 800 uA
	Vih	2.4V	5.25V	Ioh ≤ -2 mA
	Vih	4.65V	5.25V	Ioh ≤ -10 uA
<b>Open-Drain Outputs:</b>				
MSIN	Vol	0V	0.9V	Iol ≤ 113 uA
MCARRIER	Vol	0V	0.8V	Iol ≤ 113 uA
MRING*	Vol	0V	0.8V	Iol ≤ 113 uA

The modem interface takes on the following state in sleep mode:

MRCM: 0V nominal.

MRESET\*: 0V nominal.

MODEMON: 0V nominal.

MSIN: 47K pullup to 3.25V nominal.

MSOUT: 3.25V nominal.

MRING\*: 47K pullup to 3.25V nominal.

MCARRIER: 47K pullup to 3.25V nominal.

## 2.10 Plug-In Ports

The Portable PLUS provides two plug-in ports that are each capable of accepting a plug-in module, which becomes part of the system.

### 2.10.1 Generic Module Description

A plug-in module for the Portable Plus is a printed-circuit assembly mounted in a "drawer" that fits into the mainframe. This provides the capability to expand or customize the hardware configuration of the computer. The modules become an integral part of the computer bottom case.

These modules would typically be additional RAM or ROM for the computer system but might also be a serial or parallel interface or any other custom circuit which can be operated from the system bus.

The mainframe allows up to two plug-in modules to be plugged in at the same time.

### 2.10.2 Electrical Specifications

There are two categories of requirements which must be met in order for a plug-in module to operate correctly. First, the module must provide the appropriate identification and control registers to allow the system software to integrate the module into the operating environment. The second set of requirements are associated with the interfacing of the mainframe electrical circuitry with the circuitry of the module. These specifications include such things as power consumption, drive capability, loading limitations, digital signal timing, and environmental tolerances.

**Plug-In Connector.** The interface signals for the plug-in ports are listed in table 2-8.

2



An asterisk (\*) in a signal name indicates a negative-true signal (active low).

**Note**

**Table 2-8. Signals for Plug-In Ports**

Name	Description	Direction
LA19 - LA0	Latched address bus (20 bits). Low voltage on LA0 indicates data transfer on the low byte of the data bus, D7-D0. LA19-LA16 are low during I/O cycles.	→ Plug-In
LBHE*	Latched byte high enable. Low voltage indicates data transfer on the high byte of the data bus, D15-D8. This line switches with LA19-LA0.	→ Plug-In
LM/IO*	Latched memory or I/O signal. High implies memory access; low implies I/O access. This line switches with LA19-LA0.	→ Plug-In
BALE	Buffered address latch enable. Pulses high to signify start of CPU bus cycle. Occurs without RD* or WR* pulsing low during interrupt acknowledge cycles.	→ Plug-In
IOCS*	I/O space address decode. Low active. Sixteen words wide. Pulses inactive at beginning of cycle. Used to access the ID and configuration registers of the plug-in card.	→ Plug-In
D15 - D0	Demultiplexed data bus (16 bits).	↔ Plug-In

**Table 2-8. Signals for Plug-In Ports (Continued)**

<b>BRD*</b>	Buffered CPU read strobe. Low active. Due to a race in the 80C86 CPU, glitches may appear on this line.	→ Plug-In
<b>BWR*</b>	Buffered CPU write strobe. Low active.	→ Plug-In
<b>DEN*</b>	Data bus driver enable. Low active. Timing strobe to turn on data buffers when plug-in card is addressed.	→ Plug-In
<b>DT/R*</b>	Data bus buffer direction control. Low voltage: plug-in buffers drive data out to the mainframe. High voltage: plug-in buffers drive mainframe data to plug-in card.	→ Plug-In
<b>READY</b>	Handshake line that extends the length of a CPU bus cycle. A low voltage on this line extends the cycle (by inserting processor wait states) until the READY line returns to a high voltage. The processor bus cycle then ends normally. READY should be driven with open-drain devices only. The mainframe has a 4.7K-ohm pullup resistor to VccS on the line.	← Plug-In
<b>SLEEP*</b>	A low voltage on this line indicates that the system is going to sleep. Intended for use as a reset line for devices on the VccS supply. In response, a plug-in card should reduce power to standby levels, prepare for VccS to float, and the plug-in interface to assume its sleep state. Low for 200 ms after VccS re-energizes.	→ Plug-In

Table 2-8. Signals for Plug-In Ports (Continued)

DSLEEP*	Deep sleep. Low voltage indicates that the mainframe is preparing to have its plug-in cards reinserted. Intended for use as a reset line for devices on the VccDS supply. Driven to VccDS in awake mode, and when both plug-in cards are present in sleep mode. Driven low when a plug-in card is removed in sleep mode (so devices on VccDS must be able to reset with 0 volts on DSLEEP* when VccDS is at 3.25 volts). Stays low until both plug-ins are inserted and the  key is pressed to wake up the system.	→ Plug-In
PRESENT*	Low-voltage signal to mainframe that card is plugged in. Thus all cards should ground this line. Care should be taken to minimize leakage paths on this signal.	← Plug-In
INT*	Negative-edge-triggered interrupt line. Mainframe has an internal 47K ohm pullup resistor to DSLEEP* on this line. If the plug-in's software driver has enabled this interrupt, it can be used in a conventional sense in awake mode, and can wake up the system from sleep mode. (In sleep mode, the plug-in should not pull this line low, unless it desires to wake up the system.) INT* pullup is sourcing current in sleep mode, except when a plug-in card is removed. When this occurs, DSLEEP* is switched to ground. This avoids a possible latchup condition of having a high voltage on INT* before the plug-in VccDS bus energizes as the plug-in is inserted.	← Plug-In
VccDS	+5 volts nominal (awake mode), +3.25 volts nominal (sleep mode). Energized except when the mainframe is in stop mode.	→ Plug-In

**Table 2-8. Signals for Plug-In Ports (Continued)**

VccS	+5 volts nominal (awake mode). Energized except when the mainframe is in sleep mode or stop mode.	→ Plug-In
GND	Mainframe logic ground reference.	

Figure 2-9 shows the connector in the plug-in port. The even-numbered pins are on the top row (toward the top case); the odd numbered pins are on the bottom. Pin 1 is the lower-left pin when looking into the port; pin 62 is the upper-right pin.

**Figure 2-9. Plug-In Connector**

GND	62	61	VccDS
D6	60	59	D7
D4	58	57	D5
VccS	56	55	D3
D2	54	53	D1
D0	52	51	VccDS
BWR*	50	49	LM/I0*
LA0	48	47	LBHE*
GND	46	45	BALE
DT/R*	44	43	BRD*
DEN*	42	41	READY
GND	40	39	DSLEEP*
LA18	38	37	LA19
LA16	36	35	LA17
LA14	34	33	LA15
VccS	32	31	IOCS*
GND	30	29	INT*
LA12	28	27	LA13
LA10	26	25	LA11
LA8	24	23	LA9
VccS	22	21	LA7
LA6	20	19	LA5
LA4	18	17	LA3
LA2	16	15	LA1
GND	14	13	SLEEP*
D14	12	11	D15
D12	10	9	D13
VccS	8	7	D11
D10	6	5	D9
D8	4	3	VccDS
GND	2	1	PRESENT*

**Bus Specifications.** The four timing diagrams on the following pages show timing for the plug-in bus when the plug-in board contains RAM, ROM, and I/O space.

The lower 1/2 megabyte in memory space runs with no CPU wait states. There is no way to extend CPU bus cycles in this address range. The upper 1/2 megabyte of memory space and all 64K bytes of I/O space run with a minimum of one wait state, and can be further extended using the READY line.

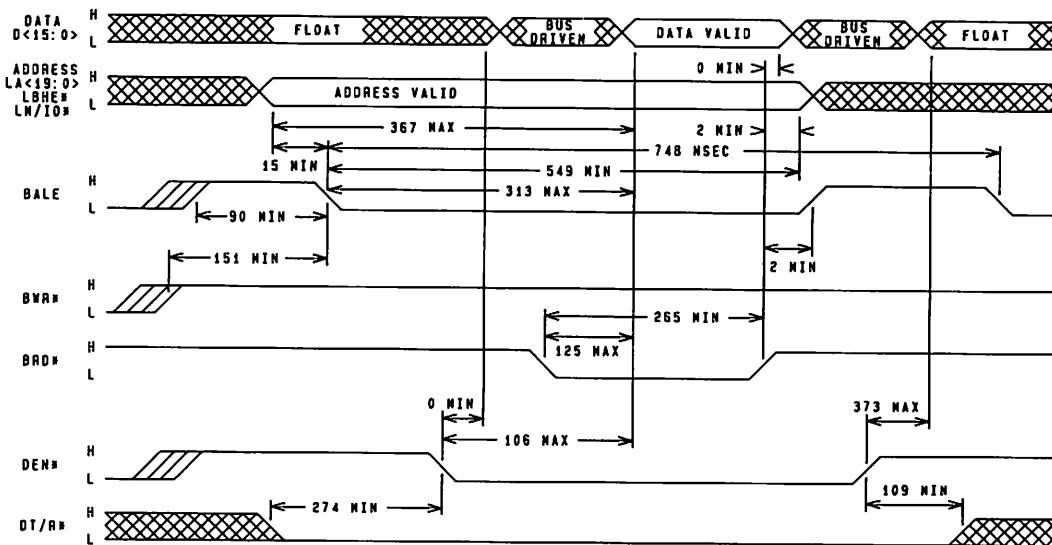
For extended cycles:

BALE low to READY low: 289 ns max.

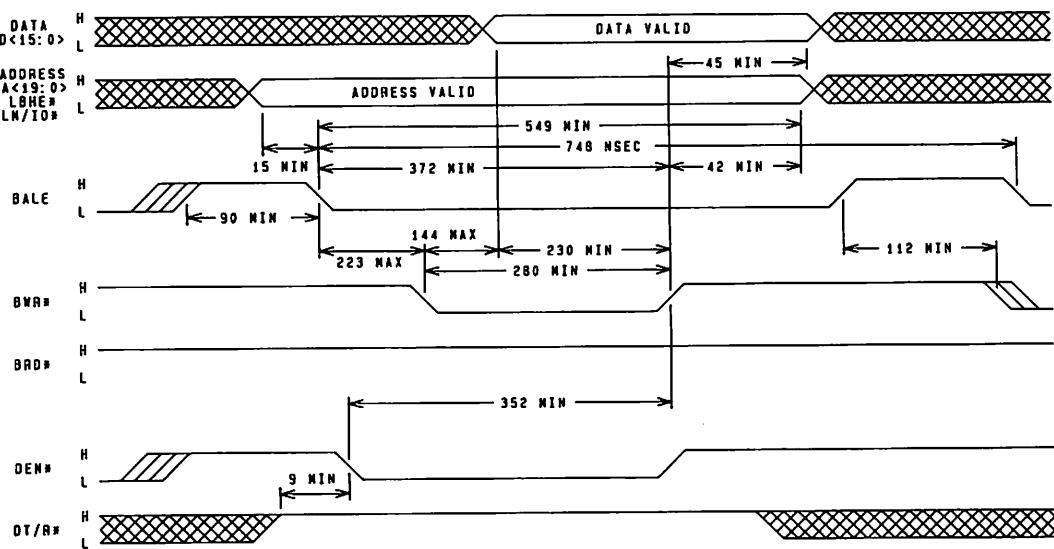
IOCS\* low to READY low: 236 ns max.

READY tristate to data is valid: 138 ns max.

**Figure 2-10. Plug-In Bus Read Cycle - Lower Memory**

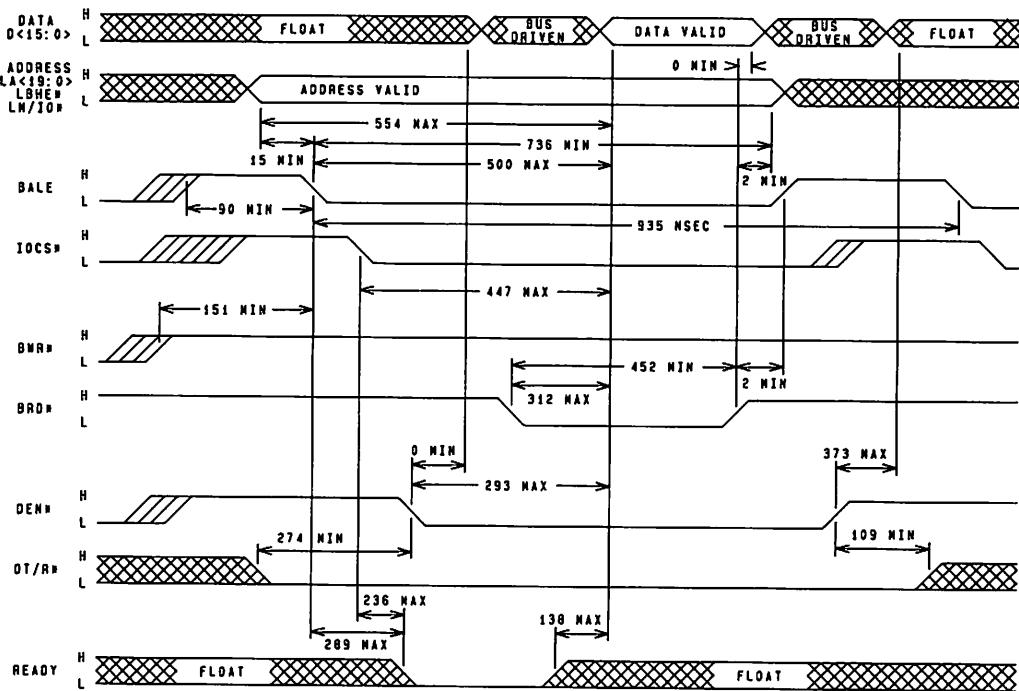


**Figure 2-11. Plug-In Bus Write Cycle - Lower Memory**



**Figure 2-12. Plug-In Bus Read Cycle - I/O and Upper Memory**

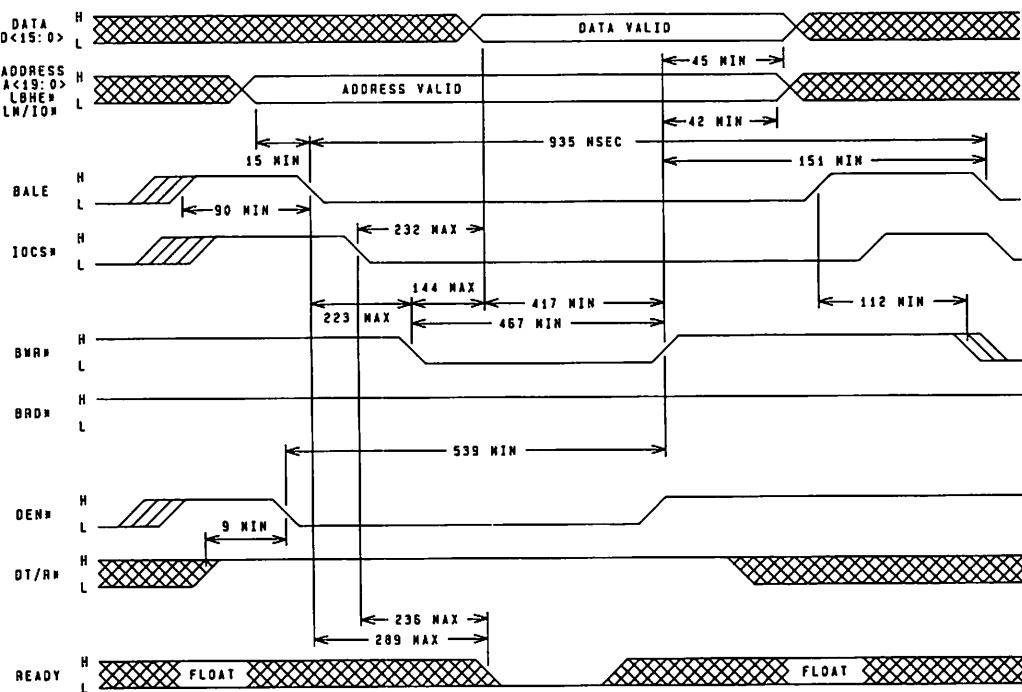
2



TIME IN NANOSECONDS.

NOTE: IOCS\* PULSES LOW ONLY FOR I/O CYCLES WITH AN ADDRESS CORRESPONDING TO THE 16-WORD PLUG-IN PORT ADDRESS RANGE.

**Figure 2-13. Plug-In Bus Write Cycle - I/O and Upper Memory**

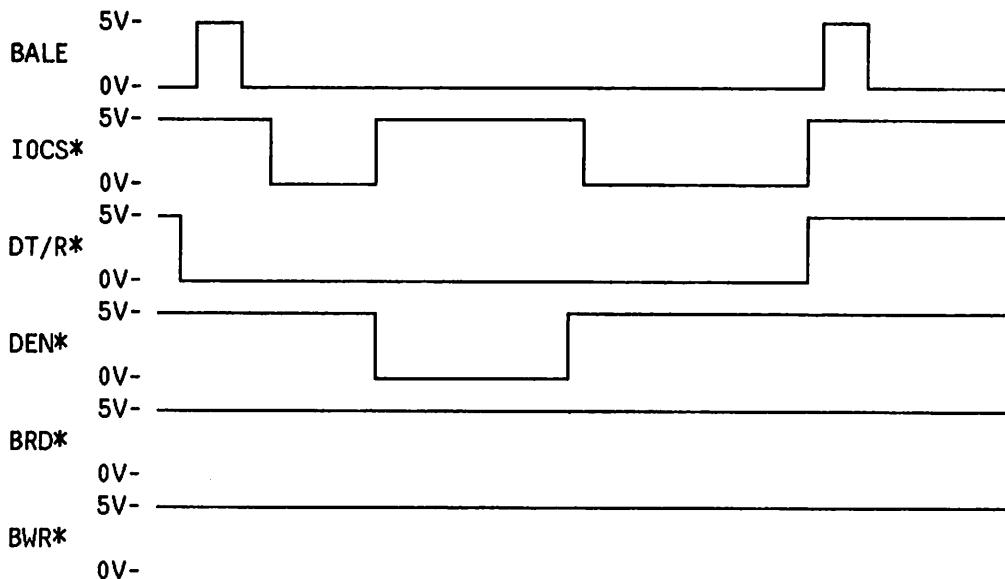


TIME IN NANoseconds.

NOTE: IOCS\* PULSES LOW ONLY FOR I/O CYCLES WITH AN ADDRESS CORRESPONDING TO THE 16-WORD PLUG-IN PORT ADDRESS RANGE.

During an interrupt acknowledge cycle, the plug-in bus receives normal BALE and DEN\* pulses, but BRD\* and BWR\* remain high. The READY line is still sampled by the mainframe. The CPU cycle is in I/O address space, but the address bus is indeterminate. If the address is that of the IOCS\* space, the waveforms shown in figure 2-14 occur.

**Figure 2-14. Plug-In Bus Interrupt Acknowledge Cycle**



The plug-in card may pull READY low (inactive) in response to the IOCS\* low, but it must release READY when IOCS\* returns high so that the bus can complete normally--bus operation will wait for READY to return high.

The addressed plug-in is allowed to drive the data bus to the mainframe in response to DEN\* and DT/R\* low.

**Plug-in Bus Loads.** Circuits connected to the plug-in ports should not exceed the capacitive loads listed in table 2-9. (Be sure to add the plug-in connector load when calculating the total capacitive load.)

2

**Table 2-9. Plug-In Bus Loading**

Signal	Capacitive Load	DC Load
LA19-LA14	70 pF max.	15 uA max.
LA13-LA1	60 pF max.	15 uA max.
LA0	70 pF max.	15 uA max.
LBHE*	70 pF max.	15 uA max.
LM/IO*	70 pF max.	15 uA max.
BALE	50 pF max.	15 uA max.
IOCS*	35 pF max.	5 uA max.
D15-D0	30 pF max.	10 uA max.
DEN*	25 pF max.	40 uA max.
DT/R*	30 pF max.	40 uA max.
DSLEEP*	50 pF max.	50 uA max.
SLEEP*	50 pF max.	50 uA max.
READY	25 pF max.	20 uA max.
BRD*	280 pF max.	40 uA max.
BWR*	280 pF max.	40 uA max.

**Plug-in Power Loads.** The allowable power supply loads are listed in table 2-10.

**Table 2-10. Plug-In Power Loads**

Supply	DC Current	Capacitance
<b>Awake Mode:</b>		
VccDS (5V ± 5%)	(75 mA max.)	25 uF max.
VccS (5V ± 5%)	(VccDS, VccS combined)	25 uF max.
<b>Sleep Mode:</b>		
VccDS (3.25V ± 5%)	300 uA max.*	
* This maximum current severely limits battery life in sleep mode. Typical plug-in modules draw less than 50 uA in sleep mode.		



**Note** Plug-in card software drivers are expected to provide power consumption levels to the system using the appropriate INT 50h service, to ensure the accuracy of the PAM battery fuel gauge.

**Voltage Levels for Plug-in Bus.** Table 2-11 shows the required input and output voltage levels for the plug-in bus.

**Table 2-11. Voltage Levels for Plug-In Bus**

Symbol	Description	Voltage
Vil	Input low voltage	0.8V max.
Vih	Input high voltage	Vcc - 1.0V min.
Vol	Output low voltage	0.4V max.
Voh	Output high voltage	Vcc - 0.5V min.

**Requirements for Plug-in Drivers.** The plug-in boards are required to drive the loads listed in table 2-12. (Be sure to exclude the connector and the board itself when evaluating the plug-in device's drivers.)

**Table 2-12. Requirements for Plug-In Drivers**

Signal	Capacitive Load	DC Current Load
D15 - D8	248 pF max.	126 uA max.
D7 - D0	272 pF max.	128 uA max.
READY	85 pF max.	1.5 mA max.
INT*	45 pF max.	118 uA max.

### 2.10.3 Architectural Requirements

Though the circuitry of a plug-in module is mostly unique to its specific function, the architecture of the module must conform to the following interface specifications in order to function properly in the mainframe environment. Identification and control registers must be provided as described in the following two sections.

**Identification and Control Registers.** Each plug-in module slot has 16 word-wide I/O address locations assigned to it. These address locations are used for the control and status information which is required by the system and by the application for which the module is designed. This group of 16 address locations is unique for each plug-in module slot, so there are two 16-word address spaces designated in the memory map for the two slots.

Since the two module slots are virtually identical, a module does not "need to know" which of the two slots it is installed in. However, for the system to associate each of the installed modules with the I/O address space into which it is mapped, the hardware pre-decodes a "module select" line which is unique to each slot and is used by the module to enable accesses to its control and identification registers. This "module select" signal is called IOCS\*. Thus of the signals going to the two plug-in module slots, one signal that is not logically identical in both slots is the IOCS\* signal, which is never

active in both module slots at the same time. (The only other signal that isn't identical in both slots is the INT\* signal.)

When the IOCS\* signal of a module is active, the group of special I/O addresses associated with a module installed in that particular slot is being selected. The module can then execute the appropriate bus cycle using signals LA4-LA1, LA0, and LBHE\* to select the specific address within the group and using signals BRD\* and BWR\* to control the type (read or write) and timing of the data transfer. The timing of the IOCS\* signal is logically equivalent to the BALE signal.



**Note**

All modules *must* use the IOCS\* signal to select their identification and control registers, since this is the only mechanism by which the system can uniquely identify and access these registers when two modules are installed.

The 16 word-wide address locations selected by the IOCS\* signal are referred to here as "local I/O address locations"--that is, the I/O address range selected by IOCS\* and the specific location defined by address lines LA4-LA0 and LBHE\*.

**Virtual Modules.** An individual plug-in module is logically partitioned into two "virtual modules", each with its unique identification and control registers. This allows for the possibility of up to four uniquely controlled and operated virtual modules to be configured into a system at once (two virtual modules per physical module).

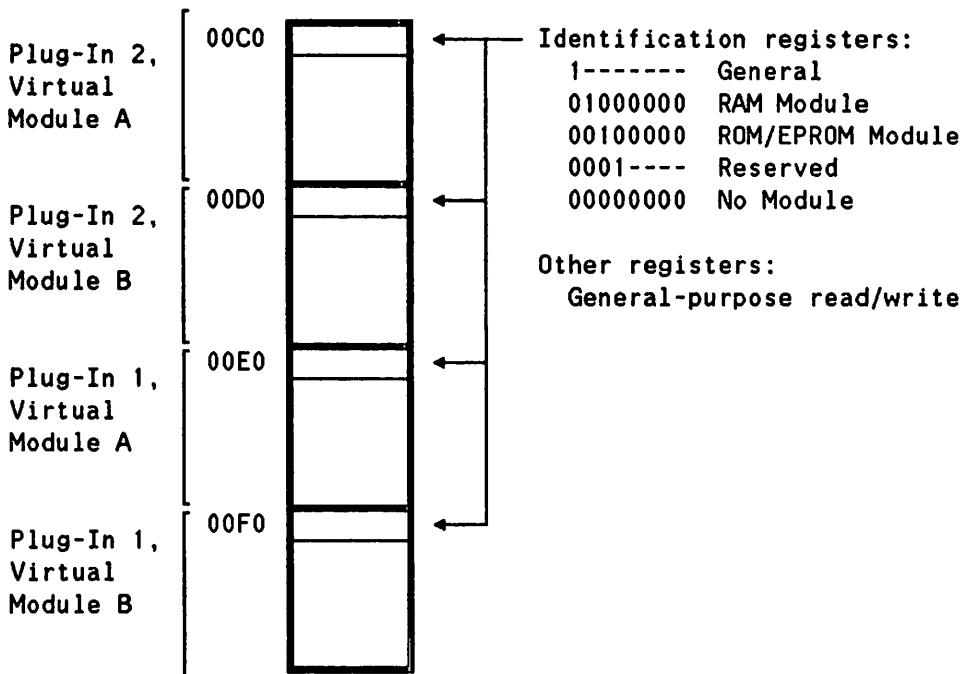
The local I/O locations of a physical module are logically partitioned into two sub-groups of eight address locations. One of these eight-word sub-groups is assigned to each of the two virtual modules. Address signal LA4 is used to select one or the other of the two sub-groups and signals LA3-LA0 and LBHE\* are used to address individual bytes within the sub-group. Signal LA4 is therefore used to distinguish between the control/status registers of the two virtual modules.

Of the eight I/O word addresses assigned to each virtual module, one byte-wide address location is reserved for the identification register of that virtual module while the remaining address locations are available to be used by the virtual module as needed by the application for which the module was designed.

The identification registers of the two virtual modules are assigned to local I/O addresses "00000" and "10000" binary (LA4-LA0, low byte). These "local" addresses correspond to the system I/O addresses C0h and D0h or E0h and F0h, depending on the

particular slot in which the module is installed. Figure 2-15 describes the identification and general-purpose registers. You can read the module identification using system services interrupt 50h (refer to chapter 5).

**Figure 2-15. Plug-In Module Registers**



Plug-in port 1 is behind the **[Return]** key.

Plug-in port 2 is behind the **[F1]** key.

**Identification of Virtual Modules.** The identification register of each virtual module is read by the system to determine the type of virtual module which is installed. The identification number read from that location must conform to the standard presented in figure 2-15 above. You can read the identification of a module using the system services interrupt 50h.

There must be an identification register for both virtual modules within a physical module. If only one virtual module is utilized, then the identification byte for the other

virtual module must be "00000000". An identification register is a single byte and is generally "read-only" since it always returns the identification code.

**Initialization of Virtual Modules.** When a system is booted, the byte 0h is written to the 16 high- and low-byte local I/O address locations of each virtual module with a "ROM" identification. For other types of virtual modules, only the 8 low-byte locations are zeroed (the high-bytes aren't altered). All modules must respond by being disabled and must not respond to any memory accesses other than to their local I/O addresses.

Since the initialization of ROM or RAM virtual modules is done by the system software, these modules must strictly conform to certain standards, which are described next.

**Special Requirements for ROM/EPROM Plug-In Modules.** The single control register for a ROM (EPROM) plug-in module is at the same location as its identification register, local I/O address "x0000". The value written to this register can select one of up to eight installed ROMs:

0----- Module Disabled.

1----XXX Module Enabled, Bank XXX Selected (000-111).

The protocol for determining whether or not a ROM is installed in the selected ROM socket involves initializing the module's data bus to zero (by reading the identification register), and then reading from the first memory location of the ROM (or reading the floating bus if no ROM is installed in the selected ROM socket). The first memory location of ROMs are specifically non-zero, so that it can be determined whether or not a ROM is installed in a particular socket position (as selected by the control register contents).

The use of this procedure requires that the ROM data bus not be subject to current leakage which would pull the bus to a logic "1" voltage level when it is not being actively driven. It is therefore recommended that the design of any ROM drawer include pull-to-ground resistors on the data bus of the module (typically 100K-ohm), thus helping to assure that the data bus retains a "0" state between the time when the identification register is read and when the address location of the first byte of the ROM is read.

**Special Requirements for RAM Plug-In Modules.** The presence of a control register for a block of RAM does not imply that any RAM components are installed for

that block. However, RAM must be installed in complete blocks of 128K bytes, and these blocks must be installed such that their respective control registers are consecutively located in the local I/O address space. In other words, the first block of RAM installed in a virtual module must be installed in the position controlled by a control register located at local I/O address "x0000" (block 0), the second at control location "x0002" (block 1), etc. Up to eight 128K-byte blocks of RAM can be installed in each virtual module, providing the potential for a plug-in module (two virtual modules) to contain up to 2M bytes of RAM.

The values written to the control registers determine the lower memory boundaries for the corresponding 128K-byte blocks of RAM:

0----- Module Disabled.

1---XXX- Module Enabled, Lower Boundary for Block set at (XXX x 128K).

For reasons similar to those described in the preceding section, RAM modules should include 100K-ohm pull-to-ground resistors on each data bus where RAM may be installed.

**Modes of Operation.** Plug-in devices should respond appropriately to three operating conditions that are implemented by the mainframe:

**Awake Mode:** The DSLEEP\* and SLEEP\* lines are both inactive (high voltage) so both supplies (VccS and VccDS) are at 5 volts. The CPU is active. Plug-in cards should be capable of handing long delays without bus cycle access, when the CPU is halted. During 80C86 interrupt acknowledge cycles an I/O space cycle occurs. BAIE pulses normally, and the address is undetermined. DEN\* and DT/R\* pulse low normally. BRD\* and BWR\* remain at high voltage levels. The addressed plug-in is free to drive the data bus at this time.

Plug-in cards must handle occasional glitches to ground on the BRD\* line after BRD\* has risen high normally at the end of a bus cycle.

**Sleep Mode:** To initiate sleep mode, the PPU drives the SLEEP\* line from 5V to 0V. This resets the CPU (80C86). After 20 microseconds, power supply sequencing begins: VccS floats, and VccDS goes from 5V to 3.25V. VccDS reaches 3.25V after about 10 milliseconds, and VccS drops to less than 1V after about 400 milliseconds.

In this state, the CPU is unpowered. Plug-in devices should reduce power consumption to standby levels (in the microamp range). The plug-in bus takes on the following state in sleep mode:

LA19-LA0: Floating (with 100K-ohm passive pulldowns to ground).  
LM/IO\*: Floating (with 100K-ohm passive pulldown to ground).  
LBHE\*: Floating (with 100K-ohm passive pulldown to ground).  
D15-D0: Floating (with 100K-ohm passive pulldowns to ground).  
BALE: Floating (with 100K-ohm passive pulldown to ground).  
BRD\*: Floating (with 100K-ohm passive pulldown to ground).  
BWR\*: Floating (with 100K-ohm passive pulldown to ground).  
DEN\*: Floating.  
DT/R\*: Floating.  
DSLEEP\*: Driven to high voltage (3.25V nominal).  
SLEEP\*: Driven to low voltage (0V nominal).  
IOCS\*: Floating (with 100K-ohm passive pulldown to ground).  
READY: Floating.  
INT\*: 47K-ohm series resistance to DSLEEP\*.  
VccDS: 3.25 volts nominal.  
VccS: Floating.

The system remains in sleep mode until the  key is pressed or a system interrupt is generated (keyboard interrupt, RS-232 ring, modem ring, plug-in interrupt, or real-time clock interrupt). The wakeup sequence depends upon the condition at that time: no plug-in module has been removed, or a plug-in module has been removed.

If *no* plug-in module has been removed, VccS energizes to 5V, and VccDS rises from 3.25V to 5V. VccS reaches 5V after about 5 milliseconds, and VccDS reaches 5V in about 8 milliseconds. After 200 milliseconds, the SLEEP\* line goes high, and the CPU begins executing instructions.

If a plug-in module *has* been removed, it triggers certain events at that time. DSLEEP\* is driven from 3.25V to 0V to reset any module that's inserted and minimize latchup. This signal also resets all devices operating on VccDS (3.25V) and disables all system interrupts. The  key can wake up the system only after a module has been installed in the port. When the  is pressed, DSLEEP\* is raised to 3.25V, then the normal wakeup sequence occurs.

Sleep mode is also active when power is first applied to the system by the battery jumper. At this time, SLEEP\* is at 0V, and DSLEEP\* follows VccDS. The system doesn't wake up until after both plug-in ports are occupied and the  key is pressed.

Then DSLEEP\* pulses low for 80 microseconds, and the normal wakeup sequence occurs.

2

Stop Mode: Removal of either one of the plug-in drawers while the system is awake causes the mainframe to enter stop mode and shut down all of its power supplies. The power supplies remain shut down until both plug-in drawers are plugged in again and the **(S)** key is pressed. Then the normal wakeup sequence occurs.



# 3

# Mechanical Design

---

## 3.1 Introduction

This chapter presents the mechanical specifications for the mainframe and for the printed-circuit boards that connect to it:

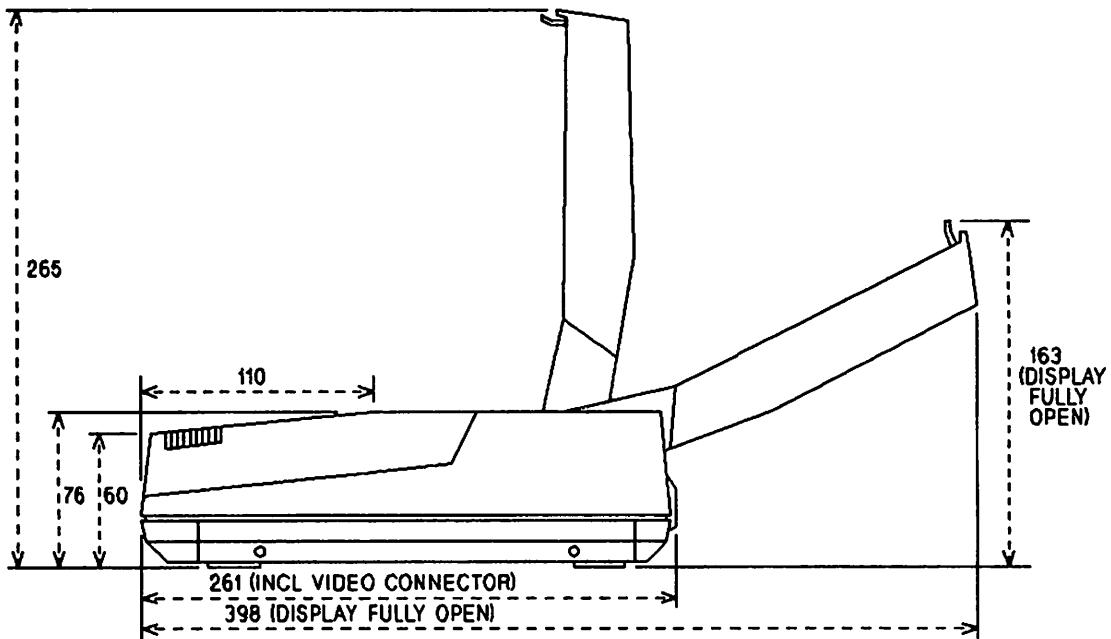
- Modem printed-circuit board.
- Plug-in printed-circuit board.

## 3.2 Mainframe

Figure 3-1 describes the outside dimensions of the Portable PLUS mainframe. The keyboard is integrated into the main body of the unit. The display housing hinges from the top-rear surface of the main body, allowing the user to adjust its angle for optimum viewing. In its closed position, the display housing covers the keyboard, providing protection for the display and keyboard.

The battery cover at the back of the computer provides access to the rechargeable battery and the video connector. Connectors for the recharger, HP-IL interface, serial interface, and optional modem are accessible at the back of the case.

**Figure 3-1. Mainframe Dimensions**



DIMENSIONS ARE IN MILLIMETERS.

OVERALL WIDTH WHEN VIEWED FROM FRONT: 330 MM.

### 3.3 Modem

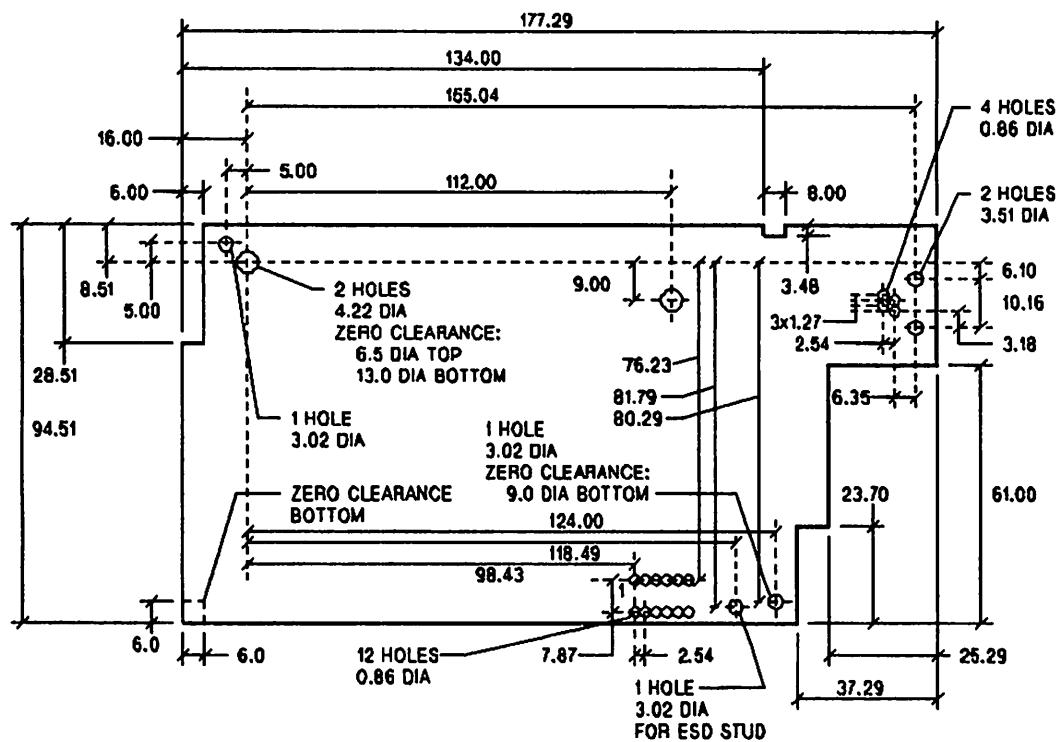
The modem interface, located inside the mainframe assembly, provides a connection to an add-on printed-circuit board. This board connects to the motherboard and extends to the modem cutout at the back of the case.

The "modem" PC board should have a 12-pin connector that connects to the motherboard. The modem board is installed in the bottom case (without using the two plastic spacers that are otherwise installed in that location). The case screws secure the board in place. Figure 3-2 shows the physical specifications for the modem PC board. (Refer to "Modem Connector" in chapter 2 for information about signals at the connector.)

The PC board should have an electrostatic discharge (ESD) ground trace around its perimeter. This trace should be wide (at least 50 mils), unmasked, and on the component side of the board. It should overlap the mounting screw holes and the ESD ground stud hole. A ground strap connects the ESD ground of this board to that of the motherboard. The ground strap is connected using studs and flanged nuts. It is essential that the ESD ground be separate from the logic signal ground.

Refer to appendix I for part numbers.

**Figure 3-2. Modem PC Board**



**DIMENSIONS ARE IN MILLIMETERS.**

**COMPONENT SIDE IS SHOWN.**

NOMINAL BOARD THICKNESS: 1.59 MM.

**COMPONENT HEIGHT RESTRICTIONS VARY WIDELY.**

**THE LAYOUT MUST BE VERIFIED IN THE UNIT.**

### 3.4 Plug-In Ports

Plug-in expansion modules have three primary components: the plastic housing, the metal protective cover with ground spring, and the printed circuit assembly. Variations of these components may include multiple printed circuit assemblies, modified housing, and/or modified cover to allow for supplementary electrical or physical access to the circuitry.

Each plug-in port accepts one drawer, which is connected externally to the mainframe system. A printed-circuit board can be installed in a blank drawer and then installed in a plug-in port.

The "plug-in" board should have a 62-pin connector that connects to the mainframe.

The metal lid for each drawer acts as a charge sink for electrostatic discharge (ESD). The lid is directly exposed to discharge at the bottom edge of the front face. To avoid damage, components on the plug-in board should have high-frequency impedance to the metal lid. The lid is retained by four screws (ISO M2 x 0.4 - 6g).

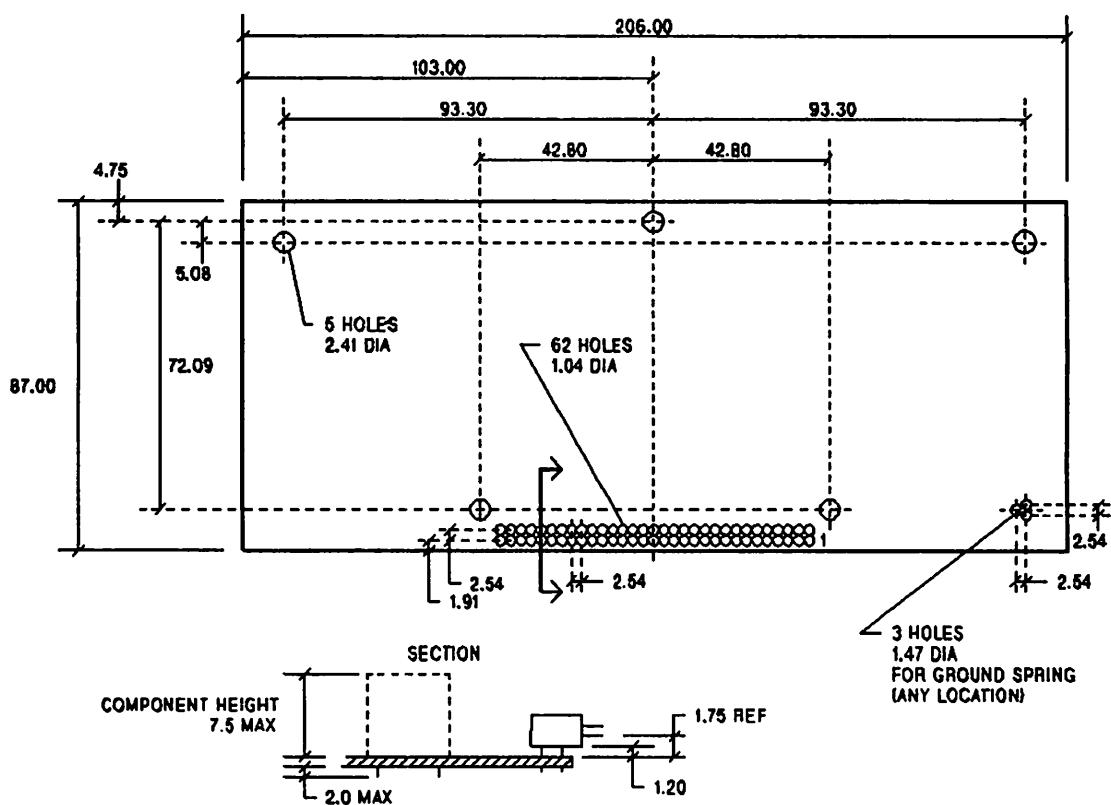
Before the pins of plug-in connector make contact with the memory board connector, the metal lid makes contact with the mainframe's ESD-ground spring. This equalizes the potential of the metal lid and the mainframe ground. In addition, the logic grounds of the mainframe and the plug-in board should be at the same potential when the board is plugged in. This can be achieved by including on the board a ground spring in series with a long, narrow logic-ground trace (a 12-mil trace 3.5 inches long is sufficient). The spring should contact the metal lid. The trace inductance is sufficient to retard the high-frequency electrostatic discharge; it also keeps the mainframe and plug-in logic grounds at the same dc potential.

Plug-in drawers with external cable connections (or I/O ports) should connect the shield ground to the plug-in's metal lid, or to the high inductance trace near the lid-shield connection--not to the plug-in logic ground.

Refer to appendix I for part numbers.

Figure 3-3 shows the physical specifications for the plug-in PC board. (Refer to "Plug-In Ports" in chapter 2 for information about signals at the connector.)

Figure 3-3. Plug-In PC Board



DIMENSIONS ARE IN MILLIMETERS.  
COMPONENT SIDE IS SHOWN.  
NOMINAL BOARD THICKNESS: 1.59 MM.



# Resetting the Portable PLUS

## 4.1 Introduction

Under certain conditions a reset to the Portable PLUS may be desirable. After changing a plug-in drawer or creating a CONFIG.SYS file, a reset is necessary so that the system will re-boot and recognize the new configuration.

Occasionally the computer may get into a state where it will not respond to particular keys and a reset may be necessary. In this case, there are a few things to try before resorting to resetting the computer.

- If the display is off and will not turn on, it is possible that the batteries are low. Plug in the recharger and press a key to turn on the display. Do not attempt to operate the computer on battery power alone until it has been recharged for at least one hour. (We recommend that if the battery charge falls to the 5% level, the computer should be fully recharged before operating it on battery power again.)
- If the computer appears to be "locked up" in the middle of an application, try pressing the **(CTRL) C** key combination. MS-DOS watches for this keystroke to break out of the current process.

As an alternative, press **Shift** **Break**. This keystroke clears the input buffer and then enters ^C (**CTRL** **C**). Sometimes MS-DOS examines only the first character in the input buffer.

- Some applications may watch for the **(CTRL) Stop** key combination to interrupt the current process. When these keys are pressed, the BIOS Break Interrupt (Int 1Bh) is called and may terminate the current application. This interrupt is a hook for applications so its use will vary.

## **4.2 Reset Options**

### **4.2.1 Reset via Shift CTRL Break**

The Portable PLUS may be reset by pressing the Shift CTRL Break key combination. Drive A: (the Edisc) is not affected by this reset, but user memory is, and any work not stored on a disc is lost. The default drive for the re-boot is A:.

### **4.2.2 Reset via Shift CTRL Extend char Break**

The computer may also be reset by pressing the Shift CTRL Extend char Break key combination. Drive A: is not affected by this reset, but again any work not stored on a disc is lost. The default drive during the re-boot is drive B:.

During this re-boot, a CONFIG.SYS file stored on drive A: is ignored since MS-DOS will search the default drive B:. If the shell is changed to boot COMMAND.COM, an AUTOEXEC.BAT file stored on A: is ignored during the re-boot since the default drive is B:. (Refer to "The CONFIG.SYS File" in chapter 11.) If PAM is the boot shell, it will always change the default drive to A: and execute AUTOEXEC.BAT if found on drive A:.

### **4.2.3 Reset via ①**

An alternate method to reset is useful when an application has "locked out" the entire keyboard. Reset the Portable PLUS by pressing and holding down the ① key for an extended period of time. The display will darken to its maximum and then, after a few seconds, turn off. Then press ① again to turn the computer back on. Drive A: is not modified but any work not stored on a disc is lost. The default drive during the re-boot is drive A:.

#### **4.2.4 Reset via the Reset Button**

This last method has the most drastic results of all the resets. Pressing the reset button located inside the battery compartment will perform a full reset.



**Caution**

Pressing the reset button erases the contents of drive A: (the Edisc) and all user memory. All data stored on drive A: will be lost. All the PAM Configuration values will be reset to their default values.

A warning message will appear after pressing the reset button to indicate that all of memory has been lost. Press the **(f1)** key to format drive A: and continue.

### **4.3 Re-Boot Screen**

#### **4.3.1 Memory Lost Message**

After each reset the first eight sectors of drive A: are evaluated for checksum errors. If no error is found, the re-boot continues and the standard re-boot display as described in the next section is shown. If there are errors found, the following message flashes on the display:

**WARNING: Memory Lost! Press [f1] to reformat drive A:  
(destroying data) or press the space bar to continue  
without reformatting (data errors on A: may result).**

The user has two options. If the information on drive A: is not needed, then the **(f1)** key should be pressed, formatting drive A: and clearing the errors.



## Caution

Pressing the **DEL** key at this time will result in all the information stored on drive A: to be erased.

The alternative to this is to press the space bar to continue and boot from drive B:. The user can then attempt to salvage some of the information from drive A: (if possible) before formatting it to clear the error(s). There is a possibility that the bad sectors on drive A: will interfere with the boot completely if those sectors must be accessed by an AUTOEXEC.BAT program on drive A:. In this case, the only option is to format drive A: at boot.

4

### 4.3.2 Standard Re-Boot Display

The standard re-boot display appears whenever the Portable PLUS is reset. This display only appears for a moment and should look *similar* to the one shown below.

```
HP 45711 Revision: A          PPU: A Configuration: A
(c) COPYRIGHT Hewlett-Packard Co. 1985
Serial Number: 2522A01142 B
Rom IDs: AAAAAA

MS-DOS version 2.11
Copyright 1981,82,83 Microsoft Corp.
```

The following information can be found on the re-boot display.

- **Product Number.** The product number of the Portable PLUS is split into two parts. The first part is the string "HP 45711" in the first line of the display. Appended to that is the last letter or letters in the serial number. The full product number in this example is HP 45711B.
- **Overall System ROM Revision.** This can be read from the first line following the word "Revision:". In this example, the overall system ROM revision is A.

- **PPU Revision.** The PPU revision is on the first line of the display following the word "PPU:". In this example, the PPU revision is A.
- **Configuration EPROM Revision.** The configuration EPROM revision is the last letter on the first line. In this example, the configuration EPROM revision is A.
- **Serial Number.** The serial number is shown on the third line of the display. This number is unique for each mainframe. The first four digits of the serial number are a date code. A single letter follows, which is the manufacturing site code. The remaining digits make up the sequence number. The last letter or letters are the last part of the product number.
- **ROM IDs.** The six individual ROM IDs are shown in the fourth line. They correspond to the six ROMs that contain the system BIOS, MS-DOS, and built in applications (such as Print and TERM). In this example, each ROM has the ID A.
- **MS-DOS Boot Information.** The last two lines are displayed when MS-DOS 2.11 is booted.



# 5

# BIOS Interrupts

---

## 5.1 Introduction

The BIOS (Basic Input/Output System) provides control of the system I/O devices. Software interrupts enable you to access the BIOS while remaining isolated from the actual system hardware. This prevents programs from being adversely affected by changes in the hardware, while providing quick and simple access to commonly desired functions.

Software interrupts are generally of two types: *services* and *hooks*. Service interrupts are events that a *program* enables and controls. Hooks are events that the *system* initiates, and whose response can be directed by a program.

An application may take over any BIOS interrupt to change the function of that interrupt. If this is done, *the interrupt should always be restored to its previous function when the application terminates*. Care should be taken when writing interrupt service routines to avoid interrupt nesting (one interrupt happening inside the service routine for another interrupt).



### Caution

An interrupt service routine should not invoke MS-DOS or other functions that result in calls to the built-in device drivers. Only a limited few of the BIOS interrupt functions are allowed within an interrupt service routine.

---

Assume an application has taken over the modem ring interrupt, and wishes to display the word "RING" if a ring interrupt ever happens. If the service routine itself issues an MS-DOS call to display the word "RING", we are effectively nesting one (software) interrupt inside another (hardware) interrupt. This could lead to unexpected results.

The problem is further complicated by the fact that MS-DOS typically enables interrupts within its own services.

The appropriate way to handle this situation is to write the new modem ring service so that it sets a flag upon receipt of a modem ring. The interrupt routine should then

clear the interrupt and return to the application. The application should test the flag at frequent intervals and, if it is set, clear the flag and issue an MS-DOS call to display the word "RING".



Unless noted otherwise, all registers should be preserved by any application that intercepts an interrupt.

**Note**

Table 5-1 lists the system hardware and BIOS interrupts.

**Table 5-1. Hardware and BIOS Interrupts**

Interrupt	Description	Type
<b>8086 Dedicated Interrupts</b>		
0h	Divide by Zero	Hook
1h	Single Step	Hook
2h	Nonmaskable	Hook
3h	Breakpoint	Hook
4h	Overflow	Hook
<b>IBM-Compatible Interrupts</b>		
Refer to appendix A for compatibility descriptions of these interrupts.		
5h	Print Screen	Service
10h	Video I/O	Service
11h	Equipment	Service
12h	Memory	Service
14h	Communications	Service
16h	Keyboard I/O	Service
17h	Printer	Service
19h	Re-Boot	Service
1Ah	Time of Day	Service
1Bh	Keyboard Break	Hook
1Ch	Timer Tick	Hook
1Fh	Graphics Character Extensions	Data Pointer

Table 5-1. Hardware and BIOS Interrupts (Continued)

**MS-DOS Interrupts**

20h	Program Terminate	Service
21h	Function Request	Service
22h	Terminate Address	Hook
23h	( <b>CTRL</b> ) ( <b>C</b> ) Exit Address	Hook
24h	Fatal Error Abort Address	Hook
25h	Absolute Disk Read	Service
26h	Absolute Disk Write	Service
27h	Terminate But Stay Resident	Service
28h	(Reserved)	
.	.	
.	.	
.	.	
3Fh	(Reserved)	

**Pseudo-Hardware Interrupts**

*There is only one true hardware interrupt (FFh). In response to it, the BIOS polls each device and then issues an appropriate "pseudo-hardware" interrupt.*

40h	Modem Output Port	Hook
41h	(Reserved)	
42h	Modem Ring or Carrier	Hook
43h	Timer 2	Hook
44h	I/O Port 1	Hook
45h	I/O Port 2	Hook
46h+	Alarm	Hook
47h+	Death/Battery Cutoff	Hook
48h+	Heartbeat Timer	Hook
49h+	Keyboard	Hook
4Ah+	Serial Output Port	Hook
4Bh+	Serial Ring or Carrier	Hook
4Ch+	HP-IL IRQ	Hook
4Dh+	Low Battery	Hook
4Eh	Modem Input Port	Hook
4Fh+	Serial Input Port	Hook

**Table 5-1. Hardware and BIOS Interrupts (Continued)**

<b>Firmware Services and Hooks</b>		
50h	System Services	Service
51h	(Reserved)	
52h+	Modifier Keys	Hook
53h+	<u>(Print) Key</u>	Hook
54h+	HP-IL Primitives	Service
55h+	Sleep	Service
56h+	<u>(Menu) Key</u>	Hook
57h+	<u>(User/System) Key</u>	Hook
58h+	<u>(Break) Key</u>	Hook
59h+	Ring Enable	Service
5Ah	(Reserved)	
5Bh	(Reserved)	
5Ch	(Reserved)	
5Dh	AUX Expansion	Hook
5Eh	CON Expansion	Hook
5Fh	Fast Video	Service
<b>Other Interrupts</b>		
60h	Available to Application Software	
.	.	
.	.	
.	.	
FEh	Available to Application Software	
FFh+	Hardware	Hook

+ These interrupts are compatible with the HP 110 Portable Computer.

---

## 5.2 Print Screen Interrupt (Int 5h)

This interrupt executes the system screen print code. The PAM Printer Mode setting determines how the screen contents should be printed; the display data is formatted accordingly and sent to the PAM Printer Interface device. While the screen print is in progress, a status byte at location 40H:100H is set to a 1. If an error occurs during the print operation, the byte is set to FFh. When the operation terminates successfully, the status byte is set to a 0.

---

## 5.3 Video I/O Interrupt (Int 10h)

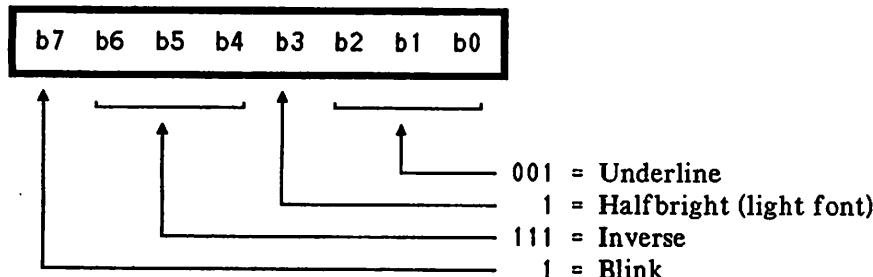
Interrupt 10h provides a modified subset of IBM's Video I/O utilities. Principal differences are due to the fixed size (480 x 200) of the liquid crystal display and custom LCD controller hardware.

Alpha characters can be written and read in graphics mode. The characters are formed from the first 128 characters of the base character set. Graphics characters in the range 128 to 255 come from a user-supplied 1K-byte font table pointed to by the vector for interrupt 1Fh. This vector is initially 0000:0000. Each character of the table consists of eight bytes, representing the pixel patterns of the first (top) through last (bottom) dot-rows in an 8-by-8 character matrix. (Only the first six bits of each byte should be used.) For graphics applications that directly read and write display RAM, display RAM starts at 8000:0000, with successive dot-rows beginning at 64-byte intervals. (Note that while 64 bytes per line times 8 pixels per byte equals 512 pixels per line, the LCD can show only 480 pixels per line; the last four bytes of each line are not used.)

For alpha applications that directly read and write display RAM, alpha page zero starts at 8000:0200 and alpha page one starts at 8000:1B00 (or, if you prefer, 8020:0000 and 8020:1900.) The high byte of the display RAM offset increments with successive character rows (8000:0200 starts page 0 row 0, 8000:0300 starts page 0 row 1, etc.), while the low byte of the offset addresses the columns within each line. Each character requires one word of display RAM: the character code is in the low byte and its attributes are in the high byte. (Note that this addressing scheme permits access to 128 characters per line, but only 80 characters are visible. The remaining 48 words of each line are used for font storage.)

When an Interrupt 10h function requests or returns attribute bytes, their format is not the same as the format in display RAM. Attribute mapping is necessary to provide some degree of IBM compatibility. The format of the Interrupt 10h attribute byte is shown in figure 5-1.

Figure 5-1. Interrupt 10h Attribute Byte



5

The Video I/O Interrupt functions for this computer are described in table 5-2. Specify the desired function code in AH, with additional parameters passed in other registers as indicated. All segment registers plus BX, CX, and DX are preserved unless otherwise noted; other registers may be destroyed.

**Table 5-2. Video I/O Interrupt 10h Functions**

Function	Description
AH=0 (00h)	<b>Set Mode</b>
	<p>Initializes the specified display mode.</p> <p><b>Specify:</b></p> <ul style="list-style-type: none"> <li>AL=0 80 x 25 Alpha. (IBM: 40 x 25 b&amp;w)</li> <li>1 80 x 25 Alpha. (IBM: 40 x 25 color)</li> <li>2 80 x 25 Alpha. (IBM: 80 x 25 b&amp;w)</li> <li>3 80 x 25 Alpha. (IBM: 80 x 25 color)</li> <li>4 480 x 200 Graphics. (IBM: 320 x 200 color)</li> <li>5 480 x 200 Graphics. (IBM: 320 x 200 b&amp;w)</li> <li>6 480 x 200 Graphics. (IBM: 640 x 200 b&amp;w)</li> <li>&gt;6 80 x 25 Alpha.</li> </ul>
AH=1 (01h)	<b>Set Cursor Type</b>
	<p>Determines the type of alpha cursor to be displayed.</p> <p><b>Specify:</b></p> <ul style="list-style-type: none"> <li>CL If CL&lt;CH, turn cursor off.</li> <li>CH≤3 Box cursor.</li> <li>&gt;3 Underscore cursor.</li> </ul>
AH=2 (02h)	<b>Set Cursor Position</b>
	<p>Position the cursor for a specified page (each alpha page has its own unique cursor).</p> <p><b>Specify:</b></p> <ul style="list-style-type: none"> <li>BH = Page number (0-1, must be 0 for graphics).</li> <li>DH = Row number (0 is top of screen, 24 is bottom).</li> <li>DL = Column number (0 is left margin, 79 is right).</li> </ul>

**Table 5-2. Video I/O Interrupt 10h Functions (Continued)**

AH=3 (03h)	<b>Read Cursor Position</b>  Returns the cursor position for a specified page. Specify: BH = Page number (0-1, must be 0 for graphics). Returns: CH = Current cursor type (as set by function AH=1). DH = Row number (0 is top of screen, 24 is bottom). DL = Column number (0 is left margin, 79 is right).
AH=4 (04h)	<b>Read Light Pen Position</b>  Because there is no light pen, this function always returns AH=0. Returns: AH=0 Light pen switch not activated.
AH=5 (05h)	<b>Select Active Display Page</b>  This function is valid only in alpha mode. Specify: AL = New page number (0-1).
AH=6 (06h)	<b>Scroll Active Page Up</b>  Scrolls the display upwards. (No response in Graphics mode.) Specify: AL = Number of lines to scroll (0 means blank entire window). BH = Attribute for scrolled-in blank lines. CH = Row number, top of scroll region (0-24). CL = Column number, left side of scroll region (0-79). DH = Row number, bottom of scroll region (0-24). DL = Column number, right side of scroll region (0-79).

**Table 5-2. Video I/O Interrupt 10h Functions (Continued)**

AH=7 (07h)	<b>Scroll Active Page Down</b>  Scrolls the display downwards. (No response in Graphics mode.) <b>Specify:</b> AL = Number of lines to scroll (0 means blank entire window). BH = Attribute for scrolled-in blank lines. CH = Row number, top of scroll region (0-24). CL = Column number, left side of scroll region (0-79). DH = Row number, bottom of scroll region (0-24). DL = Column number, right side of scroll region (0-79).
AH=8 (08h)	<b>Read Attribute/Character at Current Cursor Position</b>  Returns the character and attribute bytes at the current cursor position for a specified page. The cursor does not move. <b>Specify:</b> BH = Display page (ignored in graphics mode). <b>Returns:</b> AL = Character read. AH = Attribute of character (zero in graphics mode).
AH=9 (09h)	<b>Write Attribute/Character at Current Cursor Position</b>  Displays a new character and attribute at the current cursor position for a specified page. The cursor does not move. <b>Specify:</b> AL = Character to write. BL = Attribute of character (in alpha mode). Color of character (in graphics mode; see function 12). BH = Display page (ignored in graphics mode). CX = Count of characters to write.

**Table 5-2. Video I/O Interrupt 10h Functions (Continued)**

AH=10  
(0Ah)

**Write Character Only at Current Cursor Position**

Displays a new character at the current cursor position for a specified page. The attribute byte at that position is left unchanged.  
The cursor does not move.

Specify:

- AL = Character to write.
- BH = Display page (ignored in graphics mode).
- CX = Count of characters to write.

AH=11  
(0Bh)

**Set Color Palette**

This function is ignored.

AH=12  
(0Ch)

**Write Dot**

Places a graphics dot at the specified position.

Specify:

- AL = New pixel color (0=white, 1=black). If bit 7 is set, the new value is exclusive-ORed with the current pixel value.
- CX = Pixel column number (0-479).
- DX = Pixel row number (0-199).

AH=13  
(0Dh)

**Read Dot**

Reads the graphics dot at the specified position.

Specify:

- CX = Pixel column number (0-479).
- DX = Pixel row number (0-199).

Returns:

- AL = Current pixel value (0=white, 1=black).

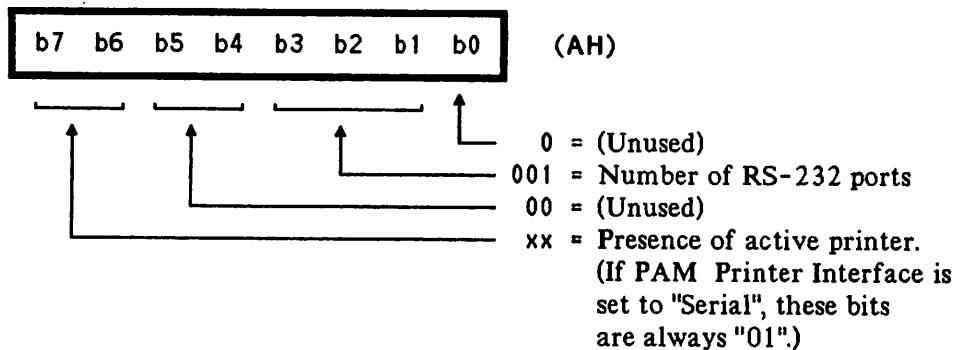
**Table 5-2. Video I/O Interrupt 10h Functions (Continued)**

AH=14 (0Eh)	<b>Write Teletype to Active Display Page</b>  This function provides a teletype-like interface to the display. The character in AL is displayed at the current cursor position, and the cursor advances (right) to the next position. If the cursor moves past the end of the line, a carriage return and line feed are performed. Backspace, carriage return, line feed, and bell are handled as commands rather than displayable characters. <b>Specify:</b> AL = Character to output. BL = Character color if in graphics mode (see function 12).
AH=15 (0Fh)	<b>Current Video State</b>  Returns display information. <b>Returns:</b> AL = Current mode (as set by function 0). AH = Number of character columns on the screen (80). BH = Currently active display page (0-1).

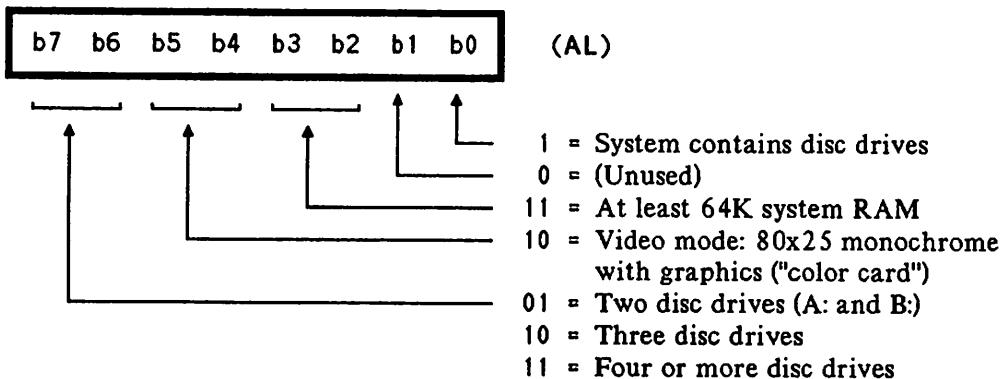
## 5.4 Equipment Check Interrupt (Int 11h)

When an Int 11h instruction is executed, system configuration information is returned in register AX according to the current system parameter values.

The following status bits are returned in AH:



The following status bits are returned in AL:



---

## **5.5 Memory Interrupt (Int 12h)**

The Memory Interrupt can be used by an application to determine the total amount of memory in the system, excluding ROM and internal Edisc. The interrupt returns in register AX the number of 1K-byte blocks of system RAM.

---

## **5.6 Communications Interrupt (Int 14h)**

The AUX driver uses this interrupt for communications with the serial port, modem, or current AUX device (as specified by PAM). The HP 82164A HP-IL/RS-232-C Interface is *not* supported by this interrupt.

Table 5-3 lists the functions provided by this interrupt.

**Table 5-3. Communications Interrupt 14h Functions**

Function	Description																																																																																	
<b>AH=0 (00h)</b>	<p><b>Initialize Communications Parameters</b></p> <p>Defines communications parameters for the specified port.</p> <p>Specify:</p> <p>AL = Datacom initialization byte in the following form:</p> <p>The diagram shows the AL byte structure as a horizontal rectangle divided into 8 bits labeled b7 through b0 from left to right. Below the rectangle, arrows point upwards to each bit position. To the right of the rectangle, the label '(AL)' is enclosed in parentheses.</p> <table border="0"> <tr> <td>b7</td> <td>b6</td> <td>b5</td> <td>b4</td> <td>b3</td> <td>b2</td> <td>b1</td> <td>b0</td> <td>(AL)</td> </tr> <tr> <td colspan="8"></td> <td></td> </tr> </table> <p>Bit meanings:</p> <ul style="list-style-type: none"> <li>10 = 7-Bit Word</li> <li>11 = 8-Bit Word</li> <li>0 = 1 Stop Bit</li> <li>1 = 2 Stop Bits</li> <li>x0 = No Parity</li> <li>01 = Odd Parity</li> <li>11 = Even Parity</li> </ul> <p>Baud rates:</p> <ul style="list-style-type: none"> <li>000 = 110 Baud</li> <li>001 = 150 Baud</li> <li>010 = 300 Baud</li> <li>011 = 600 Baud</li> <li>100 = 1200 Baud</li> <li>101 = 2400 Baud</li> <li>110 = 4800 Baud</li> <li>111 = 9600 Baud</li> </ul> <p>DX=0 Initialize the serial port.</p> <ul style="list-style-type: none"> <li>=1 Initialize the modem.</li> <li>&gt;1 Initialize the current AUX device.</li> </ul> <p>(If DX&gt;1 and AUX device is 82164, serial port will be used.)</p> <p>Returns:</p> <ul style="list-style-type: none"> <li>AH = Port data status (refer to function AH=3).</li> <li>AL = Port handshake status (refer to function AH=3).</li> </ul>	b7	b6	b5	b4	b3	b2	b1	b0	(AL)																																																																								
b7	b6	b5	b4	b3	b2	b1	b0	(AL)																																																																										

**Table 5-3. Communications Interrupt 14h Functions (Continued)**

AH=1 (01h)	<b>Send Character</b>  Sends the specified character over the selected communications line. <b>Specify:</b> AL = Character to send. DX=0 Send character to the serial port. =1 Send character to the modem. >1 Send character to the current AUX device. (If DX>1 and AUX device is 82164, serial port will be used.)  <b>Returns:</b> AL = Character sent. AH = If bit 7 is set, an error has prevented the character from being transmitted, and it should be sent again.
AH=2 (02h)	<b>Read Character</b>  Reads a character from the specified datacom device buffer. <b>Specify:</b> DX=0 Read character from the serial port. =1 Read character from the modem. >1 Read character from the current AUX device. (If DX>1 and AUX device is 82164, serial port will be used.)  <b>Returns:</b> AL = Character read (only if bit 7 of AH is "0"). AH = If bit 7 is set, no character was available.

**Table 5-3. Communications Interrupt 14h Functions (Continued)**

AH=3 (03h)	<b>AUX Status</b>  Returns the status of the specified communications port. Specify: DX=0 Request status for the serial port. =1 Request status for the modem. >1 Request status for the current AUX device. (If DX>1 and AUX device is 82164, serial port will be used.)  Returns: AH = Port data status: Bit 7: If set, the operation was not successful. Bit 6: If set, the transfer shift register is empty, and the next character can be sent. Bit 5: (Unused). Bit 4: If set, a break condition currently exists on the communications line. Bit 3: If set, a framing error has occurred. Bit 2: If set, a parity error has occurred. Bit 1: If set, a data overrun error has occurred. Bit 0: If set, data is available to be read.  AL = Port handshake status: Bit 7: If set, RLSD line is true. Bit 6: If set, Ring line is true. Bit 5: If set, DSR line is true (serial port only). Bit 4: If set, CTS line is true (serial port only). Bit 3: If set, RSLD line has changed state since last Int 14h AUX Status call. Bit 2: If set, Ring line has changed state since last Int 14h AUX Status call. Bit 1: If set, DSR line has changed state since last Int 14h AUX Status call. Bit 0: If set, CTS line has changed state since last Int 14h AUX Status call.
---------------	--

## 5.7 Keyboard I/O Interrupt (Int 16h)

Keyboard I/O Interrupt 16h is supported in the limited manner described in table 5-4. Specify the desired function code in AH. Only AX and flags change; all other registers are preserved.

Table 5-4. Keyboard I/O Interrupt 16h Functions

Function	Description
AH=0 (00h)	<b>Read Character</b>  Reads the next character from the keyboard queue. Note that in Scancode Mode the key queue will contain scancodes rather than ASCII keycodes. Scancode and keycode information cannot be returned simultaneously. If the queue is empty, this function will wait for a key to be hit before returning. <b>Returns:</b> AL = Next character from the keyboard queue. AH = 0.
AH=1 (01h)	<b>Read Character (Nondestructive)</b>  Reads next character from keyboard queue without removing it from queue. If the queue is empty, this function will return immediately with the Zero flag (ZF) set. <b>Returns:</b> ZF=1 No character is available to be read. 0 Character code is available, and is in AX.

**Table 5-4. Keyboard I/O Interrupt 16h Functions (Continued)**

AH=2  
(02h)

**Read Shift Status**

This function returns the current states of the three modifier keys, plus on/off status for Insert Character Mode, Caps Lock, and the Numeric Keypad. Note that the two shift keys are functionally identical and cannot be independently read.

Returns:

AL = Current shift status with bits set as follows:

Bit 7: Insert Character Mode is active.

Bit 6: Caps Lock is turned on.

Bit 5: Numeric keypad is active.

Bit 4: (Unused.)

Bit 3: **Extend** key is depressed.

Bit 2: **CTRL** key is depressed.

Bit 1: **Shift** key is depressed.

Bit 0: **Shift** key is depressed (same as Bit 1).

## 5.8 Print Byte Interrupt (Int 17h)

Interrupt 17h provides a low-level method of sending one byte of data to the PAM Printer Interface device. Table 5-5 describes the various Print Byte Interrupt functions. Specify the desired function code in AH; all other registers are preserved. (*IBM compatibility: The contents of DX, which normally indicate which printer is to be addressed, are ignored. Also, the bytes at locations 0040:0008 through 0040:000D are unused, rather than containing the base addresses of printer cards.*)

Table 5-5. Print Byte Interrupt 17h Functions

Function	Description
AH=0 (00h)	<b>Print Character</b>  The character in AL is sent to the current PAM Printer Interface device. <b>Specify:</b> AL = Character to be sent to printer. <b>Returns:</b> AH=01h if an error occurred. D0h if the character was sent successfully.
AH=1 (01h)	<b>Initialize Printer</b>  The printer is configured as necessary for subsequent communications. <b>Returns:</b> AH=01h Initialization failed. D0h Printer ready.
AH=2 (02h)	<b>Return Status</b>  Returns a status byte indicating whether or not subsequent printer communications is possible. This is essentially the same function as AH=1. <b>Returns:</b> AH=01h Printer is not accessible. D0h Printer is accessible.

---

## 5.9 Reboot Interrupt (Int 19h)

The Reboot Interrupt is roughly the programmatic equivalent of holding down the contrast key for more than 15 seconds. The PPU is told to reset the system; all hardware (except for RAM) is subsequently reset.

---

## 5.10 Time Of Day Interrupt (Int 1Ah)

The Time Of Day Interrupt provides a means by which an application can perform general purpose timing with approximately 1/18 second resolution. This interrupt has no effect on any other part of the system; it is provided solely for application use. If you use this interrupt to set the heartbeat timer to a new value, it will *not* change the system's time-of-day clock (a separate timer maintained by the PPU). The current time reported by MS-DOS, PAM, and the on-screen clock is read from the PPU and has no relation to the heartbeat timer accessed by this interrupt. Table 5-6 describes the functions of Int 1Ah.

Table 5-6. Time Of Day Interrupt 1Ah Functions

Function	Description
AH=0 (00h)	<b>Read Heartbeat Timer</b>  Returns the current double-word contents of the system heartbeat timer. If the timer has overflowed past 24 hours, a 1 will be returned in AL. This function clears the 24-hour overflow flag. <b>Returns:</b> AL = 24-hour overflow flag. CX = High portion of count. DX = Low portion of count.

**Table 5-6. Time Of Day Interrupt 1Ah Functions (Continued)**

AH=1 (01h)	<b>Set Heartbeat Timer</b>  Loads the double-word system timer with a new value. Note that the timer overflows and resets to zero whenever the count increments to exactly 24 hours; if you set a value that is greater than 24 hours (1,555,200 eighteenth-second intervals), the overflow will go undetected and the timer will not be reset. <b>Specify:</b> CX = New high portion of count. DX = New low portion of count.
---------------	---

## 5.11 Keyboard Break Interrupt (Int 1Bh)

A Keyboard Break interrupt is generated whenever, in Alt mode, you press the **CTRL** and **Break** keys. Normally this interrupt flushes the key queue and then puts a ^C (03h) in it. But by taking over Int 1Bh and pointing it at your own interrupt handler, you can perform your own **CTRL** **Break** key processing. When the system branches into your new interrupt handler, the state of the three modifier keys (at the time the **Break** key was pressed) are available in AH; AL will contain 0D4h, the Configuration EPROM keymap Local Function code that caused the interrupt to be issued.

This interrupt is invoked by the keyboard driver responding to a keyboard hardware interrupt. All general registers are available when the interrupt branches into your handler; they need not be saved and restored (in general, however, you should always save and restore any registers that you will use in servicing an interrupt.)

## 5.12 Timer Tick Interrupt (Int 1Ch)

Whenever a Heartbeat interrupt occurs, the system Heartbeat interrupt handler invokes the Timer Tick interrupt (1Ch); the Timer Tick vector points at code to be executed on every heartbeat tick (nominally 18 times per second).

Normally this vector points at a dummy IRET instruction. If you take over this vector and point it at your own Timer Tick handler, you should end your routine with an IRET. As with any interrupt handler, you should also save and restore any registers that your routine will use.



### Note

Do not re-enable interrupts within an interrupt handler unless you are very careful! In this particular case, the heartbeat interrupt has not yet been cleared at the time Int 1Ch is called; turning on the interrupt system will cause the heartbeat interrupt to recursively appear!

## 5.13 Graphics Character Extensions (Int 1Fh)

The Video I/O Interrupt (Int 10h) allows you to display alpha characters in graphics mode. The font patterns for character codes in the range 0 to 127 are taken from the first font table in the system ROMs, while the font patterns for character codes in the range 128 to 255 are taken from a font table pointed to by the vector at interrupt 1Fh. At reboot, this vector is initialized to 0000:0000; it is the user's responsibility to point this vector at an appropriate 1K-byte font table.

Each character font in the table is represented by eight bytes of graphic information. An alpha character must fit within a 6x8 cell, so only the high-order six bits of each byte are meaningful. For example, the eight-byte table entry for the letter "E" might look like this:

bit:	7	6	5	4	3	2	1	0	
1st byte:	.	■	■	■	■	■	.	.	= 0F8h
2nd byte:	.	■	■	.	.	.	.	.	= 0C0h
3rd byte:	.	■	■	.	.	.	.	.	= 0C0h
4th byte:	.	■	■	■	■	.	.	.	= 0C0h
5th byte:	.	■	■	.	.	.	.	.	= 0C0h
6th byte:	.	■	■	.	.	.	.	.	= 0C0h
7th byte:	.	■	■	■	■	■	.	.	= 0F8h
8th byte:	.	.	.	.	.	.	.	.	= 000h

The rightmost two bits of each row are not used; the left dot-column and bottom dot-row of the remaining 6x8 matrix are left blank to provide separation between adjacent characters and lines on the display (although you can use the entire 6x8 cell if necessary.)



**Note** The Graphics Character Extensions table is only used in conjunction with Video I/O Interrupt 10h. It is never accessed by Fast Alpha, system services, or standard alpha and graphics CON output.

---

## **5.14 Modem Transmit Interrupt (Int 40h)**

This interrupt is the same as Serial Transmit Interrupt 4Ah, but applies to the built-in Modem interface. It operates in an identical manner, except that the I/O addresses are in the Axh range instead of 4xh.

---

## **5.15 Modem Ring/Carrier Interrupt (Int 42h)**

This interrupt is the same as Serial Ring/Carrier Interrupt 4Bh, but applies to the built-in Modem interface. It operates in an identical manner, except that the I/O addresses are in the Axh range instead of 4xh.

---

## **5.16 Timer 2 Interrupt (Int 43h)**

Interrupt 43H is generated whenever the second interval timer (at I/O addresses 4Eh-5Eh) is enabled and reaches zero. (For more detailed information on the interval timer, refer to "Registers - Interval Timer" in chapter 7). To service the timer interrupt, a "1" must be written to bit 0 of the Timer 2 Control Register (address 58h). This will clear the current interrupt, load the value in the interval reference registers into the counter registers, and restart the timer.

Whenever the modem is on, the BIOS starts Timer 2 running at a rate of 50 ticks per second. The purpose of this is to allow software implementation of the modem's Return-to-Command Mode feature. Whenever the modem is off, Timer 2 is available to application programs. However, any application that takes over Interrupt 43h *must* save the old interrupt vector, and restore it before terminating, or the Return-To-Command-Mode feature will not function properly the next time the modem is used.

---

## 5.17 Plug-in 1 Interrupt (Int 44h)

This hook is called whenever an interrupt is generated by the Plug-in 1 drawer. Plug-in 1 is at configuration I/O address E0h or F0h, and is physically located on the right side of the computer under the Return key.

This interrupt will not occur unless Plug-in 1 interrupts are enabled (refer to the Int 50h "Alter Interrupt Control Register A2h" function). Once the interrupt occurs, the service routine should clear the interrupt using the Clear Interrupt Request Register (see hardware description in Chapter 7).

---

## 5.18 Plug-in 2 Interrupt (Int 45h)

This hook is called whenever an interrupt is generated by the Plug-in 2 drawer. Plug-in 2 is at configuration I/O address C0h or D0h, and is physically located on the left side of the computer under the F1 key.

This interrupt will not occur unless Plug-in 2 interrupts are enabled (refer to the Int 50h "Alter Interrupt Control Register A2h" function). Once the interrupt occurs, the service routine should clear the interrupt using the Clear Interrupt Request Register (see hardware description in Chapter 7).

---

## 5.19 PPU Alarm Interrupt (Int 46h)

If a PPU alarm has been set (via the PPU Set Alarm command), interrupt 46h will be called when that alarm occurs. No special service action is required on the part of the service routine.

The alarm interrupt is primarily used by PAM. For more information on what happens when an alarm occurs, refer to "PAM And Alarms" in chapter 10.

## **5.20 Death/Battery Cutoff Interrupt (Int 47h)**

The Battery Cutoff Interrupt is automatically called when the battery charge drops below the 5 percent charge-remaining level. The default handler for this interrupt forces the computer into a sleep state as quickly as possible to avoid any loss of data and to permit the current application to be resumed once the battery level has come back up.

If the battery cutoff point is reached and the computer is not put to sleep quickly enough, the PPU will automatically shut the system down (and lose the state of the current application). For this reason any application that takes over Int 47h should put the system to sleep (via the Sleep Interrupt 55h) as soon as possible.

## 5.21 Keyboard Interrupt (Int 49h)

All of the keys in the hardware keyboard matrix generate a hardware interrupt on both upward and downward transitions. (Note that the Shift, CTRL, and Extend modifier keys, plus the contrast key, are not in the matrix.) The system keyboard interrupt handler services these transitions by analyzing the states of the matrix and modifier keys, and then performing an appropriate action.

If you plan to take over the keyboard hardware interrupt, your interrupt handler should have the following form:

Kbd\$INT: *Save all registers*

```
    mov    al,02h      ;Clear and disable
    out    0B8h,al     ; heartbeat interrupts
    mov    ax,80h      ;Clear this keyboard
    out    0A0h,al     ; matrix interrupt
    mov    bx,18h      ;Disable keyboard
    int    50h         ; matrix interrupts

Process the interrupt
    mov    ax,180h      ;Re-enable keyboard
    int    50h         ; matrix interrupts
    mov    al,1          ;Re-enable
    out    0B8h,al      ; heartbeat interrupts

Restore all registers
    iret
```

The system keyboard interrupt handler should never be invoked via a software interrupt. For further information about using the keyboard, refer to "Multi-Controllers" in chapter 7.

---

## **5.22 Serial Transmit Interrupt (Int 4Ah)**

If the Serial Transmit Data Register Empty Interrupt has been enabled by writing a 0 to bit 7 of the Serial Interrupt Control register (address 4Ch), interrupt 4Ah will be called by the system each time a character is transferred from the Serial transmit data register to the transmit shift register. This indicates that the transmit data register is empty--ready to accept the next character to be transmitted to the Serial interface.

If an application enables this interrupt, the service routine must write a 1 to bit 6 of the Serial interrupt control register to clear and reenable the interrupt.

This interrupt is normally disabled by the system BIOS, since the same function can be achieved by waiting for bit 1 (Transmitter Empty) of the Serial Status register (address 48h) to become a "1" before writing a data byte to the Transmit Data register. Note that the transmitter must be empty before attempting to write to the Data register, or else data could be lost.

5

---

## **5.23 Serial Ring/Carrier Interrupt (Int 4Bh)**

When the Serial RING signal becomes true, indicating a ring condition on the interface, or the RLSD signal becomes false, indicating a loss of carrier, interrupt 4Bh is called by the system. The interrupt service routine must read the Interrupt Status register (address 40h) to determine which interrupt has occurred. If bit 4 of this register is a 1, the interrupt was caused by a ring condition. If bit 3 is a 1, a loss of RLSD caused the interrupt. To clear either of these interrupts, a 1 must be written to the appropriate bit in the Clear Interrupt Request register (address 40h).

---

## **5.24 HP-IL IRQ Interrupt (Int 4Ch)**

The HP-IL IRQ Interrupt is generated as a system interrupt request by the HP-IL Controller. The interrupt must have been enabled using the Int 50h service Modify Interrupt Control Register 42h. Once the interrupt occurs, the interrupt routine should then clear the interrupt using the Clear Interrupt Request Register (see hardware description chapter 7).

A thorough description of HP-IL can be found in the Osborne/McGraw-Hill publication, *The HP-IL System: An Introductory Guide to the Hewlett-Packard Interface Loop*, by Kane, Harper, and Ushijima (1982).

---

## 5.25 Low Battery Interrupt (Int 4Dh)

The Low Battery Interrupt is issued by the BIOS to warn the user of a low battery condition. If the display is currently in alpha mode when the condition is detected, the default handler for this interrupt causes a blinking, inverse-video "Low Battery!" message to be displayed in the lower left-hand corner of the screen. If the current display is graphics, the display will blink a few times.

The hardware indicates a low-battery condition when the battery level drops below about 5.8V (20 percent charge remaining).

---

## 5.26 Modem Input Interrupt (Int 4Eh)

The Modem Input Interrupt is functionally identical to the Serial Input Interrupt 4Fh, except that it is generated for each byte of data that arrives through the modem rather than the serial port. Servicing the interrupt should be done in the same way as for the serial port, except that the I/O addresses used are AAh instead of 4Ah, A8h instead of 48h, and ACh instead of 4Ch.

## 5.27 Serial Input Interrupt (Int 4Fh)

Interrupt 4Fh is generated whenever a character is received by the serial port while interrupts are enabled for the corresponding UART. The received data is available to be read in the Serial Received Data register at I/O address 4Ah.

It is the responsibility of the interrupt service routine to reset the serial port UART so that subsequent characters can be received. The sample routine shown below illustrates how this is done. Note that the service routine must maintain a local copy of the serial port status register (which contains information about word length, stop bits, parity, etc.); this is necessary since the actual status register cannot be read (i.e., it is write-only).

5

```
Sin$INT: Save all registers
    in     al,4Ah          ;Read the serial port data byte
    Process the data
    mov     al,SerConfig   ;Get our local copy of the serial port
    or      al,2            ; port status byte, set the "Clear Status"
    out     48h,al          ; bit, and reset the serial port UART
    in     al,4Ch           ;Get transmit/receive interrupt status,
    or      al,10h          ; set the "Clear Interrupt" bit, and
    and    al,NOT 20h       ; zero the "Disable Interrupt" bit
    out     4Ch,al          ;Make serial port ready for next character
    Restore all registers
    iret
```

## 5.28 System Services Interrupt (Int 50h)

A number of system service functions are provided to allow the applications programmer easy and fast access to a wide variety of unique system features. In most cases, these service functions permit you to easily and safely perform tasks that would otherwise be very difficult, dangerous, or impossible without communicating directly with the hardware. In some cases, they provide you with faster, more efficient alternatives to the usual methods of performing certain functions -- such as displaying a string of characters or flushing the keyboard input buffer.

In any case, the programmer should be aware that use of these services tailors an application to run efficiently on the Portable PLUS, while sacrificing portability to other HP (or anyone else's) computers. In particular, The Portable (HP 110) does not currently support any of these services; a Portable PLUS application that makes use of any system service *will not run on an HP 110*.

All services are invoked by placing the service number in BX, additional parameters in other registers as required, and performing an Int 50h.

Table 5-7 summarizes the routines provided by the system services interrupt. The table which follows that describes these services in detail.

**Table 5-7. System Services Interrupt 50h Functions**

Function	Service
00	Enable Plug-in ROM
01	Call Plug-in ROM
02	Format RAM Disc
03	Memory Initialization
04	Get RAM Disc Limit
05	Get Maximum System Size
06	Display String
07	Display Character
08	Flush Keyqueue
09	Initialize Alpha Display
0A	Simulate Keyboard Input

**Table 5-7. System Services Interrupt 50h Functions (Continued)**

0B	Fast Write to Display
0C	Add New Font
0D	Display CONFIG ROM String
0E	Get CONFIG ROM String
0F	Update On-screen Clock
10	Restart Heartbeat
11	Wait for Interrupt
12	Read Timeout
13	Write Timeout
14	Read Status Limit
15	Write Status Limit
16	Return Card IDs
17	Alter Interrupt Control Register 42h
18	Alter Interrupt Control Register A2h
19	Conditonal Sleep
1A	(Reserved)
1B	PPU Communications
1C	Set Plug-in Card Power Estimate
1D	Read Clock or Alarm
1E	Write Clock or Alarm
1F	Read Time Zone
20	Write Time Zone
21	Read ROM Slot 7 Subdirectory Name
22	HP-IL Sleep
23	HP-IL Wake
24	Datacomm Sleep
25	Datacomm Wake
26	Security Enable
27	Security Disable

The system services are described in detail in table 5-8.

**Table 5-8. System Services Int 50h Detailed Description**

Function	Description
<b>BX=0 (00h)</b>	<p><b>Enable Plug-in ROM</b></p> <p>This routine attempts to enable (or disable) a plug-in ROM. The ROM can be specified by name or number. The ROM number is dependent on slot and type, and in general will only be known by being previously returned by this service.</p> <p><b>Specify:</b></p> <ul style="list-style-type: none"><li>AL&gt;0 Number of plug-in ROM to be enabled (01h-0FDh)<ul style="list-style-type: none"><li>=0 Disable any currently enabled plug-in ROM</li><li>=OFFh Enable plug-in ROM named in specified string</li><li>=OFEh Return current ROM enabled status unchanged</li></ul></li><li>DS:DX = Address of 8-byte string containing the name of the ROM to be enabled (only if AL=OFFh)</li></ul> <p><b>Returns:</b></p> <ul style="list-style-type: none"><li>AH = Number of ROM enabled (0 if no ROM enabled)</li><li>AL = Number of previously enabled ROM (0 if none enabled)</li><li>BX:0 = Starting address of plug-in ROM space (BX=segment no.)</li><li>CY=0 ROM specified exists (or 0 is specified)<ul style="list-style-type: none"><li>=1 The specified ROM does not exist (does not change the status of currently enabled ROM)</li></ul></li></ul> <p><b>Destroys:</b></p> <p>Nothing</p>

**Table 5-8. System Services Int 50h Detailed Description (Continued)**

**BX=1      Call Plug-in ROM  
(01h)**

This routine passes control to a specified address in a specified plug-in ROM and will allow control to be returned to the point at which it was invoked. The service can be invoked from anywhere including another plug-in ROM. The ROM can be identified by name or number as with the enable ROM service except that the specified ROM must be a full bank (a half bank ROM cannot contain ROM executable code). Execution will be passed to the specified paragraph number of the ROM. Note that this requires all ROM entry points to begin on a paragraph boundary, but allows the address to be specified with a single word and frees the calling program from having to know the address at which the ROM is mapped. The invocation will set the segment address to the start of the code with the offset address zero. The invoked code must exit with a far return to restore the environment appropriately.

**Specify:**

AL = Plug-in ROM number (FFh specifies the plug-in ROM named in the string at DS:DX)

DS:DX = Address of 8-byte string containing the name of the desired ROM (only if AL=FFh)

CX = Address of code to invoke (paragraph number relative to the start of the ROM)

**Returns:**

AX=0 Call failed due to erroneous parameter

CY=1 Call failed due to erroneous parameter

*If the call is successful: flag, register, and stack states at time of return are determined by the invoked code. Note that the invoked code should not return both AX=0 and CY=1, as this condition will make a successful call appear to have failed.*

**Destroys:**

BX

**Table 5-8. System Services Int 50h Detailed Description (Continued)**

**BX=2 Format Edisc  
(02h)**

This routine initializes the Edisc (destroying any prior contents). The size of the disc is determined from previously established system variables. The structure of the Edisc is described in chapter 8.

**Specify:**

AX=OBEACh Safety check to avoid inadvertent data loss

**Returns:**

Nothing

**Destroys:**

BX

**BX=3 Memory Initialization  
(03h)**

This service initializes the plug-in cards, checks the Edisc integrity, and sets up the memory parameters for the Edisc and system memory. If the Edisc is corrupt then some memory parameters are reset to maximum Edisc to prevent the accidental destruction of Edisc data. Plug-in RAM cards that contain system memory are enabled; all other RAM cards are disabled.

**Specify:**

Nothing

**Returns:**

AX = Number of paragraphs of total RAM memory in the machine

BX = Number of paragraphs of system memory in the machine

CY=1 Edisc is corrupt

**Destroys:**

Nothing

**Table 5-8. System Services Int 50h Detailed Description (Continued)**

BX=4  
(04h)

**Get Edisc Limit**

This service examines the file allocation table of the Edisc and finds the highest numbered sector that is not available (is part of a file). This service returns the number of sectors of Edisc needed to include all present data without packing. This number will include the space required for the special checksum sectors, and will also include at least one sector for data beyond the root directory (even on an empty disc).

**Specify:**

Nothing

**Returns:**

BX = Number of sectors required by the Edisc to retain the current data

**Destroys:**

Nothing

BX=5  
(05h)

**Get/Set Maximum System Size**

This service allows the system size value (which is maintained in the boot sector) to be set and retrieved. This service maintains the checksum of the boot sector if the value is set. The value is the number of 4K byte units of system memory existing beyond 64K. This service does not check for a valid value when setting the system size (whatever is passed to the service will be set).

**Specify:**

AX = 0 (system size value returned; not set)

AX ≠ 0 (system size value to be set)

AL = system size value: (system size - 64K)/4K

**Returns:**

AX = Value of system size variable

**Destroys:**

BX

**Table 5-8. System Services Int 50h Detailed Description (Continued)**

**BX=6  
(06h)**

**Display String**  
Output an asciz (null terminated) string to the display.  
The string can contain control characters and escape sequences.

**Specify:**  
DS:SI = Address of buffer containing asciz string

**Returns:**

Nothing

**Destroys:**

BX

**BX=7  
(07h)**

**Display Character**  
Output a single character to the display. The character can  
be a control character or part of an escape sequence.

**Specify:**  
AL = Character to be displayed

**Returns:**

Nothing

**Destroys:**

BX

**BX=8  
(08h)**

**Flush Keyqueue**  
Empties out the keyboard type-ahead buffer.

**Specify:**  
Nothing

**Returns:**

Nothing

**Destroys:**

BX

**Table 5-8. System Services Int 50h Detailed Description (Continued)**

BX=9 (09h)	<b>Initialize Alpha Display</b>  Resets the alpha display. HP mode is selected, softkey buffers are initialized, and HP fonts are loaded. Display RAM is erased, the screen is positioned at the first line of display RAM, and the underscore cursor is placed at the upper left corner. <b>Specify:</b> Nothing <b>Returns:</b> Nothing <b>Destroys:</b> BX
BX=10 (0Ah)	<b>Simulate Keyboard Input</b>  Simulates the pressing of a key on the keyboard, either by forcing the keyboard driver to process a specified scancode and modifiers, or by forcing a keycode into the key queue. If the scancode/modifier approach is used, you can additionally specify whether or not a Keyboard CON Expansion interrupt will be generated. <b>To add character to key queue, specify:</b> DH = Any negative value DL = Character to be added to key queue <b>To simulate scancode/modifier combination, specify:</b> DH = Modifier bits (Bits 0/1/2 = Ctrl/Shift/Extend) DL = Scancode (0-71) <b>AH#0 Bypass keyboard CON Expansion processing</b> =0 Pass scancode and modifiers through CON Expansion <i>Scancode/modifier simulation works only in normal keyboard mode, and should not be used in Scancode or Modifier modes.</i> <b>Returns:</b> Nothing <b>Destroys:</b> BX

**Table 5-8. System Services Int 50h Detailed Description (Continued)**

**BX=11 Fast Write to Display  
(0Bh)**

Displays an asciz (null terminated) string. Characters are displayed verbatim; there is no escape processing. The only control characters that are recognized are carriage return and linefeed. Note that there also is no end-of-line wrap; nothing prevents you from writing "beyond" the right margin and into the font tables.

**Specify:**

AH = Attributes to go with each character

DH = Starting row (0-24, screen-relative)

DL = Starting column (0-79)

DS:SI = Buffer containing asciz string to be displayed

**Returns:**

Nothing

**Destroys:**

BX

**BX=12 Add New Font  
(0Ch)**

Replaces a current font table with a new one that has the same ID. The new font table can be packed or unpacked, and can define either 128 or 256 characters; it must, however be the same size or smaller than the table it replaces.

If there currently is no font with the specified ID, an attempt is made to install the new font in any available (currently unused) fontspace.

**Specify:**

DH = Font ID character (e.g., 'A')

DL = Font ID number (e.g., 8)

CH=0 Font table is packed (otherwise, unpacked)

CL=0 Font table defines 128 characters (otherwise, 256)

DS:SI = Address of new font table

**Returns:**

Nothing

**Destroys:**

BX

**Table 5-8. System Services Int 50h Detailed Description (Continued)**

**BX=13      Display Config EPROM String  
(0Dh)**

Displays an asciz (null-terminated) string obtained from the config EPROM. For ease of localization, the EPROM contains many asciz strings along with a table of pointers to the start of each string. By specifying the unique number of the desired string, this service will send that string to the display. The string can contain control characters and escape sequences. Note that the string number is not checked -- an invalid number will display a garbage string.

**Specify:**

DX = String number

**Returns:**

Nothing

**Destroys:**

BX

**BX=12      Get Config EPROM String  
(0Eh)**

Reads an asciz (null-terminated) string obtained from the config EPROM into a specified buffer. The entire string, including the final null, is returned. Note that the string number is not checked -- an invalid number will return a garbage string (which may be VERY VERY long).

**Specify:**

DX = String number

DS:SI = Address of buffer to receive string

**Returns:**

Specified string in buffer at DS:SI.

**Destroys:**

BX

**Table 5-8. System Services Int 50h Detailed Description (Continued)**

**BX=15      Update On-screen Clock  
(0Fh)**

Updates the on-screen clock (the one at the bottom center of the screen, among the softkey labels) to display the current time as read from the PPU.

**Specify:**

Nothing

**Returns:**

Nothing

**Destroys:**

BX

**BX=16      Restart Heartbeat  
(10H)**

Restarts the heartbeat timer at its normal 18-ticks-per-second rate.

**Specify:**

Nothing

**Returns:**

Nothing

**Destroys:**

BX

**BX=17      Wait for Interrupt  
(11H)**

Causes the computer to execute a HLT instruction with the interrupt system left enabled. Use of this service is preferred over simply executing an STI followed by a HLT since it informs the PPU of the halt, thereby maintaining fuel gauge accuracy.

**Specify:**

Nothing

**Returns:**

Nothing

**Destroys:**

BX

**Table 5-8. System Services Int 50h Detailed Description (Continued)**

**BX=18    Read Timeout  
(12h)**

Returns the current system timeout interval. This is the number of seconds that must elapse during which no I/O takes place in order for the system to go to sleep.

Specify:

Nothing

Returns:

DX = Current timeout interval (in seconds)

Destroys:

BX

**BX=19    Write Timeout  
(13h)**

Sets the current system timeout interval. This is the number of seconds that must elapse during which no I/O takes place in order for the system to go to sleep. If you specify an interval of zero, sleep is disabled (the system never times out).

Specify:

DX = New timeout interval (in seconds)

Returns:

Nothing

Destroys:

BX

**BX=20    Read Status Limit  
(14h)**

Returns the current keyboard status-call limit required to cause the system to enter a low-power halt (power-save mode).

Specify:

Nothing

Returns:

DX = Current keyboard status-call limit

Destroys:

BX

**Table 5-8. System Services Int 50h Detailed Description (Continued)**

**BX=21 Write Status Limit  
(15h)**

Sets the keyboard status-call limit required to cause the system to enter a low-power halt (power-save mode). This limit is the number of keyboard status requests which must be made within one second before the system will enter a halt state (extending battery life). If the status limit is zero, both power-save mode and system timeout are disabled. If the status limit is greater than 2000h, only power-save mode is disabled.

**Specify:**

DX = New keyboard status-call limit

**Returns:**

Nothing

**Destroys:**

BX

**BX=22 Return Card IDs  
(16h)**

Returns the card IDs for each logical card so that its enable address can be determined. This service should be used instead of reading directly from the hardware since the value read from a dummy drawer is related to the values on the bus before the read. The only way to ensure that a card is in the system is to use this service and check the registers returned for the desired ID.

**Specify:**

Nothing

**Returns:**

AH = Card 2A (I/O address 0C0h)

AL = Card 2B (I/O address 0D0h)

BH = Card 1A (I/O address 0E0h)

BL = Card 1B (I/O address 0F0h)

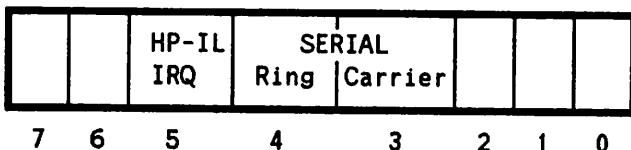
**Destroys:**

Nothing

**Table 5-8. System Services Int 50h Detailed Description (Continued)**

**BX=23 Alter Interrupt Control Register 42h  
(17h)**

Modifies I/O register 0042h to enable or disable selected interrupts without modifying all the interrupts in the register. The BIOS retains a copy of the current setting of this register since it can not be read directly (see hardware description Chapter 7). By using this service an application can modify the ring interrupt control without affecting the status of the carrier interrupt control.



**Specify:**

AL = Pattern of selected bits to change.

AH=0 To disable the selected interrupts

1 To enable the selected interrupts

**Returns:**

Nothing

**Destroys:**

BX

**Table 5-8. System Services Int 50h Detailed Description (Continued)**

**BX=24 Alter Interrupt Control Register A2h  
(18h)**

Modifies I/O register 00A2h to enable or disable selected interrupts without modifying all the interrupts in the register. The BIOS retains a copy of the current setting of this register since it can not be read directly (see hardware description Chapter 7). By using this service an application can modify the ring interrupt control without affecting the status of the keyboard interrupt control.

Keyboard Matrix	PLUG-IN #1	PLUG-IN #2	MODEM Ring	MODEM Carrier	MODIFIER KEYS Extend	Shift	Ctrl
7	6	5	4	3	2	1	0

**Specify:**

AL = Pattern of selected bits to change.

AH=0 To disable the selected interrupts

1 To enable the selected interrupts

**Returns:**

Nothing

**Destroys:**

BX

**BX=25 Conditional Sleep  
(19h)**

Puts the unit into a sleep state to save power if the recharger isn't plugged in or if the battery voltage drops below 5.8V (80% level) (and if a modem or serial carrier signal isn't present). Any interrupt will wake up the unit (key hit, alarm, modem ring, etc.) and return it to the calling application.

**Specify:**

Nothing

**Returns:**

Nothing

**Destroys:**

BX

**Table 5-8. System Services Int 50h Detailed Description (Continued)**

**BX=27 PPU Communications  
(1Bh)**

Communicates with the PPU to perform an assortment of commands. Before exiting, this service will send a byte to the PPU, read a byte from the PPU, or wait until the PPU is not busy.

For the write service, the byte is sent to the PPU once it is ready. For the read service, a byte is sent to the PPU requesting data once it is ready. Then, when the data is available, it is read and returned. The wait service will return once the PPU is ready. The PPU wait service is handy for making sure the PPU has completed a task. For example, the Serial and Modem ON services take a relatively long time to perform and you must make sure these devices are on before sending data to them.

*The system interrupts must be disabled during multi-byte commands.* Talking to the PPU is a very slow process--a one-byte data transfer takes about 2.3 ms to complete. For very long commands, the interrupts will be disabled for a long period of time so the service polls the serial and modem ports for input. This ensures that the system will not drop incoming data during long commands.

**Specify:**

AH=0 Read a single byte from the PPU

- 1 Write a single byte to the PPU
- 2 Wait until the PPU is not busy

AL = Byte to send (when AH=1)

**Returns:**

AH=0 Byte read/written successfully

- 1 Read/write failed (PPU wait timeout)

AL = Byte read from PPU (when AH=0)

**Destroys:**

BX

Table 5-9, which follows this table of System Services, describes the PPU commands.

**Table 5-8. System Services Int 50h Detailed Description (Continued)**

**BX-28 Set Plug-in Power Estimate  
(1Ch)**

Sets the charge levels for a plug-in drawer for use in calculating the battery percentile reading. The plug-in drawers configuration address is used to select one of the two possible drawers. To determine the configuration address, use the Return Plug-in Card IDs service. Cards 1A and 1B are mapped to configuration address 0E0H and Cards 2A and 2B are mapped to 0C0H.

There are two power levels required, each calculated from its current usage (mA) according to the following formula, and then converted to hex:

$$\text{level (decimal)} = 61.084 \times \text{current}$$

**Specify:**

AX = Power usage when system is asleep

CX = Power usage when system is awake

DL = Configuration address (0C0h or 0E0h)

**Returns:**

Nothing

**Destroys:**

BX

**Table 5-8. System Services Int 50h Detailed Description (Continued)**

**BX=29    Read Clock or Alarm**  
(1Dh)    Reads the current Clock or Alarm time information into a 6-byte buffer.

**Specify:**

    AH=0    Read the clock  
    \*0    Read the alarm

    ES:SI = Pointer to 6-byte buffer for return of the following:  
        Bytes 1&2 - (Word) Number of days since 1/1/1980  
        Byte 3 - Minutes (of the current hour, 0-59)  
        Byte 4 - Hour (of the current day, 0-23)  
        Byte 5 - 1/100's of seconds (of current second, 0-99)  
        Byte 6 - Seconds (of the current minute, 0-59)

**Returns:**

    AH=0    Clock/alarm read successfully  
    \*0    Data read is invalid

**Destroys:**

    BX

**BX=30    Write Clock or Alarm**  
(1Eh)    Resets the clock or alarm according to the information supplied by the user in a 6-byte buffer.

**Specify:**

    AH=0    Write to the clock  
    \*0    Write to the alarm

    ES:SI = Pointer to 6-byte buffer containing the following data:  
        Bytes 1&2 - (Word) Number of days since 1/1/1980  
        Byte 3 - Minutes (of the current hour, 0-59)  
        Byte 4 - Hour (of the current day, 0-23)  
        Byte 5 - 1/100's of seconds (of current second, 0-99)  
        Byte 6 - Seconds (of the current minute, 0-59)

**Returns:**

    AH=0    Clock/alarm written successfully  
    \*0    Write failed

**Destroys:**

    BX

**Table 5-8. System Services Int 50h Detailed Description (Continued)**

**BX=31    Read Time Zone  
(1Fh)**

Returns the current Time Zone setting. Only hourly time zones are supported. The zone ranges from -12 (Alaska) to 0 (London) to +12 (USSR). When the clock or alarm is read, the time zone is added to the value.

Once the time has been set to a correct local time, changing the time zone causes the clock to be read in the correct value for that zone. The alarm function is not affected by changing time zones -- the alarm will go off at the initially specified time (an alarm set for 16:00 PST will go off at 15:00 MST).

The parameter is calculated from the time zone (relative to Greenwich Mean Time) as:

$$\text{parameter (decimal)} = 12 + \text{time zone}$$

**Specify:**

Nothing

**Returns:**

AH=0 Valid time zone returned

#0 Invalid time zone read

AL = Parameter (see above)

04h Pacific Standard (-8).

05h Pacific Daylight.

05h Mountain Standard (-7).

06h Mountain Daylight.

06h Central Standard (-6).

07h Central Daylight.

07h Eastern Standard (-5).

08h Eastern Daylight.

0Ch Greenwich Mean (+0)

0Dh Europe (+1)

**Destroys:**

BX

**Table 5-8. System Services Int 50h Detailed Description (Continued)**

**BX=32 Write Time Zone  
(20h)**

Changes the current Time Zone setting. For further information regarding time zones and the parameter required by this function, see *Read Time Zone* above.

**Specify:**

- AL = Parameter (see above)
  - 04h Pacific Standard (-8)
  - 05h Pacific Daylight
  - 05h Mountain Standard (-7)
  - 06h Mountain Daylight
  - 06h Central Standard (-6)
  - 07h Central Daylight
  - 07h Eastern Standard (-5)
  - 08h Eastern Daylight
  - 0Ch Greenwich Mean (+0)
  - 0Dh Europe (+1)

**Returns:**

- AH=0 Time zone changed successfully
- \*0 Time zone not altered

**Destroys:**

BX

**BX=33 Read ROM Slot 7 Subdirectory Name  
(21h)**

Returns the name of the ROM in the special ROM Slot 7 (see chapter 9) to the eight byte buffer. (Any ROM plugged into this slot is treated differently at boot.)

**Specify:**

ES:DI Pointer to 8-byte buffer to receive subdirectory name

**Returns:**

AX=0 ROM not found in slot 7

\*0 ROM found in slot 7

ES:DI = Pointer to 8-byte buffer containing subdirectory name

**Destroys:**

BX

**Table 5-8. System Services Int 50h Detailed Description (Continued)**

**BX=34 HP-IL Sleep  
(22h)**

Turns off the HP-IL interface. The state of the HP-IL controller is saved, and the controller is powered down. *This service should not be called if the HP-IL controller is already asleep.*

**Specify:**

Nothing

**Returns:**

Nothing

**Destroys:**

BX

**BX=35 HP-IL Wake  
(23h)**

Turns on the HP-IL interface. The state of the HP-IL controller is restored to the condition it had before being powered down.

**Specify:**

Nothing

**Returns:**

Nothing

**Destroys:**

BX

**BX=36 Datacom Sleep  
(24h)**

If the serial port is currently on, it is turned off (including the DTR and RTS lines). If the built-in modem is currently on, the contents of the internal registers (plus some additional state information) is saved, and the modem is turned off.

**Specify:**

Nothing

**Returns:**

Nothing

**Destroys:**

BX

**Table 5-8. System Services Int 50h Detailed Description (Continued)**

**BX=37      Datacom Wake  
(25h)**

If the serial port was turned on when the *Datacom Sleep* service was called, it is turned on, restoring the state that it was in before it went to sleep. If the serial port was already off when the sleep service was last called, it remains off. Similarly, if the built-in modem was turned on the last time *Datacom Sleep* was called, it is turned on, restoring all of its internal registers plus the state it was in before it went to sleep. If the modem was already off when the sleep service was last called, it remains off.

**Specify:**

Nothing

**Returns:**

Nothing

**Destroys:**

BX

**BX=38      Security Enable  
(26h)**

Puts the unit into a secured sleep state. The only valid reset of a secured unit is waking from sleep. Any other attempts at rebooting will result in having all the RAM (including the internal Edisc) set to zeros. The service to disable security must be called to disable the security function. *This service should be used very carefully due to the potential for losing data.*

**Specify:**

AX = ABCDh

**Returns:**

Nothing

**Destroys:**

BX

**Table 5-8. System Services Int 50h Detailed Description (Continued)**

**BX=39 Security Disable  
(27h)**

Clears the security function. While the unit is secured, the only valid reset is waking from sleep. Any attempts at rebooting will result in having all the RAM (including the internal Edisc) set to zeros.

**Specify:**

Nothing

**Returns:**

Nothing

**Destroys:**

BX

The PPU commands which can be invoked by using the PPU Communications system service (BX=27) are described in table 5-9.

Table 5-9. PPU Commands

Command	Description
40h Serial On	<p>Turns on the serial port power supply. The DTR and RTS outputs should be initialized before this command is given.</p> <p>Sequence:</p> <ul style="list-style-type: none"><li>Send opcode 40h</li><li>Wait until PPU is not busy before sending data</li></ul>
41h Serial Off	<p>Turns off the serial port power supply. The DTR and RTS and TxD outputs will then float.</p> <p>Sequence:</p> <ul style="list-style-type: none"><li>Send opcode 41h</li></ul>
42h DTR Off	<p>Makes serial DTR output false (low voltage). If the serial port is off, this will have no effect on the interface.</p> <p>Sequence:</p> <ul style="list-style-type: none"><li>Send opcode 42h</li></ul>
43h DTR On	<p>Makes serial DTR output true (high voltage). If the serial port is off, this will have no effect on the interface.</p> <p>Sequence:</p> <ul style="list-style-type: none"><li>Send opcode 43h</li></ul>

**Table 5-9. PPU Commands (Continued)**

**44h RTS Off**

Makes serial RTS output false (low voltage). If the serial port is off, this will have no effect on the interface.

**Sequence:**

Send opcode 44h

**45h RTS On**

Makes serial RTS output true (high voltage). If the serial port is off, this will have no effect on the interface.

**Sequence:**

Send opcode 45h

**47h Modem Reset On**

Makes the modem reset line go active (low voltage). This command does not depend on the state of the MODEMON pin.

**Sequence:**

Send opcode 47h

**48h Modem Reset Off**

Makes the modem reset line go inactive (high voltage). This command does not depend on the state of the MODEMON pin.

**Sequence:**

Send opcode 48h

**4Bh Reset CPU**

Requests a reset. The recommended way to do this is through the Re-Boot interrupt 19h. The PPU pulses the logic reset lines SLP and DSLP active.

**Sequence:**

Send opcode 4Bh

**Table 5-9. PPU Commands (Continued)**

**4Ch      Beep Frequency**

Sets the beeper frequency. The frequency (in hertz) is inversely proportional to the specified number and can be approximated by the formula:

$$\text{number (decimal)} = 17925 / (\text{frequency} - 23.71)$$

The decimal number must be converted to hex before being sent. The highest frequency corresponds to 01h, decreasing through FFh, with the lowest frequency at 00h. The default value is 58h.

Sequence:

- Disable Interrupts
- Send opcode 4Ch
- Send number (in hex)
- Restore Interrupts

**4Dh      Beep Duration**

Sets the beeper duration. For durations greater than 100 ms, the length of beep (in seconds) is approximately half the duration value divided by the frequency (in Hertz). The shortest duration corresponds to 10h, increasing through FFh, with the longest duration at 00h. The default value is 80h periods.

Sequence:

- Disable Interrupts
- Send opcode 4Dh
- Send duration (in hex)
- Restore Interrupts

**Table 5-9. PPU Commands (Continued)**

**4Eh Power Initialize**

Initializes the charge levels used to calculate the battery percentile reading. Charge levels are kept for CPU running and halted, serial interface, modem, and the two plug-in cards (denoted by their configuration addresses C0h and E0h), as well as for the charge supplied by the ac recharger. The levels consist of two bytes (with the most significant (MS) byte sent first). Each level is calculated from its current usage (in mA) as:

$$\text{level (decimal)} = 61.084 \times \text{current}$$

This value must be converted to hex before being sent.

**Sequence:**

Disable Interrupts

Send opcode 4Eh

Send MS byte for CPU running

Send LS byte for CPU running

Send MS byte for CPU halted

Send LS byte for CPU halted

Send MS byte for Sleep mode

Send LS byte for Sleep mode

Send MS byte for Deep Sleep mode

Send LS byte for Deep Sleep mode

Send MS byte for serial interface

Send LS byte for serial interface

Send MS byte for modem

Send LS byte for modem

Send MS byte for recharger

Send LS byte for recharger

Send MS byte for 0C0h plug-in when system is awake

Send LS byte for 0C0h plug-in when system is awake

Send MS byte for 0C0h plug-in when system is asleep

Send LS byte for 0C0h plug-in when system is asleep

Send MS byte for 0E0h plug-in when system is awake

Send LS byte for 0E0h plug-in when system is awake

Send MS byte for 0E0h plug-in when system is asleep

Send LS byte for 0E0h plug-in when system is asleep

Restore Interrupts

**Table 5-9. PPU Commands (Continued)**

**50h      Pulse RCM**

Modem returns to command mode. Pulses modem return-to-command line high for 100 ms and then returns it to the inactive state (low). If the return-to-command line is already high, this command leaves it high for 100 ms longer, then drops it low. This command does not depend on the MODEMON pin state.

**Sequence:**

Send opcode 50h

Wait until PPU is not busy (modem in command mode)

**51h      RCM On**

Makes the modem return-to-command line go to a high voltage. This command does not depend on the MODEMON pin state.

**Sequence:**

Send opcode 51h

**52h      RCM Off**

Makes the modem return-to-command line to go a low voltage. This command does not depend on the MODEMON pin state.

**Sequence:**

Send opcode 52h

**53h      Set Contrast**

Sets the LCD contrast. Value 00h is the darkest, 0Fh is the lightest. The highest four bits are discarded.

**Sequence:**

Disable Interrupts

Send opcode 53h

Send contrast (in hex)

Restore Interrupts

**Table 5-9. PPU Commands (Continued)**

<b>62h</b>	<b>Interrupt Enable</b>
	Enables the PPU to interrupt the CPU for alarm, low battery and shut down. Sequence: Send opcode 62h
<b>63h</b>	<b>Interrupt Disable</b>
	Disables the PPU interrupt of the CPU. Internal PPU interrupts are queued and will interrupt the CPU once PPU interrupts are re-enabled. Sequence: Send opcode 63h
<b>64h</b>	<b>Version</b>
	Returns the PPU version number, an ASCII character. Sequence: Disable Interrupts Send opcode 64h Read version Restore Interrupts
<b>65h</b>	<b>CPU Running</b>
	Tells the PPU that the CPU is running (not halted). This data is used in the battery charge level calculations. Sequence: Send opcode 65h
<b>66h</b>	<b>CPU Halted</b>
	Tells the PPU that the CPU is halted. This data is used in the battery charge level calculations. Sequence: Send opcode 66h

**Table 5-9. PPU Commands (Continued)**

**6Ch      Read Fuel Level**

Returns the battery charge level, a one-byte quantity with FFh denoting a full charge and 00h denoting no charge. This value is calculated -- it is not a measured value.

**Sequence:**

- Disable Interrupts
- Send opcode 6Ch
- Read fuel level
- Restore Interrupts

**6Dh      Interrupt Status**

Returns the PPU internal interrupt status. This command should only be sent during interrupt hardware polling so that no interrupts are dropped. The PPU prioritizes its interrupts and indicates only the highest priority one. The highest priority interrupt is then cleared. If no interrupts are pending, a value of zero is returned. The status byte indicates the cause of the interrupt:

(highest priority)	04h: Shut down
	02h: Low Battery
	20h: Alarm
	00h: No interrupts pending

**Sequence:**

- Disable Interrupts
- Send opcode 6Dh
- Read interrupt status
- Restore Interrupts

**Table 5-9. PPU Commands (Continued)**

**76h Set Accuracy**

Sets the real-time clock accuracy adjust. The first byte is an offset: FFh causes the clock to run slower, 00h causes it to run at normal speed; 01h causes it to run faster. The next two bytes specify a counter (in hex) of the number of seconds between adjustments. The clock is adjusted by 0.0004 second each time the counter cycles. The counter value can be calculated as 2103.8 divided by the adjustment in minutes per year, or 345.8 divided by the adjustment in seconds per day.

**Sequence:**

- Disable Interrupts
- Send opcode 76h
- Send offset
- Send LS counter (in hex)
- Send MS counter (in hex)
- Restore Interrupts

**77h Read Accuracy**

Returns the PPU accuracy adjust value. (Refer to the previous command for a description of the parameter values.)

**Sequence:**

- Disable Interrupts
- Send opcode 77h
- Read offset
- Read LS counter
- Read MS counter
- Restore Interrupts

**7Ah Modem Off**

Puts the modem RCM line to a low voltage. Sets modem reset line active (low voltage). Then turns off modem power supplies.

**Sequence:**

- Send opcode 7Ah

**Table 5-9. PPU Commands (Continued)**

**7Ch Modem On**

Sets modem RCM line to a low voltage. Turns on the modem power supplies and auto-sequences its reset lines. There is a built-in delay that allows the modem power supplies to stabilize and its reset sequence to complete. After the delay, the modem is ready for communication with the mainframe.

**Sequence:**

Send opcode 7Ch

Wait until PPU is not busy (modem then ready)

**7Dh Beep**

Causes the beeper to beep at the current frequency and duration. (Refer to commands 4Ch and 4Dh).

**Sequence:**

Send opcode 7Dh

**Table 5-9. PPU Commands (Continued)**

**7Eh      Read Status**

Returns the PPU internal status byte. The parameter error bit (bit 6) is cleared when the PPU is reset; it is set if an illegal value is passed when setting the clock or alarm, the time zone, or plug-in address in the Set IO Drawer Power Service; it remains set until the PPU is reset. Each bit is mapped accordingly:

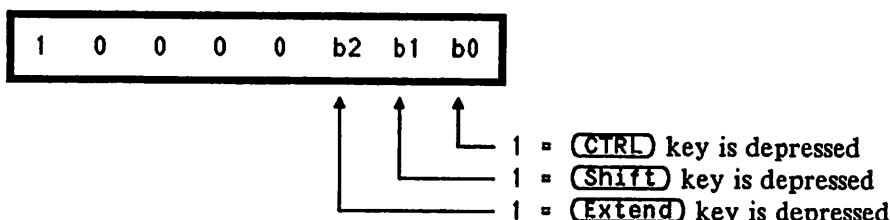
- Bit 7: Interrupts Enabled
- Bit 6: Parameter Error
- Bit 5: CPU ("1"=running "0"=halted)
- Bit 4: Alarm Enabled
- Bit 3: Timeout Disabled
- Bit 2: Low Battery
- Bit 1: Shut Down Active
- Bit 0: Always 0

**Sequence:**

- Disable Interrupts
- Send opcode 7Eh
- Read status byte
- Restore Interrupts

## 5.29 Modifier Key Interrupt (Int 52h)

When the keyboard is in Scancode, Modifier, or Numeric Keypad mode, each up- and down-transition of any modifier key (**CTRL**, **Shift**, and **Extend**) generates a Modifier Key interrupt 52h. In Scancode and Modifier modes, the default handler for this interrupt responds to each transition by adding the resultant state of the modifier keys, plus 80h, to the key queue. The state is encoded in the low three bits of the character, resulting in a byte of the following form:



5

Note that although the interrupt is generated by the transition of just one modifier key, the byte that is added to the queue reflects the final states of all three modifier keys. This means, for example, that if **CTRL** and **Shift** are already down when the **Extend** key is depressed, an 87h will be generated; if the **Shift** key is then released, an 85h will be generated.

The system will call the Modifier Key Interrupt routine if the keyboard is in normal mode (neither Scancode nor Modifier mode is active) only if the numeric keypad is enabled. In this situation, the handler normally does nothing (it simply returns) except under the following conditions:

- The system is in Alt mode.
- An upward transition by the **Extend** key caused the interrupt.
- The last character typed (while the **Extend** key was depressed) was a digit on the numeric keypad.

If all of these conditions are true, the handler adds the numeric-keypad-generated character to the keyqueue.

If you write your own *Modifier Key Interrupt handler*, the Modifier Key interrupt is the result of a hardware interrupt; no special information is passed into the interrupt handler. All registers used by your handler should be saved upon entry and restored at exit, and the routine should end with an IRET instruction. Just prior to restoring registers, you must clear the modifier key interrupt by writing a 07h to I/O address 0A0h.

---

### 5.30 Print Key Interrupt (Int 53h)

The Print Key Interrupt is issued whenever, in normal keyboard mode, the **(Print)** key is pressed. The default handler for this interrupt causes the current contents of the display to be dumped to the PAM Printer Interface device.

This interrupt is intended to be a hook by which an application can trap the **(Print)** key.

---

### 5.31 HP-IL Primitives Interrupt (Int 54h)

Both the Portable and Portable PLUS use the Hewlett-Packard Interface Loop (HP-IL) to communicate with discs, printers and various other I/O devices. HP-IL can also be used to control HP-IB (IEEE-488) devices through a HP82169 HP-IL/HP-IB interface. Each device driver communicates with its device by calling HP-IL primitive routines.

Instruments and other devices not currently supported by MS-DOS device drivers may also be controlled with a suitable application program. The application programmer can call these same HP-IL primitives to communicate with the device to be controlled. Using these routines will simplify the program and allow the application to share the bus with the MS-DOS drivers without conflict. It is recommended that devices which have MS-DOS drivers be controlled through standard MS-DOS system calls and that HP-IL primitives be used only for unsupported devices.

HP-IL Primitives Interrupt 54h is used to invoke the various primitive functions, which in turn provide low level control of the HP-IL interface and serve to isolate from each other the various routines that use HP-IL. For example, an application can read data from an HP-IL voltmeter, use MS-DOS Int 21h functions to print the data, and save the data on an HP-IL disc without any conflicts.

By using Int 54h, the application programmer can do the following operations:

- Configure the loop and assign addresses to all the devices on it.
- Address any device to either send data or receive it.
- Send data bytes to any device on the loop.
- Receive data bytes from any device on the loop.
- Send or receive individual frames.
- Search for devices according to their accessory IDs.
- Set the expected time out period for each operation.
- Read interface status information.
- Read the Accessory ID of any device on the loop.
- Search the loop for a device with a certain accessory ID.

5

Many instruments can be controlled by simply sending data bytes and receiving information back. For example, to instruct an HP 3421A Data Acquisition unit to take a voltage reading and report the results requires the following steps:

1. Configure the loop to put it into a known state.
2. Locate the HP 3421A and determine its address.
3. Address the Portable PLUS to talk and the HP 3421A to listen.
4. Set the time out to an appropriate value for the HP 3421A.
5. Send the data bytes "DCV<cr>".
6. Address the HP 3421A to talk and the Portable PLUS to listen.
7. Receive the data bytes that the HP-3421A returns.

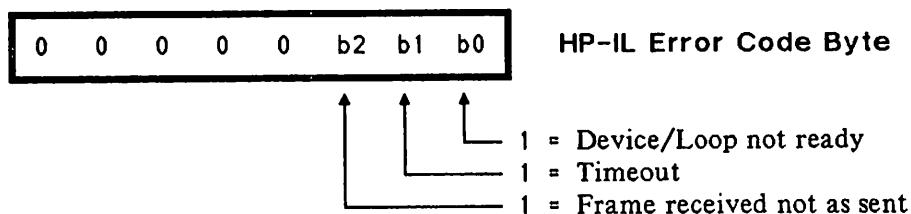
Other instruments may require the application to directly send HP-IL loop commands and be able to respond to service requests. An application can directly send and receive

HP-IL frames and so can generate any command sequence that an instrument may require. A thorough description of HP-IL can be found in the Osborne/McGraw-Hill publication, *The HP-IL System: An Introductory Guide to the Hewlett-Packard Interface Loop*, by Kane, Harper, and Ushijima (1982).

The HP-IL primitives provide the low level control over the HP-IL interface in the Portable PLUS. They permit an application to interleave I/O operations and allow optimizations that lead to more efficient operation. The routines keep track of loop information such as the time of the last frame transmission and the current state of the loop. If a command is given that would be unnecessary due to the current state of the loop then it will be safely ignored. For example if an application issues an Address command for devices that are currently addressed then no frames will be sent.

Before each major group of operations (or any time the caller cannot be sure of the loop configuration) the Config function must be called. No frames will be sent during this call if it has been only a short time since the most recent loop operation. Note that because this call checks the time since the previous loop operation, any HP-IL operations performed immediately before the call will cause the Config call to be ignored.

Each HP-IL function returns a completion code in AL and error status in the CY (Carry) flag. The remaining flags and registers are not changed. If the function fails due to a loop problem (ie. not connected or one or more devices turned off) then it will set the carry flag and return an error code. The following bits will be set for errors:



In most cases an error requires that the application use the Config command to restore the loop to a known state. Applications that issue Send Frame and Get Frame commands may have to process the error according to which command was sent over the loop. For example if the application sends an autoaddress command frame it will be modified by any devices in the loop and cause the HP-IL interface to issue a "Frame Received Not as Sent" error. In this case, the error is expected and can be ignored by the application. However, if a Config command fails, user intervention may be needed to restore the loop.

If a Timeout is indicated, the next call will generate a power up sequence (multiple IFCs followed by a single RFC). This sequence is also performed before the first function call after power on. This standard timeout recovery may not be overridden.

After configuring the loop the application must determine the address of the device that it is controlling. If it is a HP-IB device that is connected via a HP 82169A interface then the address will be the same as the HP-IB address. If the device is an HP-IL device then it will be autoaddressed according to its position on the loop. Note that the user should be careful when choosing HP-IB device addresses since it is possible to have an HP-IB and an HP-IL device that both respond to the same address. *Only addresses 0 through 7 are allowed for HP-IB devices. HP-IL addresses will then start at 8 and be sequentially assigned to all devices around the loop.*

For HP-IL devices the best way for an application to determine the address of a device is with Accessory IDs. If the device supports this feature then the application can issue a Find command that will return the address of the desired device. An application that uses the Find function will not be dependent on the order that the devices are placed on the loop. An alternative way to locate devices is to request a device to talk and use the Input Data Block command to enter a device ID. This is normally an ASCII string that identifies the particular device.

5  
Most of these routines require the Portable PLUS to be the active controller on the loop, although some applications (such as HPLINK) use the computer as a non-controller. Any application that passes control of the loop to another device *must* regain control before making any MS-DOS calls that could require use of the loop.

A typical character device driver will call the HP-IL functions in the following order for each character to be sent to the target device:

Config	<i>;Configure the loop if necessary</i>
Find	<i>;Find desired device</i>
Address	<i>;Address device to listen</i>

*;No frames are sent above under normal conditions*

SetTimeout	<i>;Set to reasonable value for device</i>
SendFrame	<i>;Send byte ("Output" also valid here)</i>
.	
.	
.	

A typical block device driver will perform HP-IL operations in the following order:

```
Config      ;Check loop configuration
Find       ;Find disk drive
SetTimeout
SendFrame   ;Use "SendFrame" for specific frame sequence
.
.
```

Table 5-10 describes the HP-IL primitives interrupt functions.

**Table 5-10. HP-IL Primitives Interrupt 54h Functions**

Function	Description
AH=0 (00h)	<p><b>Configure Loop</b></p> <p>The Configure Loop primitive checks the elapsed time since the last HP-IL operation and performs loop configuration if necessary. This function should be called before each major group of loop operations to ensure correct configuration. No frames are sent unless absolutely necessary. Configuring the loop consists of giving each device an address according to its position on the loop. The first device on the loop is given address 8, the second device is address 9, and so on up through a maximum of 23 devices. (Addresses 0-7 are reserved for HP-IB devices that may be connected to the loop through a HP 82169A HP-IL/HP-IB Interface.) If 23 or more devices exist on the loop, all devices beyond the first 22 are assigned address 30 (decimal).</p> <p>Returns: AL = Completion code.</p>

**Table 5-10. HP-IL Primitives Interrupt 54h Functions (Continued)**

AH=1  
(01h)

**Find Device**

This primitive searches the loop starting at the address specified in BH, looking for a device with an accessory ID that matches the one in BL. If the value of BL is *x*Fh, only a class match is performed (only the top four bits are compared with the device accessory IDs). If a matching device is found, its address is returned in BL. The BIOS maintains a table of all devices that have been found on the loop and will return data from this table if it is available. No frames are sent unless absolutely necessary.

**Specify:**

BH = Starting address (00h-1Eh).

BL = Desired accessory ID (00h-FFh, excluding FEh).

**Returns:**

AL = Completion code.

BL = Address of device (1Fh if not found).

**Destroys:**

BH

**Table 5-10. HP-IL Primitives Interrupt 54h Functions (Continued)**

AH=2  
(02h)

**Get Accessory ID**

This primitive returns the accessory ID of the HP-IL device at the address specified in BL. Several conditions exist that will cause a value of FEh to be returned:

- 1) Addressed device does not support accessory ID.
- 2) Addressed device does not exist.
- 3) Addressed device has an accessory ID of FEh. FEh is the ID for an Extended class, General device. A program that needs to control a device with this ID will require some other means to determine if the device is on the loop.

No frames will be sent unless absolutely necessary.

**Specify:**

BL = Device address.

**Returns:**

AL = Completion code.

BL = Accessory ID. (FEh = no device at address, device doesn't support accessory ID, or accessory ID is FEh.)

**Destroys:**

BH

**Table 5-10. HP-IL Primitives Interrupt 54h Functions (Continued)**

AH=3 (03h)	Address
	<p>This function prepares the loop for a data transfer between the computer and a device on the loop. It is used before an <b>Input Data Block</b> or <b>Output Data Block</b> primitive to select the device that will either supply or receive data. The address of the selected device is sent as either a talk or listen address while 1Fh designates the address for the Portable PLUS itself. The BIOS keeps track of which devices are currently addressed as talker and listener and will only send an address command if they are changed. Setting both talk address and listen address to 1Fh is a special case that sends <b>UNTalk</b> and <b>UNListen</b> commands to all other devices on the loop.</p> <p>These HP-IL primitives do not support direct data transfer between two other devices on the loop. All transfers must go through the Portable PLUS. If direct transfer is required then the application programmer must use the <b>Send Frame</b> primitive to address the loop and start the transmission of data.</p> <p><b>Specify:</b></p> <ul style="list-style-type: none"><li><b>BH</b> = Talker address (00h-1Eh or 1Fh).</li><li><b>BL</b> = Listener address (00h-1Eh or 1Fh).</li></ul> <p><b>Returns:</b></p> <ul style="list-style-type: none"><li><b>AL</b> = Completion code.</li><li><b>BH</b> = Old talker address.</li><li><b>BL</b> = Old listener address.</li></ul> <p><b>Destroys:</b></p> <ul style="list-style-type: none"><li><b>DX</b></li></ul>

Table 5-10. HP-IL Primitives Interrupt 54h Functions (Continued)

AH=4  
(04h)

### Output Data Block

The **Output Data Block** primitive causes the block of CX bytes of data at location ES:DI to be sent over the loop. If the End Option is set to 1, the last byte of data is sent as an END frame. If DI+CX is greater than 65536, a segment wraparound will occur and incorrect data may be sent.

Specify:

CX = Byte count (0 - 65,536).

DX = End option (0 - 1).

ES:DI = Address of buffer containing data to be sent.

Returns:

AL = Completion code.

CX = Number of bytes actually transferred.

**Table 5-10. HP-IL Primitives Interrupt 54h Functions (Continued)**

AH=5  
(05h)

**Input Data Block**

This primitive reads CX bytes of data from the loop into a buffer at location ES:DI. If BX is a valid SOT frame, it is sent out to initiate the transfer. If BX is 00h or not an SOT frame, the transfer is assumed to have already been started. (Valid SOT frames include SDA, SST, SDI and SAI(1)). The transfer will terminate when all CX bytes have been received or when an ETO is received. If the buffer fills before ETO is received, an NRD sequence is transmitted. There is an ambiguity that occurs when the caller tries to input *exactly* 65536 bytes; this routine will return CX=0 for both the case of a successful operation and the case where no bytes are received.

SDA 560h	Send Data
SST 561h	Send Status
SDI 562h	Send Device ID
SAI 563h	Send Accessory ID

**Specify:**

BX = Optional SOT frame to be sent ("0" = none).  
CX = Count of bytes to accept before NRD.  
ES:DI = Pointer to data buffer.

**Returns:**

AL = Completion code.  
CX = Count of bytes received.

**Table 5-10. HP-IL Primitives Interrupt 54h Functions (Continued)**

AH=6 (06h)	<b>Send Frame</b>  The specified frame is sent out over the loop and an error code is returned. Frame is the frame value (0-2047) to be sent. The values and their usage are defined in the HP-IL interface specification manual, which should be consulted if the programmer needs this level of interface control. The options supported are "Wait For Loop Ready Before Transmit" (DX=0) and "No Wait Before Transmit" (DX=1).  Specify: BX = Frame to be sent (00h-7FFh). DX = Wait option (0=wait; 1=no wait). Returns: AL = Completion code.
AH=7 (07h)	<b>Get Frame</b>  This primitive waits for a frame to be received, and returns it in BX. If no frame is available from the HP-IL interface then 00h is returned. Get Frame is normally used in conjunction with the Send Frame primitive when the programmer must have complete control of the Loop. Returns: AL = Completion code. BX = Received frame (DAB0h=Frame unavailable).

**Table 5-10. HP-IL Primitives Interrupt 54h Functions (Continued)**

AH=8 (08h)	<b>Status</b>  Returns current loop status in BX. If the loop is ready for a frame to be sent, bit 0 will be set. If a frame is available, then bit 1 will be set. Note that any operation that sends a frame will effectively erase the frame available bit. The timeout is also tested if the loop is not ready for a frame, and a timeout error is returned if necessary.  Returns: AL = Completion code. BX = Status: Bit 15-7: (Not used.) Bit 6: Controller active. Bit 5: Talker active. Bit 4: Listener active. Bit 3: Service request received. Bit 2: (Not used.) Bit 1: Frame available. Bit 0: Loop ready for frame.
AH=10 (0Ah)	<b>Set Timeout</b>  An HP-IL device that sends data can perform a loop test on each byte sent by comparing the byte it sends with the byte that returns after traveling around the loop. By specifying a timeout interval, an application can dictate how long the HP-IL driver should wait for a response to each byte sent out over the loop. If a device response is not detected by the time the timeout period has elapsed, an error is declared and control is returned to the calling application.  Variable timeout periods allow the programmer to handle devices that require a long period of time to complete their operation. For example a printer may take 20 seconds to do a formfeed and hold up the loop during that period of time.  Specify: BX = New timeout (in 1/16 seconds). Returns: AL = Completion code. BX = Old timeout (in 1/16 seconds).

**Example:** This program demonstrates using the BIOS HP-IL interrupt (54h) to talk to a device on HP-IL. In this case, the device is an HP 3421A Data Acquisition/Control Unit. The program:

1. Configures the loop and sets the loop timeout value.
2. Finds the HP 3421's loop address.
3. Addresses the HP 3421 to listen and sends the "Read Voltage" command to it.
4. Addresses the HP 3421 to talk and enters the voltage reading.
5. Displays the voltage read.
6. Exits.

The program is structured to be run through EXE2BIN (converted to a .COM program) and therefore doesn't set up DS, ES, or SS, since MS-DOS does that before passing control to the program.

```
page 60,132
title INSTRUMENT CONTROL --- using the BIOS HP-IL interrupt
;
cseg    segment para public 'code'
        assume  cs:cseg, ds:cseg
        org     100h
start   proc   far
        call    config           ; configure the loop
        call    find              ; find the HP 3421
        call    output             ; send the READ VOLTAGE command
        call    enter              ; enter the voltage
        mov     dx,offset buffer  ; address of string
        mov     ah,9                ; display it
        int    21h
        mov     ax,4C00h            ; terminate our program
        int    21h
start   endp
page
;
; CONFIG --- configures the loop into a known state and assigns addresses
; terminates with ERRORLEVEL=1 if timeout
;
```

```

config:
    mov     ah,0                      ; configure the loop
    int     54h                       ; BIOS HP-IL call
    test    al,7                      ; not ready?
    jnz    error1                    ; jif yes
    mov     ah,10                     ; set timeout
    mov     bx,64                      ;   for 4 seconds (64*(1/16))=4
    int     54h
    test    al,7                      ; not ready?
    jnz    error1                    ; jif yes
    ret                            ; else keep going
error1:
    mov     dx,offset loopfail      ; address of error message
    mov     al,1                      ; errorlevel=1
error:
    push   ax                        ; save errorlevel
    mov     ah,9                      ; DOS function to display message
    int     21h
    pop    ax                        ; recover errorlevel code
    mov     ah,4ch                    ; terminate code
    int     21h                      ; terminate
    page

;
; FIND --- finds a device on the loop by its accessory ID
;           terminates with ERRORLEVEL=2 if not found
;

find:
    mov     ah,1                      ; find function
    mov     bh,0                      ; starting search address
    mov     bl,53h                    ; accessory ID for HP 3421
    int     54h                      ; BIOS HP-IL call
    test    al,7                      ; not ready?
    jnz    error2                    ; jif yes
    cmp     bl,1fh                    ; not found?
    jz     error2                    ; jif yes
    mov     address,bl                ; else save the address
    ret

error2:
    mov     dx,offset nodevice      ; address of error message
    mov     al,2                      ; errorlevel
    jmp    error
    page

```

```

;
; OUTPUT --- sends the READ VOLTAGE command to the HP 3421.
;           terminates with ERRORLEVEL=3 if fails.
;
output:
    mov     ah,3          ; address the loop
    mov     bl,address    ; get HP 3421 address (LISTENER)
    mov     bh,1fh        ; get PORTABLE address (TALKER)
    int     54h          ; BIOS HP-IL call
    test   al,7          ; any problems?
    jnz    error3        ; jif yes
    mov     ah,4          ; send data block
    mov     dx,1          ; finish with END frame
    mov     cx,5          ; length of command
    mov     di,offset command ; address of the command
    int     54h          ; BIOS HP-IL call
    test   al,7          ; any problems?
    jnz    error3        ; jif yes
    cmp     cx,5          ; all bytes transferred?
    jnz    error3        ; jif no
    ret
error3:
    mov     dx,offset sendfail ; address of error message
    mov     al,3          ; set errorlevel
    jmp     error
page
;
; ENTER --- read the voltage back from the HP 3421
;           terminate with errorlevel=4 if fails.
;
enter:
    mov     ah,3          ; do loop addressing
    mov     bh,address    ; address of TALKER (HP 3421)
    mov     bl,1fh        ; address of LISTENER (The PORTABLE)
    int     54h          ; BIOS HP-IL call
    test   al,7          ; any problems?
    jnz    error4        ; jif yes
    mov     cx,32         ; maximum number of bytes to read
    mov     bx,560h        ; SDA (Send DAta) command
    mov     di,offset buffer ; address of scratch buffer
    mov     ah,5          ; input data block
    int     54h

```

```

        test    al,7           ; any problems?
        jnz     error4         ; jif yes
        mov     di,offset buffer ; address of start of data
        add     di,cx          ; move to end of data
        mov     ax,0a0dh        ; CR/LF
        stosw
        mov     al,36           ; ascii for $
        stosb
        ret
error4:
        mov     dx,offset entrfail ; address of error message
        mov     al,4             ; errorlevel
        jmp     error
page
;
;
loopfail      db      'Loop failure',13,10,'$'
nodevice       db      'Device not found',13,10,'$'
sendfail       db      'Send failure',13,10,'$'
entrfail       db      'Enter failure',13,10,'$'
command        db      'DCV',13,10
;
;
address db    ?           ; address of HP 3421
buffer db     32 dup (?) ; scratch buffer
;
cseg ends
end   start

```

## 5.32 Sleep Interrupt (Int 55h)

This service is used to force the computer into a recoverable sleep state. Any subsequent interrupt that occurs after the computer goes to sleep (keyhit, serial ring, alarm, etc.) will wake the unit and return it to its state prior to the sleep. *The computer does not reboot.* Before going to sleep, the states of the HP-IL hardware, modem, LCD controller and the current stack are saved so that they may be restored upon waking.

## 5.33 Menu Key Interrupt (Int 56h)

A Menu Key Interrupt is generated whenever, in HP mode, you press the **(Menu)** key. The default handler for this interrupt performs one of three functions:

- If the keyboard is in Modifier mode, adds an 8Ch to the key queue.
- If the softkey labels are turned on, turns them off.
- If the softkey labels are turned off, turns them on.

An application can use the Menu key interrupt in two ways:

- By taking over Int 56h and pointing it at your own Menu key interrupt handler, you can perform your own Menu key processing. When the system branches into your new interrupt handler, the state of the three modifier keys (at the time the **(System)** key was pressed) are available in AH; AL will contain FCh, the Configuration EPROM keymap Local Function code that caused the interrupt to be issued. This interrupt is invoked by the keyboard driver responding to a keyboard hardware interrupt. All general registers are available when the interrupt branches into your handler; they need not be saved and restored (in general, however, you should always save and restore any registers that you will use in servicing an interrupt).
- If you leave the Int 56h vector pointing at the default handler, you can programmatically simulate the **(Menu)** key by issuing an Int 56h software interrupt. No registers are altered by the default handler.

In Alt mode, the **(Menu)** key represents function key **(F9)**; the Menu key interrupt is never generated. Two-byte codes will be added to the key queue according to the following table (E=Extend, S=Shift, C=Control):

---	--C	-S-	-SC	E--	E-C	ES-	ESC
00 43	00 66	00 5C	00 66	00 70	00 70	00 70	00 70

The Configuration EPROM keymap entry that generates a Menu key interrupt is Local Function FCh. Behavior in the various keyboard modes is summarized as follows:

**Normal Mode:** In HP mode, pressing **(Menu)** generates a Menu key interrupt (56h); the default handler toggles the softkey labels on or off. In Alt mode, an appropriate two-byte code is added to the key queue.

**Scancode Mode:** Pressing the **(Menu)** key adds its scancode, 17 decimal (11h) to the key queue. The Menu key interrupt is not generated.

**Modifier Mode:** In HP mode, pressing the **(Menu)** key generates a Menu key interrupt (56h); the default handler adds an 8Ch to the key queue. In Alt mode, an appropriate two-byte code is added to the key queue.

---

## 5.34 System Key Interrupt (Int 57h)

A System Key Interrupt is generated whenever, in HP mode, you press the **User/System** key. If the keyboard is in Modifier Mode, the default interrupt handler adds an 8Bh to the key queue; otherwise, it does nothing. An application can use the System key interrupt in two ways:

- By taking over Int 57h and vectoring it to your own System key interrupt handler, you can perform your own System key processing. When the system branches into your new interrupt handler, the state of the three modifier keys (at the time the System key was pressed) are available in AH; AL will contain FBh, the Configuration EPROM keymap Local Function code that caused the interrupt to be issued. This interrupt is invoked by the keyboard driver responding to a keyboard hardware interrupt. All general registers are available when the interrupt branches into your handler; they need not be saved and restored (in general, however, you should always save and restore any registers that you will use in servicing an interrupt).
- If you leave the Int 57h vector pointing at the default handler, you can programmatically simulate the System key by issuing an Int 57h software interrupt. No registers are altered by the default handler.

In Alt mode, the System key represents function key (f10); the System key interrupt is never generated. Two-byte codes will be added to the key queue according to the following table (E=Extend, S=Shift, C=Control):

---	--C	-S-	-SC	E--	E-C	ES-	ESC
00 44	00 67	00 5D	00 67	00 71	00 71	00 71	00 71

The Configuration EPROM keymap entry that generates a System key interrupt is Local Function FBh. Behavior in the various keyboard modes is summarized as follows:

**Normal Mode:** In HP mode, pressing User/System generates a System key interrupt (57h); the default handler simply returns. In Alt mode, an appropriate two-byte code is added to the key queue.

**Scancode Mode:** Pressing the User/System key adds its scancode, 16 decimal (10h) to the key queue. The System key interrupt is not generated.

**Modifier Mode:** In HP mode, pressing the User/System key generates a System key interrupt (57h); the default handler adds an 8Bh to the key queue. In Alt mode, an appropriate two-byte code is added to the key queue.

## 5.35 Break Key Interrupt (Int 58h)

A Break Key Interrupt is generated by the keyboard driver when you hold down the Shift key and press Break. The default interrupt handler responds by first flushing the key queue, and then putting a ^C (03h) in it. An application can use the Break key interrupt in two ways:

- By taking over Int 58h and vectoring it to your own Break key interrupt handler, you can perform your own Break key processing. This interrupt is invoked by the keyboard driver responding to a keyboard hardware interrupt. All general registers are available when the interrupt branches into your handler; they need not be saved and restored (in general, however, you should always save and restore any registers that you will use in servicing an interrupt).
- If you leave the Int 58h vector pointing at the default handler, you can programmatically simulate Shift Break by issuing an Int 58h software interrupt. No registers are altered by the default handler.

**Shift Break** functions identically in both HP and Alt modes. The Configuration EPROM keymap entry that generates a Break key interrupt is Local Function 09h. Behavior in the various keyboard modes is summarized as follows:

**Normal Mode:** Pressing **(Shift) (Break)** generates a Break key interrupt (58h), which normally flushes the key queue and then puts a ^C (03h) in it.

**Scancode Mode:** Pressing the **(Shift)** key causes the resultant state of all three modifier keys, plus 80h, to be added to the key queue. Pressing the **(Break)** key adds its scancode, 71 decimal (47h) to the key queue. The Break key interrupt is not generated.

**Modifier Mode:** Pressing the **(Shift)** key causes the resultant state of all three modifier keys, plus 80h, to be added to the key queue. Subsequently pressing the **(Break)** key adds an 8Dh to the key queue. The Break key interrupt is not generated.

## 5.36 Enable/Disable Ring Interrupt (Int 59h)

The Serial and Modem ring interrupts can be controlled by interrupt 59h. When this service is called, AL specifies the action to be taken. If AL contains an odd value, both the Modem and Serial Interface ring interrupts are enabled. When they are enabled, the system will call interrupts 42h and 4Bh whenever the RING signal goes from a low to a high state.

If AL contains an even value, the ring interrupts are disabled. Interrupts 42h and 4Bh will then never occur, although the state of the RING line can still be read on a polled basis from the Control Status register at I/O address 42h (for the serial port) or A2h (for the modem).

## 5.37 AUX Expansion Interrupt (Int 5Dh)

The AUX driver expansion interrupt (5Dh) provides a means of examining, trapping, and altering characters coming from the serial port or modem. The driver calls this interrupt whenever a character has been read from the I/O port, but before the character is put into the input queue. A typical application of the AUX Expansion Interrupt might be to map incoming EBCDIC characters into ASCII.

The driver automatically performs any hardware handshaking required by the serial port before interrupt 5Dh is called, so an application doesn't have to observe handshake signals. At exit, if no character is to be placed in the input queue (indicated by returning AX=-1), the driver ignores the normal XON/XOFF software protocol. The application must perform this handshake if it's required.



### Note

An application should always return to the driver by using IRET. This interrupt is called while processing the hardware interrupt that occurs when a character is received. There are hardware-specific protocols that must be observed in order to clear the interrupt.

An application must *not* alter any register except AX.



### Note

If you plan to use Int 5Dh, you should turn the modem on *before taking over the expansion interrupt*. If the modem is turned on *after* taking over the interrupt, you must allow the first 12 characters to be passed through and placed into the BIOS input queue. This is necessary because the BIOS sends a string of 12 characters to power up and configure the modem, and expects all 12 to echo back if the modem is present in the system. If fewer than 12 characters return from the modem to the BIOS, the system will assume the modem is not there.

Table 5-11 describes the AUX expansion interrupt function.

**Table 5-11. AUX Expansion Interrupt 5Dh Function**

Function	Description																																								
<b>AH=0 (00h)</b>	<p><b>Character Received By Serial Port</b></p> <p>The character in AL has just been received through the serial port and is about to be processed by the AUX driver. The expansion code can inspect or alter the character before it is added to the input queue, or discard the character by returning FFh in AX.</p> <p>Register contents:</p> <ul style="list-style-type: none"> <li>AL = Character received from serial port.</li> <li>BL = Serial port status byte:</li> </ul> <table style="margin-top: 10px; border-collapse: collapse;"> <tr> <td>b7</td> <td>--</td> <td>b5</td> <td>b4</td> <td>b3</td> <td>b2</td> <td>b1</td> <td>b0</td> </tr> <tr> <td>↑</td> <td></td> <td>↑</td> <td>↑</td> <td>↑</td> <td>↑</td> <td>↑</td> <td>↑</td> </tr> <tr> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> </table> <ul style="list-style-type: none"> <li>— Break being sent</li> <li>— Xmit data register empty</li> <li>— Break being received</li> <li>— Framing Error</li> <li>— Parity Error</li> <li>— Overrun Error</li> <li>— Data ready to be read</li> </ul>	b7	--	b5	b4	b3	b2	b1	b0	↑		↑	↑	↑	↑	↑	↑																								
b7	--	b5	b4	b3	b2	b1	b0																																		
↑		↑	↑	↑	↑	↑	↑																																		
<b>AH=1 (01h)</b>	<p><b>Character Received By Modem</b></p> <p>The character in AL has just been received through the modem and is about to be processed by the AUX driver. The expansion code can inspect or alter the character before it is added to the input queue, or discard the character by returning FFh in AX.</p> <p>Register contents:</p> <ul style="list-style-type: none"> <li>AL = Character received from modem.</li> <li>BL = Modem status byte (same as "Serial Port Status Byte" above).</li> </ul>																																								

## 5.38 CON Expansion Interrupt (Int 5Eh)

The CONsole driver expansion interrupt (5Eh) provides a means by which user-written applications can preprocess characters being manipulated by the CONsole driver, perform additional Alpha Mode initialization, and perform additional Graphics Mode initialization. This makes it possible for an application to process additional escape sequences, remap characters before they are displayed, prohibit certain characters from being seen by the CONsole driver, preprocess keyhits, replace default fonts, and cause Alpha or Graphics mode to display a specific image immediately after it is reset.

The Interrupt 5Eh expansion code should check the AH register to determine why it was called. If the code in AH is not one of interest, an IRET should immediately be performed.



### Note

An application should always return to the driver by using IRET. This interrupt may be called in the middle of processing a keyboard matrix hardware interrupt. There are hardware-specific protocols that must be observed in order to clear the interrupt.

An application must *not* alter any registers unless noted otherwise.



### Note

CON expansion code must not call any built-in system drivers via standard MS-DOS protocol. In order to output text to the display from within the expansion code, for example, you must use Video I/O Interrupt 10h, Fast Video Interrupt 5Fh, or System Utility Interrupt 50h rather than the usual MS-DOS Interrupt 21h. The expansion code should also take care to avoid recursion; CON output via service interrupt 50h (and in some cases 5Fh) can cause the code to be entered recursively.

Table 5-12 lists the functions performed by this interrupt.

**Table 5-12. CON Expansion Interrupt 5Eh Functions**

Function	Description
AH=0 (00h)	<p><b>Output Character About To Be Processed</b></p> <p>The character in AL is about to be processed by the CONsole output driver. The expansion code can inspect, alter, or trap the character before the driver sees it. When the expansion routine terminates (via IRET), the driver examines CX to determine what it should do. If the expansion code returns CX=-1, the character in AL is discarded; if CX=0, the character in AL is passed on to the output driver (and escape sequence parser) for normal processing; if CX&gt;0, the character in AL is processed normally and then the expansion code is rerun with the same character in AL, AH=0, and CX still set to the same positive value. The first time the expansion code is called with any given character, CX=0.</p> <pre> graph LR     A[Set CX := 0] --&gt; B[Set AH := 0]     B --&gt; C[CON EXP Int 5Eh]     C --&gt; D{Is CX = -1?}     D -- Y --&gt; E[Output AL]     E --&gt; F[Set AH := 4]     F --&gt; G[CON EXP Int 5Eh]     G --&gt; H{Is CX = 0?}     H -- N --&gt; I[Output AL]     H -- Y --&gt; A   </pre>

The AH=0 CON Expansion interrupt was designed as one possible way to add additional escape sequences to the CON driver. In general, the expansion code examines incoming characters and optionally passes them through or traps them in a buffer; CX can eventually be used as a loop counter if it becomes necessary to pass a string of buffered characters to the escape parser/display driver for normal processing in one fell swoop.

Table 5-12. CON Expansion Interrupt 5Eh Functions (Continued)

AH=1 (01h)	<b>Alpha Initialization Completed</b>  The display has just been put into Alpha mode, but it has not yet been turned on. The display has been cleared, the cursor is at the home position, and all of the fonts for the current mode have been reloaded. Expansion code could be used here to replace default fonts with new ones, enable Arabic mode, reposition the cursor, or display text.
AH=2 (02h)	<b>Graphics Initialization Completed</b>  The display has just been put into Graphics mode and cleared, but it has not yet been turned on. The graphics cursor is off, the relocatable origin is at (0,0), drawing mode is "set pixels", the linetype is 1 (solid), and the pen is up.
AH=3 (03h)	<b>Matrix Key About To Be Processed</b>  A matrix key has been hit; its scancode is in DX, and the current states of the three modifier keys are in the low three bits of AL. At this point, the expansion code has the option of altering either or both of these registers to make the key look like another key (by changing the scancode in DX and/or the modifiers in AL). If the expansion code passes back a -1 in AX, normal matrix key processing is bypassed and the keystroke will appear to have never happened.  Register contents: AL = Bit 0: State of the <b>CTRL</b> key. Bit 1: State of the <b>Shift</b> key. Bit 2: State of the <b>Extend</b> key. DX = Scancode of matrix key (0-71).

**Table 5-12. CON Expansion Interrupt 5Eh Functions (Continued)**

AH=4 (04h)	<b>Display Character Just Processed</b>  The character in AL has just been processed by the CONsole output driver. CX contains the same value it had at the completion of the AH=0 expansion call (see the diagram for CON Expansion Function 0). If, at this time, CX is passed back to the CON driver equal to zero, output processing concludes normally. If CX is set to some non-zero value, the character in AL will be resubmitted to the display output routine (and escape sequence parser), passing once again through the AH=0 expansion call.
AH=5 (05h)	<b>Display Character and Attribute About To Be Stored</b>  A character and attribute are about to be stored in display RAM (and consequently displayed on the LCD). Expansion code has the option of altering the character and attribute in BX. <b>Register contents:</b> BL = Character code. BH = Attribute byte. ES:DI = LCD address at which character and attribute will be stored.

**Example:** This program adds the escape sequence "ESC &a?" to the CONsole driver. After running this program (which stays resident), the printing of that escape sequence to the console will cause the string "HP Portable PLUS..." to be printed in the escape sequence's place. There are three states in which the program can be:

- State 0: No escape sequence is being printed (parsed).
- State 1: Parsing an escape sequence that matches (so far) the one we're looking for.
- State -1 (FFh): Printing a string of characters; either the previous part of an escape sequence we didn't match (which we have saved in a buffer), or we're printing the replacement string "HP Portable PLUS..." .

The console expansion interrupt will not work in conjunction with another program which also tries to do console expansion, such as the TERM program on B:\BIN. To test this example, run the program listed below, then (in MS-DOS commands) type

## ECHO0 **(ESC)** &&a? **(Return)**

The first ampersand disappears and causes the escape character to be displayed--but you have to type another ampersand for the escape sequence.

```
page 60,132
title CONSOLE EXPANSION --- adding an escape sequence to the console driver
;
cseg    segment para public 'code'
        assume cs:cseg, ds:cseg
        org     100h
start   proc    far
        jmp    init           ; do the init code
start   endp
;
; This is the CONSOLE EXPANSION interrupt (5Eh) service routine. We're only
; concerned with console-expansion-call 0 (AH=0) which is character output.
;
int5e:
        push   ds             ; save data seg reg
        push   es
        push   cs             ; set up our data segment
        pop    ds
        push   cs             ; set up our extra segment
        pop    es
        cmp    ah,0            ; char about to be processed?
        jnz   bailout
        cmp    byte ptr state,0 ; are we parsing an escape sequence?
        jnz   instate
        cmp    cx,0            ; jif yes
        jnz   somebad          ; first time here?
        cmp    al,27            ; jif no, something wrong
        jnz   bailout          ; character to be output an escape?
        mov    word ptr buffptr,0 ; jif no, ignore it, let CON do it
        mov    byte ptr state,1 ; init buffer pointer
        mov    byte ptr state,1 ; set state to parsing escape seq
savechar:
        push   di             ; save register
        mov    di,buffptr      ; get buffer pointer
        mov    [di+buffer],al   ; push char into buffer
        inc    di
        cmp    byte ptr [di+escseq],0 ; got entire escape sequence?
        jz    itsours          ; jif yes, now do our thing
```

```

        mov    buffptr,di          ; save new ptr value
        pop    di
somebad:
        mov    cx,0ffffh          ; tell CON to ignore it
bailout:
        pop    es
        pop    ds          ; restore data seg reg
        iret              ; return back to con driver
itsours:
        mov    word ptr buffptr,offset ourmsg ; address of our message
        mov    byte ptr state,0ffh      ; set state to output mode
        pop    di          ; restore register
outloop:
        mov    cx,1          ; set flag for CON
        push   si          ; save register
        mov    si,buffptr      ; get pointer to next char
        lodsb              ; get next char of output
        mov    buffptr,si      ; save output ptr
        pop    si          ; restore reg
        cmp    al,0          ; done?
        jnz    bailout       ; jif no
        mov    byte ptr state,0      ; clear state flag
        jmp    somebad       ; exit, with CON flag to ignore
instate:
        cmp    byte ptr state,0ffh      ; are we in output state?
        jz     outloop       ; jif yes
        push   di          ; save register
        mov    di,buffptr      ; get buffer ptr
        cmp    al,[di+escseq]      ; does new char match next char?
        jnz    notours      ; jif no, dump our buffer
        pop    di          ; restore reg
        jmp    savechar      ; put the char in our buffer
notours:
        mov    [di+buffer],al      ; save the char
        inc    di
        mov    byte ptr [di+buffer],0 ; terminate buffer
        mov    word ptr buffptr,offset buffer ; point to the start of saved
        mov    byte ptr state,0ffh      ; set output state
        pop    di          ; restore register
        jmp    outloop       ; output what we've saved
;
;

```

```
buffer db 32 dup (?) ; escape sequence buffer
buffptr dw ?
state db ?
escseq db 27,'&a?',0 ; escape sequence we're adding
ourmsg db 'HP Portable PLUS...',13,10,0 ; our message
;
;
endkeep: ; end of resident code
;
;
init:
    mov byte ptr state,0 ; normal state: not parsing
    mov dx,offset int5e ; address of our int 5e ISR
    mov ax,255eh ; take the 5e interrupt vector
    int 21h
    mov dx,offset endkeep ; address of end of code to keep
    mov cl,4
    shr dx,cl ; get size in 16-byte paragraphs
    inc dx ; round up one for remainder
    mov ax,3100h ; terminate but stay resident
    int 21h ; with ERRORLEVEL = 0
;
cseg ends
end start
```

## 5.39 Fast Video Interrupt (Int 5Fh)

The liquid-crystal display (LCD) can be quickly manipulated through calls to the Fast Video service interrupt (5Fh). This interrupt provides facilities for cursor positioning, character and/or attribute reading and writing, management of user-defined windows, and various graphics functions such as pixel placement and line drawing.

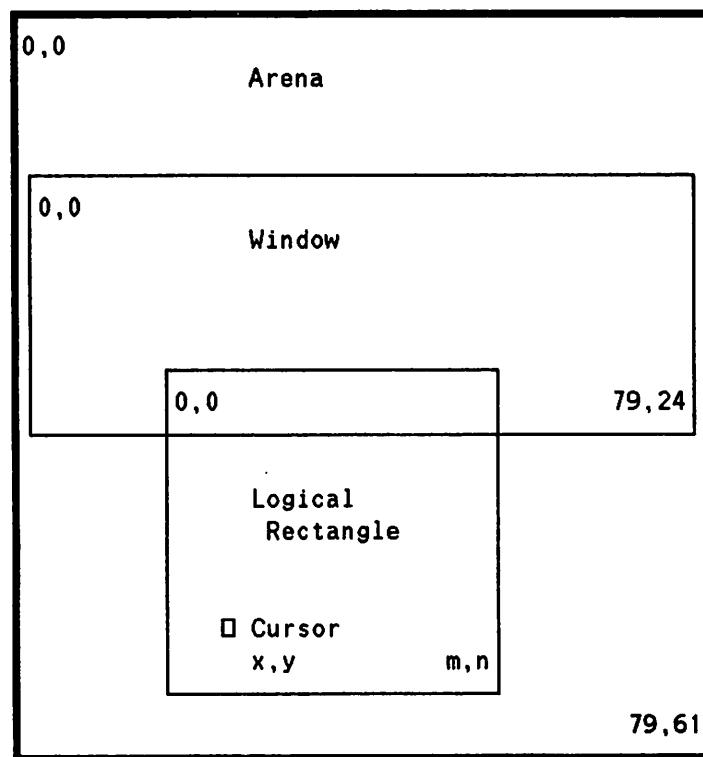
Fast Video is composed of two distinct groups of functions: *Fast Alpha* and *Fast Graphics*.

### 5.39.1 Fast Alpha

Fast Alpha operates on four structures as shown in figure 5-2.

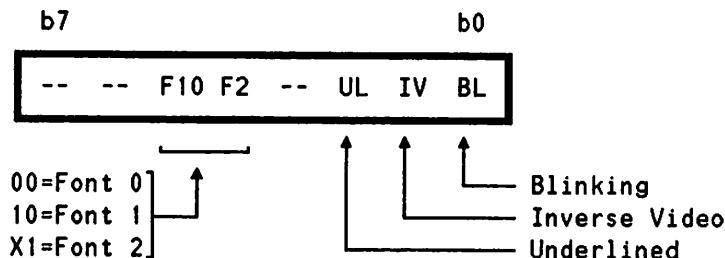
- The *Arena* is the display database, the display RAM in which the Window and Logical Rectangle (defined below) can be moved. The Arena bounds are fixed by hardware at 62 lines of 80 characters per line. Most Fast Alpha coordinates are Arena-relative, with (0,0) being the (row,column) of the upper left corner. The Arena is of fixed size and origin; it cannot be altered.
- The *Window* is the physical display, the area within the Arena that the user actually sees. The Window size is fixed by hardware at 25 lines of 80 characters per line (except when softkey labels are displayed -- it is then 23 lines). The Window origin, however, is variable in the Y (or row) direction and can be moved to any Arena-relative coordinate with the constraint that no part of the Window extends outside the Arena.
- The *Logical Rectangle* is the Fast Alpha workspace. Nearly all Fast Alpha routines manipulate this region or its contents. The user can define both the size and the arena-relative origin of the LR (with the constraint that no part of the LR extends outside the Arena); note that the LR and the Window are completely independant.
- The *Cursor* is a pointer to a particular row and column within the current Logical Rectangle. The cursor moves within the bounds of the LR; it may or may not be visible within the Window.

**Figure 5-2. Fast Alpha Structures**



Many of the Fast Alpha functions deal with characters and attributes. A character can take on any value from 0 to 255 (one byte); each character byte is stored in display RAM with an associated attribute byte. The attribute byte defines the various characteristics of the character, as shown in figure 5-3. (Note that bits 5 and 4 aren't interpreted as a binary form.)

**Figure 5-3. Display Attribute Byte**



5



**Caution** Use caution in mixing Fast Alpha, Video I/O Interrupt 10h, and CON output. The CONsole driver makes internal use of several Fast Alpha facilities, and may subsequently corrupt current Logical Rectangle, Cursor, and Window settings. Video I/O Interrupt 10h calls may similarly affect Fast Alpha and CONsole driver variables. Also, while Fast Alpha permits the cursor to be moved off screen, anything output via the CONsole driver may force the window to be adjusted to bring the cursor back into view.

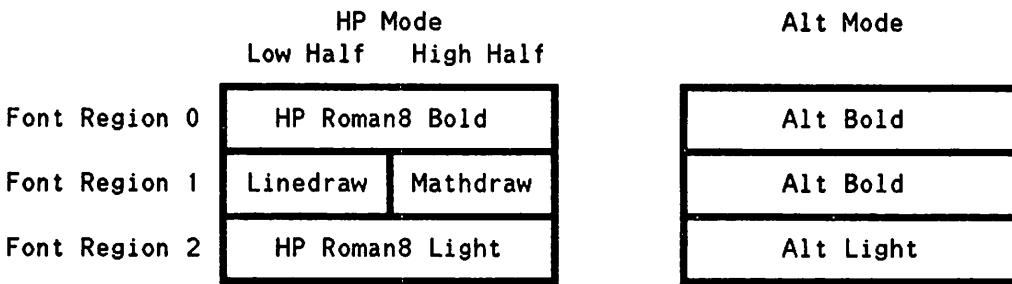
Display RAM resides at segment 8000h. Fast alpha places the first line of the Arena at 8000:0200h and extends down 62 lines to 8000:3F00h. Display RAM from 8000:0000h through 8000:01FFh is reserved for softkey label storage, is not touched by any Fast Alpha routines, and should not be used. (If you prefer to think of the first line as offset 0000h, consider alpha memory as starting at segment 8020h instead of 8000h.)

Alpha mode can simultaneously support three 256-character fonts (or six 128-character fonts, depending how you look at it). These fonts are stored in a reserved area of display RAM which is subdivided into three 256-character regions: LCD font regions 0, 1, and 2. When an alpha character is stored in display RAM, two bits of its attribute byte specify which LCD font region is to be used; the sign bit of the character

byte specifies whether that character is in the low half or high half of that region. (Refer also to "Display Controller" in chapter 7.)

Fast Alpha provides the user with total access to the character fonts in any of the three regions by requiring the user to explicitly specify each character's attribute byte. This means that, in order to achieve desired results, the user must know precisely what font resides in each of the font regions. The exact arrangement of fonts is determined by the Initial Font Load table in the configuration EPROM, plus any font changes that may have been made by CON Expansion code or by the user's application itself. Barring any such changes, the Initial Font Load will look like that shown in figure 5-4.

**Figure 5-4. Initial Font Load**



The configuration EPROM provides separate Initial Font Load tables for HP and Alt modes.

Video I/O Interrupt 10h (described earlier in this chapter) generally accesses only fonts in region 0. The exception here is graphics mode; characters with codes of 128-255 come from a user-supplied table pointed to by a double word vector for interrupt 1Fh.

Standard CON output works in terms of Primary and Alternate fonts. The primary font generally resides in Font Region 0; the alternate font is in Font Region 1. The characters in Font Region 2 are stick figures used to mimic the halfbright enhancement; if halfbright is enabled, characters that normally use fonts from Region 0 and 1 are automatically mapped into Font Region 2. (The only place this could present a problem is in the halfbright enhancement of HP mode line and math symbols, which have no equivalents in Font Region 2.)

Whenever alpha mode is initialized (at system reboot, in response to "ESC E", or at exit from graphics mode) the font tables are reloaded as directed by the Initial Font Load tables in the configuration EPROM. After everything has been initialized, but before

the display is turned on, CON Expansion code is given a chance to alter (remove, modify, or add to) the newly loaded fonts. The first font that was loaded becomes the Primary Font. The second font loaded becomes the Alternate Font.

In display RAM, fonts are stored in an eight-byte format in which only the high six bits of each byte are significant. When installing fonts, you have the option of specifying the new font table in either the same eight-byte form or in a more space-efficient six-byte (packed) form. The two formats are shown in figure 5-5.

**Figure 5-5. Font Formats**

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

6x8 Matrix  
Unpacked Format  
(8 bytes)

Significant	Significant
0 0 0 0 0 0	0 0
0 0 1 1 1 1	1 1
1 1 1 1 2 2	2 2
2 2 2 2 2 2	2 2
3 3 3 3 3 3	3 3
3 3 4 4 4 4	4 4
4 4 4 4 5 5	5 5
5 5 5 5 5 5	5 5

6x8 Matrix  
Packed Format  
(6 bytes)

In the unpacked case, bytes 0 through 7 define eight six-bit dotrows from top to bottom of the 6x8 matrix; the rightmost two bits of each byte are ignored. In the packed case, bytes 0 through 5 define the same eight dotrows in a raster-scan format, starting with the most significant bit of the first byte at the upper left corner of the matrix. Subsequent bits are found by moving to the right, wrapping to the left end of the next dotrow down as the matrix becomes filled.

All of the ROM-resident system default fonts are stored in the packed format. A word in the configuration EPROM specifies the segment address of the first system font table; subsequent font tables follow.

Graphics mode consumes all of the font regions for use as display RAM. Therefore, an appropriate escape sequence or service function should always be used to switch from graphics back to alpha to insure that the correct fonts will be reloaded.

Table 5-13 lists the Fast Alpha functions for Interrupt 5Fh. Specify the desired function code in AH, with additional parameters passed in other registers as indicated. All registers except AX are preserved unless otherwise noted.

A subset of the Fast Alpha functions described in table 5-13 can be accessed from high-level languages such as Lattice C, MS Pascal, and MS Fortran through libraries that will be provided with these products. Two such libraries provide a standard interface that allows rapid output while maintaining portability of code. The HP-UX Fast Alpha library (fa.lib) provides compatibility among the Portable PLUS, the Integral PC, and other HP computers featuring HP-UX. This library is designed so that it can be implemented on MS-DOS computers. The PCD library (pcdfa.lib) offers greater functionality, but provides compatibility between only the Portable PLUS and the Integral PC. This library probably won't be implemented on future HP computers.

Table 5-13. Fast Video Interrupt 5Fh Alpha Functions

Function	Description
AH=0 (00h)	<b>Get Arena (GETAR)</b>  Returns the size of the Fast Alpha Arena. <b>Returns:</b> CL = Arena width (80 columns). CH = Arena height (62 lines).
AH=1 (01h)	<b>Get Window (GETW)</b>  Returns the current Window size and origin. <b>Returns:</b> BL = Window X-origin (Arena column 0). BH = Window Y-origin (Arena line, 0-61). CL = Window width (80 columns). CH = Window height (23/25 lines with/without softkey labels).

**Table 5-13. Fast Video Interrupt 5Fh Alpha Functions (Continued)**

AH=2 (02h)	<b>Get Logical Rectangle (GETLR)</b>  Returns the current Logical Rectangle size and origin. Returns: BL = LR X-origin (Arena column, 0-79). BH = LR Y-origin (Arena line, 0-61). CL = LR width (columns). CH = LR height (lines).
AH=3 (03h)	<b>Get Cursor Position (GETCUR)</b>  Reads the current LR-relative cursor position. Returns: BL = Cursor column (LR column). BH = Cursor row (LR line).
AH=4 (04h)	<b>Set Window (SETW)</b>  Repositions the Window within the Arena. If any part of the new Window falls outside of the Arena, the desired Window origin will be adjusted to ensure that the entire window is within the Arena bounds. Specify: BL = Desired Window X-origin (Arena column, 0-79 -- ignored). BH = Desired Window Y-origin (Arena line, 0-61). Returns: BL = Actual Window X-origin (Arena column 0). BH = Actual Window Y-origin (Arena line).

**Table 5-13. Fast Video Interrupt 5Fh Alpha Functions (Continued)**

AH=5 (05h)	<b>Set Logical Rectangle (SETLR)</b>  Redefines the current LR. If any part of the new LR falls outside the Arena, the LR size will be clipped to fit within the Arena bounds. The cursor will be positioned at the upper left corner (LR coordinate [0,0]) of the new LR.  <b>Specify:</b> BL = Desired LR X-origin (Arena column, 0-79). BH = Desired LR Y-origin (Arena line, 0-61). CL = Desired LR width (columns). CH = Desired LR height (lines).  <b>Returns:</b> BL = Actual LR X-origin (Arena column). BH = Actual LR Y-origin (Arena line). CL = Actual LR width (columns). CH = Actual LR height (lines).
AH=6 (06h)	<b>Set Cursor Position (SETCUR)</b>  Moves the cursor to a new LR coordinate. If the new position is outside of the current LR, nothing happens.  <b>Specify:</b> AL = 0 Turn off cursor. 1 Turn on primary cursor. 2 Turn on block cursor as primary. 3 Turn on underscore cursor as primary. > 3 Don't change cursor. BL = Cursor column (LR column). BH = Cursor row (LR line).
AH=7 (07h)	<b>Read Character (GETC)</b>  Reads the single character at the specified LR coordinate. The cursor does not move.  <b>Specify:</b> BL = LR column. BH = LR line.  <b>Returns:</b> DL = Character at specified LR coordinate.

**Table 5-13. Fast Video Interrupt 5Fh Alpha Functions (Continued)**

AH=8 (08h)	<b>Read Attribute (GETA)</b>  Reads the single attribute at the specified LR coordinate. The cursor does not move. <b>Specify:</b> BL = LR column. BH = LR line. <b>Returns:</b> DH = Attribute at specified coordinate.
AH=9 (09h)	<b>Read Attribute/Character (GETCA)</b>  Reads the single attribute and character at the specified LR coordinate. The cursor does not move. <b>Specify:</b> BL = LR column. BH = LR line. <b>Returns:</b> DL = Character at specified coordinate. DH = Attribute at specified coordinate.
AH=10 (0Ah)	<b>Read Character at Cursor (GETCM)</b>  Reads the single character at the current cursor position. The cursor advances to the next LR position, possibly causing the LR to scroll up one line. <b>Returns:</b> DL = Character at cursor.
AH=11 (0Bh)	<b>Read Attribute at Cursor (GETAM)</b>  Reads the single attribute at the current cursor position. The cursor advances to the next LR position, possibly causing the LR to scroll up one line. <b>Returns:</b> DH = Attribute at cursor.

**Table 5-13. Fast Video Interrupt 5Fh Alpha Functions (Continued)**

AH=12 (0Ch)	<b>Read Attribute/Character at Cursor (GETCAM)</b>  Reads the single attribute and character at the current cursor position. The cursor advances to the next LR position, possibly causing the LR to scroll up one line.  Returns: DL = Character at cursor. DH = Attribute at cursor.
AH=13 (0Dh)	<b>Write Character (PUTC)</b>  Writes a single character at a specified LR coordinate. The existing attribute at that position stays unchanged; the cursor does not move.  Specify: BL = LR column. BH = LR line. DL = Character to be displayed.
AH=14 (0Eh)	<b>Write Attribute (PUTA)</b>  Writes a single attribute at a specified LR coordinate. The existing character at that position stays unchanged; the cursor does not move.  Specify: BL = LR column. BH = LR line. DH = Attribute to be displayed.
AH=15 (0Fh)	<b>Write Attribute/Character (PUTCA)</b>  Writes a single attribute and character at a specified LR coordinate. The cursor does not move.  Specify: BL = LR column. BH = LR line. DL = Character to be displayed. DH = Attribute to be displayed.

**Table 5-13. Fast Video Interrupt 5Fh Alpha Functions (Continued)**

AH=16 (10h)	<b>Write Character at Cursor (PUTCM)</b>  Writes a single character at the current cursor position. The existing attribute at that position stays unchanged; the cursor advances to the next LR position, possibly causing the LR to scroll up one line. Specify: DL = Character to be displayed.
AH=17 (11h)	<b>Write Attribute at Cursor (PUTAM)</b>  Writes a single attribute at the current cursor position. The existing character at that position stays unchanged; the cursor advances to the next LR position, possibly causing the LR to scroll up one line. Specify: DH = Attribute to be displayed.
AH=18 (12h)	<b>Write Attribute/Character at Cursor (PUTCAM)</b>  Writes a single attribute and character at the current cursor position. The cursor advances to the next LR position, possibly causing the LR to scroll up one line. Specify: DL = Character to be displayed. DH = Attribute to be displayed.
AH=19 (13h)	<b>Read Character String (GETCS)</b>  Reads a string of characters from the current LR, starting at the specified coordinate. Attempts to read past the end of a line will wrap to the next line; reading past the bottom of the LR will produce blanks. The cursor does not move; text within the LR never scrolls. Specify: BL     = LR column. BH     = LR line. CX     = Number of characters to read. DS:SI = Address of buffer to receive string.

**Table 5-13. Fast Video Interrupt 5Fh Alpha Functions (Continued)**

AH=20  
(14h)

**Read Attribute String (GETAS)**

Reads a string of attributes from the current LR, starting at the specified coordinate. Attempts to read past the end of a line will wrap to the next line; reading past the bottom of the LR will produce blanks. The cursor does not move; text within the LR never scrolls.

Specify:

- BL = LR column.
- BH = LR line.
- CX = Number of bytes to read.
- DS:DI = Address of buffer to receive string.

AH=21  
(15h)

**Read Character and Attribute Strings (GETCAS)**

Reads a string of attributes and characters from the current LR, starting at the specified coordinate. Attempts to read past the end of a line will wrap to the next line; reading past the bottom of the LR will produce blanks. The cursor does not move; text within the LR never scrolls.

Specify:

- BL = LR column.
- BH = LR line.
- CX = Number of characters to read.
- DS:DI = Address of buffer to receive attribute string.
- DS:SI = Address of buffer to receive character string.

AH=22  
(16h)

**Read Character String at Cursor (GETCSM)**

Reads a string of characters from the current LR, starting at the current cursor position. Attempts to read past the end of a line will wrap to the next line; reading past the bottom of the LR will produce blanks. The cursor advances with each character read, possibly scrolling the text in the LR.

Specify:

- CX = Number of characters to read.
- DS:SI = Address of buffer to receive character string.

**Table 5-13. Fast Video Interrupt 5Fh Alpha Functions (Continued)**

AH=23 (17h)	<b>Read Attribute String at Cursor (GETASM)</b>  Reads a string of attributes from the current LR, starting at the current cursor position. Attempts to read past the end of a line will wrap to the next line; reading past the bottom of the LR will produce blanks. The cursor advances with each attribute read, possibly scrolling the text in the LR. <b>Specify:</b> CX = Number of bytes to read. DS:DI = Address of buffer to receive attribute string.
AH=24 (18h)	<b>Read Character and Attribute Strings at Cursor (GETCASM)</b>  Reads a string of attributes and characters from the current LR, starting at the current cursor position. Attempts to read past the end of a line will wrap to the next line; reading past the bottom of the LR will produce blanks. The cursor advances with each character read, possibly scrolling the text in the LR. <b>Specify:</b> CX = Number of characters to read. DS:DI = Address of buffer to receive attribute string. DS:SI = Address of buffer to receive character string.
AH=25 (19h)	<b>Write Character String (PUTCS)</b>  Writes a string of characters into the current LR, starting at the specified coordinate. Attempts to write past the end of a line will wrap to the next line; writing past the bottom of the LR will be ignored. The cursor does not move; text within the LR never scrolls. <b>Specify:</b> BL = LR column. BH = LR line. CX = Number of characters to write. DS:SI = Address of character string buffer.

**Table 5-13. Fast Video Interrupt 5Fh Alpha Functions (Continued)**

AH=26  
(1Ah)

**Write Attribute String (PUTAS)**

Writes a string of attributes into the current LR, starting at the specified coordinate. Attempts to write past the end of a line will wrap to the next line; writing past the bottom of the LR will be ignored. The cursor does not move; text within the LR never scrolls. Before being written to the display, the specified attribute byte is ANDed with the mask in DH, then XORed with a second mask in DL.

Specify:

- BL = LR column.
- BH = LR line.
- CX = Number of bytes to write.
- DH = First attribute mask (AND).
- DL = Second attribute mask (XOR).
- DS:DI = Address of attribute string buffer.

AH=27  
(1Bh)

**Write Character and Attribute Strings (PUTCAS)**

Writes a string of attributes and characters into the current LR, starting at the specified coordinate. Attempts to write past the end of a line will wrap to the next line; writing past the bottom of the LR will be ignored. The cursor does not move; text within the LR never scrolls. Before being written to the display, the specified attribute byte is ANDed with the mask in DH, then XORed with a second mask in DL.

Specify:

- BL = LR column.
- BH = LR line.
- CX = Number of characters to write.
- DH = First attribute mask (AND).
- DL = Second attribute mask (XOR).
- DS:DI = Address of attribute string buffer.
- DS:SI = Address of character string buffer.

**Table 5-13. Fast Video Interrupt 5Fh Alpha Functions (Continued)**

AH=28  
(1Ch)

**Write Character String With One Attribute (PUTCOS)**

Writes a string of characters with a single attribute into the current LR, starting at the specified coordinate. Attempts to write past the end of a line will wrap to the next line; writing past the bottom of the LR will be ignored. The cursor does not move; text within the LR never scrolls.

Specify:

- BL = LR column.
- BH = LR line.
- CX = Number of characters to write.
- DH = Attribute.
- DS:SI = Address of character string buffer.

AH=29  
(1Dh)

**Write Character String at Cursor (PUTCSM)**

Writes a string of characters into the current LR, starting at the current cursor position. Attempts to write past the end of a line will wrap to the next line. The cursor advances with each character written, possibly scrolling the text in the LR.

Specify:

- CX = Number of characters to write.
- DS:SI = Address of character string buffer.

AH=30  
(1Eh)

**Write Attribute String at Cursor (PUTASM)**

Writes a string of attributes into the current LR, starting at the current cursor position. Attempts to write past the end of a line will wrap to the next line. The cursor advances with each character written, possibly scrolling the text in the LR. Before being written to the display, the specified attribute byte is ANDed with the mask in DH, then XORed with a second mask in DL.

Specify:

- CX = Number of bytes to write.
- DH = First attribute mask (AND).
- DL = Second attribute mask (XOR).
- DS:DI = Address of attribute string buffer.

**Table 5-13. Fast Video Interrupt 5Fh Alpha Functions (Continued)**

AH=31 (1Fh)	<b>Write Character and Attribute Strings at Cursor (PUTCASM)</b>  Writes a string of characters and attributes into the current LR, starting at the current cursor position. Attempts to write past the end of a line will wrap to the next line. The cursor advances with each character written, possibly scrolling the text in the LR. Before being written to the display, the specified attribute byte is ANDed with the mask in DH, then XORed with a second mask in DL.  Specify: CX = Number of characters to write. DH = First attribute mask (AND). DL = Second attribute mask (XOR). DS:DI = Address of attribute string buffer. DS:SI = Address of character string buffer.
AH=32 (20h)	<b>Write Character String With One Attribute at Cursor (PUTCOSM)</b>  Writes a string of characters with a single attribute into the current LR, starting at the current cursor position. Attempts to write past the end of a line will wrap to the next line. The cursor advances with each character written, possibly scrolling the text in the LR.  Specify: CX = Number of characters to write. DH = Attribute. DS:SI = Address of character string buffer.
AH=33 (21h)	<b>Scroll Window (TERMSCROLL)</b>  Scrolls the characters and attributes in the Window by moving the window relative to the Arena. (Note that since the Window width equals the Arena width, left and right Window scrolling is meaningless.)  Specify: AL = Scroll direction ("u"p, "d"own, "l"eft, or "r"ight). CL = Number of lines/columns to scroll.

**Table 5-13. Fast Video Interrupt 5Fh Alpha Functions (Continued)**

AH=34 (22h)	<b>Scroll Logical Rectangle (SCROLLN)</b>  Scrolls characters and attributes within the Logical Rectangle. The vacated lines or columns are filled with blanks. <b>Specify:</b> AL = Scroll direction ("u"p, "d"own, "l"eft, or "r"ight). CL = Number of lines/columns to scroll.
AH=35 (23h)	<b>Scroll/Fill Logical Rectangle (SCROLLNA)</b>  Scrolls characters and attributes within the Logical Rectangle. The vacated lines or columns are filled with blanks and the specified attribute. <b>Specify:</b> AL = Scroll direction ("u"p, "d"own, "l"eft, or "r"ight). CL = Number of lines/columns to scroll. DH = Attribute to use for blank fill.
AH=36 (24h)	<b>Scroll Logical Rectangle One Line (SCROLL)</b>  Scrolls characters and attributes within the Logical Rectangle one line or column. The vacated line or column is filled with blanks. <b>Specify:</b> AL = Scroll direction ("u"p, "d"own, "l"eft, or "r"ight).
AH=37 (25h)	<b>Scroll/Fill Logical Rectangle One Line (SCROLLA)</b>  Scrolls characters and attributes within the Logical Rectangle one line or column. The vacated line or column is filled with blanks and the specified attribute. <b>Specify:</b> AL = Scroll direction ("u"p, "d"own, "l"eft, or "r"ight). DH = Attribute to use for blank fill.

**Table 5-13. Fast Video Interrupt 5Fh Alpha Functions (Continued)**

AH=38 (26h)	<b>Fill Logical Rectangle With Character (FILLC)</b>  Fills the entire current Logical Rectangle with a single character and no attribute. <b>Specify:</b> DL = Fill character.
AH=39 (27h)	<b>Fill Logical Rectangle With Attribute (FILLA)</b>  Fills the entire current Logical Rectangle with a specified attribute. <b>Specify:</b> DH = Fill attribute.
AH=40 (28h)	<b>Fill Logical Rectangle With Attribute/Character (FILLCA)</b>  Fills the entire current Logical Rectangle with a single character and a specified attribute. <b>Specify:</b> DL = Fill character. DH = Fill attribute.
AH=41 (29h)	<b>Read Character Font (GETFONT)</b>  Reads a single character font from the LCD font storage area. The character's font matrix is returned in eight bytes. Only the leftmost six bits of each byte are significant. <b>Specify:</b> CX = 1. DL = Character code of initial character (0-255). DH = LCD font region (0-2). DS:SI = Pointer to start of buffer to receive font matrix.

**Table 5-13. Fast Video Interrupt 5Fh Alpha Functions (Continued)**

AH=42  
(2Ah)

**Write Character Font (PUTFONT)**

Writes one or more character fonts into the LCD font storage area. Each character's matrix is specified in either six or eight bytes (packed and unpacked formats). In unpacked (eight-byte) format, only the leftmost six bits of each byte are significant. Packed (six-byte) format eliminates the two unused bits from each byte, shifting subsequent bytes into the vacated bit positions. You can specify either a user font table or a system ROM font table.

Specify:

- AL < 0 Use packed user table (6-byte format).
  - = 0 Use unpacked user table (8-byte format).
  - = 1 Use HP Bold characters 0-127 (ROM font table 1).
  - = 2 Use HP Bold characters 128-255 (ROM font table 2).
  - = 3 Use HP Thin characters 0-127 (ROM font table 3).
  - = 4 Use HP Thin characters 128-255 (ROM font table 4).
  - = 5 Use Alt Bold characters 0-127 (ROM font table 5).
  - = 6 Use Alt Bold characters 128-255 (ROM font table 6).
  - = 7 Use Alt Thin characters 0-127 (ROM font table 7).
  - = 8 Use Alt Thin characters 128-255 (ROM font table 8).
  - = 9 Use Line Draw characters 0-127 (ROM font table 9).
  - =10 Use Math Draw characters 0-127 (ROM font table 10).
- CX = Number of characters to write.  
DL = Initial character code within LCD font region (0-255).  
DH = LCD font region (0-2).  
DS:SI = Pointer to user font table (if AL<=0).

## 5.39.2 Fast Graphics

Fast Graphics functions work in essentially the same space as Video Interrupt 10h graphics, except that the origin (0,0) is at the lower left of a 480x200 display area rather than at the upper left as it is in the Int 10h service. The X axis increases to the right; the Y axis increases upward.

Display RAM resides at segment 8000h. In graphics mode, the byte at the upper left corner of the display is at 8000:0000h, with byte addresses increasing as you move to the right; the rightmost byte on the top line is therefore at 8000:004Bh. The next dotrow begins at 8000:0050h. The dotrow pitch is 64 bytes/line, but only the first 60 bytes are visible.

The Fast Graphics functions support two special pointing devices: a graphics "pen", and a graphics "cursor". *The two pointers are not the same.* The graphics cursor is similar to the alpha cursor, except that it appears on the display as a small arrow pointing up and to the left. When the keyboard is in normal mode, the graphics cursor can be manually positioned using the  $\leftarrow$ ,  $\uparrow$ ,  $\downarrow$ , and  $\rightarrow$  keys. The graphics cursor can be programmatically turned on and off, moved to new coordinates, and have its current coordinates read. Additionally, any text that is sent to the graphics display will appear at the graphics cursor.

The graphics pen is *never* visible. It serves only as a pointer to the final endpoint of the last line drawn using the Write Pen function (AH=52) or a line-drawing escape sequence. To draw a line using the graphics pen, you specify only the target coordinates; the line is drawn from the current pen position to the specified coordinate, and the pen is then moved to the new endpoint in preparation to draw another segment.

Table 5-14 lists the Fast Graphics functions for Interrupt 5Fh. Specify the desired function code in AH, with additional parameters in registers as required. Nothing is destroyed unless otherwise indicated.

**Table 5-14. Fast Video Interrupt 5Fh Graphics Functions**

Function	Description
<b>AH=43 (2Bh)</b>	<p><b>Reset Display (GRESET)</b></p> <p>Selects and initializes the specified display mode.</p> <p>Specify:</p> <ul style="list-style-type: none"><li>AL=0 Initialize Alpha mode.</li><li>#0 Initialize Graphics mode.</li></ul>
<b>AH=44 (2Ch)</b>	<p><b>Read Display Specs (GSPECS)</b></p> <p>Returns graphics display parameters.</p> <p>Returns in Alpha Mode:</p> <ul style="list-style-type: none"><li>AL=0 Alpha mode.</li><li>BX = Display RAM offset of top-left corner of screen.</li><li>DH = Lines on screen (23 or 25).</li><li>DL = Columns on screen (80).</li></ul> <p>Returns in Graphics Mode:</p> <ul style="list-style-type: none"><li>AL=1 Graphics mode.</li><li>BX = Display width (480).</li><li>DX = Display height (200).</li></ul>
<b>AH=45 (2Dh)</b>	<p><b>Graphics Erase (GERASE)</b></p> <p>Either sets or clears the graphics display, erasing or filling any image in that area. Note that the functions for clearing and setting the entire display will execute quickly; the coordinate-bound erase function will be considerably slower since it works on a pixel-by-pixel basis.</p> <p>Specify:</p> <ul style="list-style-type: none"><li>AL&lt;0 Clear area specified by BX, DX, SI, and DI.</li><li>=0 Clear entire display.</li><li>&gt;0 Set entire display.</li></ul> <p>BX = Area top left corner X coordinate (0-479) for AL&lt;0. DX = Area top left corner Y coordinate (0-199) for AL&lt;0. SI = Size in X direction for AL&lt;0. DI = Size in Y direction for AL&lt;0.</p>

**Table 5-14. Fast Video Interrupt 5Fh Graphics Functions (Continued)**

AH=46 (2Eh)	<b>Draw Line (GDRAW)</b>  Draws a line between the two specified absolute coordinates. Coordinate values can range from -32768 to +32767; lines that extend beyond any screen bounds ( $0 \leq X \leq 479$ and $0 \leq Y \leq 199$ ) will automatically be clipped at screen edges. (The graphics pen and cursor positions are unchanged.) Specify: BX = Starting point X coordinate (0-479). DX = Starting point Y coordinate (0-199). SI = Ending point X coordinate (0-479). DI = Ending point Y coordinate (0-199).
AH=47 (2Fh)	<b>Read Graphics Attributes (GRDATTR)</b>  Returns current graphics attributes. Returns: BL = Current line pattern (8-bit pixel string). BH = Current line scale factor. DL = Current drawing mode: 0 No change. 1 Clear pixels. 2 Set pixels (default). 3 Complement pixels. 4 Jam pixels.

**Table 5-14. Fast Video Interrupt 5Fh Graphics Functions (Continued)**

AH=48 (30h)	<b>Write Graphics Attributes (GWRATTR)</b>  Allows you to specify new graphics attributes. Note that you cannot use this function to change just the drawing mode; you must always specify a line pattern and scale factor as well. <b>Specify:</b> BL = New line pattern (8-bit pixel string). BH = New line scale factor. DL = New drawing mode: 0 No change. 1 Clear pixels. 2 Set pixels. 3 Complement pixels. 4 Jam pixels.
AH=49 (31h)	<b>Read Pixel (GRDDOT)</b>  Reads the state of the pixel at a given graphics coordinate. <b>Specify:</b> BX = Pixel X coordinate. DX = Pixel Y coordinate. <b>Returns:</b> AL = Pixel value (0=off, 1=on).
AH=50 (32h)	<b>Write Pixel (GWRDOT)</b>  Lets you alter a pixel at any given graphics coordinate according to the current drawing mode (function AH=48). <b>Specify:</b> AL = New pixel value (0=off, 1=on) if in "Jam pixels" mode. BX = Pixel X coordinate. DX = Pixel Y coordinate.

Table 5-14. Fast Video Interrupt 5Fh Graphics Functions (Continued)

AH=51 (33h)	<b>Read Pen (GRDPEN)</b>  Reads the state and current position of the graphics pen. (Note that the graphics pen is <i>not</i> the graphics cursor. The "cursor" is a visible, movable arrow pointer; the "pen" always points at the end of the most recently pen-drawn line.)  Returns: AL=0 Pen is currently raised. 1 Pen is currently lowered. BX = Current pen X coordinate. DX = Current pen Y coordinate.
AH=52 (34h)	<b>Write Pen (GWRPEN)</b>  Alters the state and current position of the graphics pen. If the pen is lowered before moving, a line will be drawn in accordance with the current linetype and drawing mode. (Note that the graphics pen is <i>not</i> the graphics cursor. The "cursor" is a visible, movable arrow pointer; the "pen" always points at the end of the most recently pen-drawn line.)  Specify: AL=2 Move, then raise pen. 1 Raise pen, then move. 0 Move pen without changing its height. -1 Lower pen, then move. -2 Move, then lower pen. BX = New pen X coordinate. DX = New pen Y coordinate.
AH=53 (35h)	<b>Read Graphics Cursor (GRDCUR)</b>  Reads the state and current position of the graphics cursor. Returns: AL=0 Cursor is off (not displayed). 1 Cursor is on (displayed). BX = Graphics cursor X coordinate. DX = Graphics cursor Y coordinate.

**Table 5-14. Fast Video Interrupt 5Fh Graphics Functions (Continued)**

AH=54 (36h)	<b>Write Graphics Cursor (GWRCUR)</b>  Alters the state and current position of the graphics cursor. The cursor will be moved as far as possible toward the newly specified coordinates; the actual updated coordinates will be returned in (BX,DX) (just in case the specified coordinates were out of bounds).  Specify: AL=0 Turn the cursor off. #0 Turn the cursor on. BX = New graphics cursor X coordinate. DX = New graphics cursor Y coordinate.  Returns: BX = Updated graphics cursor X coordinate. DX = Updated graphics cursor Y coordinate.
AH=55 (37h)	<b>Read Area (GRDAREA)</b>  Reads a block of pixels from graphics memory into a user-specified buffer. The area must be a multiple of eight pixels wide. Data will be read from the screen in a left-to-right, top-to-bottom order.  Specify: AL = Height of area (in lines, or pixels). CH = Width of area (in bytes, or 8-pixel groups). BX = Area top left corner X coordinate. DX = Area top left corner Y coordinate. DS:SI = Address of buffer to receive data (AL*CH bytes long).

**Table 5-14. Fast Video Interrupt 5Fh Graphics Functions (Continued)**

AH=56  
(38h)

**Write Area (GWRAREA)**

Writes a block of pixels from a user-specified buffer into graphics memory. The area must be a multiple of eight pixels wide. Data will be written to the screen in a left-to-right, top-to-bottom order and be displayed in accordance with the current drawing mode.

**Specify:**

- AL = Height of area (in lines, or pixels).
- CH = Width of area (in bytes, or 8-pixel groups).
- BX = Area top left corner X coordinate.
- DX = Area top left corner Y coordinate.
- DS:SI = Address of data buffer (AL\*CH bytes long).

AH=57  
(39h)

**Copy Area (GCOPY)**

Copies an area of the graphics display. This is a direct pixel- to-pixel transfer; current drawing mode is ignored.

**Specify:**

- AL = Height of area to copy (in pixels).
- CX = Width of area to copy (in pixels).
- SI = Source area top left corner X coordinate.
- DI = Source area top left corner Y coordinate.
- BX = Destination area top left corner X coordinate.
- DX = Destination area top left corner Y coordinate.

**Example:** This program demonstrates the use of fast video graphics calls in the BIOS. It allows line drawing (with rubber-band line), point plotting, and clearing of the graphics display. Softkeys are used to change plotting mode. When a mode is selected, an asterisk appears in the associated softkey label. When entering "line" mode, the current cursor position becomes the fixed end of a "rubber-band" line. As the cursor is moved, it is the other endpoint. The next time the LINE softkey is pressed, a permanent line is drawn where the rubber-band line was. When in "point" mode, each time the POINT softkey is pressed, the dot at the current cursor location is written. When in "draw" mode, draw are drawn automatically as the cursor is moved. All drawing is done by XORing a 1 with the current state of the pixel being drawn.

```

page 60,132
title GRAPHICS EXAMPLE --- using the BIOS fast video graphics calls
;
cseg    segment para public 'code'
        assume cs:cseg, ds:cseg
        org    100h
LINE    equ     0
POINT   equ     1
DRAW    equ     2
BIGJUMP equ     5
;
start   proc   far
restart:
        call   init           ; init program
keyloop:
        cmp    byte ptr mode,LINE      ; in line mode?
        jnz   keyl            ; jif no, don't unwrite line
        call   drawline        ; else undraw line
keyl:
        mov    dl,0ffh          ; get key
        mov    ah,6
        int    21h
        jz    keyl            ; jif no key
        cmp    al,27            ; escape code?
        jz    nextcode         ; jif yes, get next code
beep:
        mov    al,7dh          ; beep command
        mov    ah,1
        mov    bx,27            ; write command to PPU
        int    50h
        jmp   keyl            ; try again
nextcode:
        mov    dl,0ffh          ; see if second one
        mov    ah,6
        int    21h
        jz    beep             ; jif just escape key
        cmp    byte ptr mode,LINE      ; in line mode?
        jnz   norubber         ; jif no, don't unwrite line
        push   ax              ; save keycode
        call   drawline        ; else undraw line
        pop    ax              ; restore keycode
norubber:

```

```

        cmp     al,65          ; up cursor?
        jnz    not65          ; jif no
        call   setpen          ; move pen to cursor loc
        inc    dx              ; move cursor up

setcurs:
        mov     ax,3401h         ; raise pen then move
        cmp     byte ptr mode,DRAW  ; in draw mode?
        jnz    sc              ; jif no
        mov     al,0ffh          ; lower pen then move

sc:
        int    5fh              ; move pen, drawing if in DRAW mode
        mov     ax,3601h          ; write cursor
        int    5fh
        jmp    keyloop

not65:
        cmp     al,66          ; down cursor?
        jnz    not66          ; jif no
        call   setpen          ; move down
        dec    dx
        jmp    setcurs

not66:
        cmp     al,67          ; right cursor?
        jnz    not67          ; jif no
        call   setpen          ; move right
        inc    bx
        jmp    setcurs

not67:
        cmp     al,68          ; left cursor?
        jnz    not68          ; jif no
        call   setpen          ; move left
        dec    bx
        jmp    setcurs

not68:
        cmp     al,86          ; extend up cursor?
        jnz    not86          ; jif no
        call   setpen          ; big jump
        add    dx,BIGJUMP
        jmp    setcurs

not86:
        cmp     al,85          ; extend down cursor?
        jnz    not85          ; jif no
        call   setpen

```

5

```
        sub    dx,BIGJUMP           ; move down
        jmp    setcurs
not85:
        cmp    al,70                ; extend right cursor?
        jnz    not70               ; jif no
        call   setpen
        add    bx,BIGJUMP          ; move right
        jmp    setcurs
not70:
        cmp    al,104               ; extend left cursor?
        jnz    not104              ; jif no
        call   setpen
        sub    bx,BIGJUMP          ; move left
        jmp    setcurs
not104:
        cmp    al,112               ; F1?
        jnz    not112              ; jif no
        cmp    byte ptr mode,LINE  ; already in line mode?
        jz     doline              ; jif yes
        mov    byte ptr mode,LINE  ; set that mode
        call   movptr              ; move the *
        jmp    keyloop
doline:
        call   drawline             ; draw the line
getends:
        mov    ah,53                ; read cursor
        int    5fh
        mov    endpointx,bx         ; save new end point
        mov    endpointy,dx
        jmp    keyloop
not112:
        cmp    al,113               ; F2 ?
        jnz    not113              ; jif no
        cmp    byte ptr mode,POINT ; already in point mode?
        jz     dopoint              ; jif yes
        mov    byte ptr mode,POINT ; set that mode
        call   movptr              ; move the *
        jmp    keyloop
dopoint:
        mov    ah,53                ; read cursor
        int    5fh
        mov    ax,3201h              ; write pixel
```

```

        int      5fh
        jmp     keyloop
not113:
        cmp     al,114          ; F3?
        jnz     not114          ; jif no
        mov     byte ptr mode,DRAW ; set draw mode
        call    movptr           ; move the *
        jmp     keyloop
not114:
        cmp     al,116          ; F5?
        jnz     not115          ; jif no
        jmp     restart          ; re-init
not115:
        cmp     al,119          ; F8?
        jnz     unknown          ; jif no
exit:
        mov     ah,43            ; force alpha mode
        mov     al,0
        int     5fh
        mov     dx,offset exitesc
        mov     ah,9
        int     21h
        mov     ax,4c00h
        int     21h
unknown:
        mov     al,7dh            ; beep command
        mov     ah,1              ; write command to PPU
        mov     bx,27             ; PPU system services call
        int     50h
        jmp     keyloop
start  endp
;
setpen:
        mov     ah,53            ; read cursor
        int     5fh
        mov     ax,3401h          ; raise pen, then move
        int     5fh
        cmp     byte ptr mode,DRAW ; in draw mode?
        jnz     sp1              ; jif no
        mov     ah,50              ; do end point
        int     5fh
sp1:

```

```

        ret
;

init:
    mov     byte ptr mode,POINT      ; init mode to POINT plotting
    mov     dx,offset menu          ; address of escape sequences
    mov     ah,9                      ; output them to display
    int    21h
    mov     al,1
    mov     ah,43                     ; init graphics mode
    int    5fh
    mov     ah,54
    mov     al,0
    mov     bx,0
    mov     dx,0
    int    5fh
    mov     dx,offset labels        ; a blank
    mov     ah,9
    int    21h
    mov     bx,0                      ; for x=0 to
initloop:
    call   drawbox                 ; draw a box around key 1
    add    bx,9*6                  ; move to next box
    cmp    bx,4*9*6                ; skip center?
    jnz   boxjmp                  ; jif no
    add    bx,8*6+1                ; skip center
boxjmp:
    cmp    bx,470                  ; done?
    jb    initloop                ; jif no
    mov    ah,54                    ; move graphics cursor
    mov    al,1                      ; turn it on
    mov    bx,240                   ; x = 240
    mov    dx,100                   ; y = 100
    int    5fh
    mov    word ptr endpointx,240  ; init endpoints
    mov    word ptr endpointy,100
    mov    ah,47                     ; read graphics attributes
    int    5fh
    mov    dl,3                      ; set to complement mode
    mov    ah,48                     ; set graphics attributes
    int    5fh
    call  drawline
    ret

```

```

;
;

drawbox:
    push    bx          ; save x coordinate
    mov     dx,0          ; y
    mov     ax,3401h      ; set pen location
    int    5fh
    pop    bx          ; get x
    push    bx
    mov     dx,10         ; y=10
    mov     ax,34ffh      ; lower pen and move
    int    5fh
    pop    bx          ; get x
    push    bx
    add    bx,8*6+4      ; move to right of box
    mov     dx,10         ; top right corner
    mov     ax,3400h      ; draw
    int    5fh
    pop    bx          ; get x
    push    bx
    add    bx,8*6+4      ; move to right
    mov     dx,0          ; bottom right
    mov     ax,3400h      ; draw
    int    5fh
    pop    bx          ; get x
    push    bx
    mov     dx,0          ; bottom left
    mov     ax,3402h      ; draw
    int    5fh
    pop    bx          ; restore x
    ret
    page
drawline:
    mov     ah,53          ; read cursor position
    int    5fh
    mov     ax,3401h      ; raise pen and move
    int    5fh
    mov     bx,endpointx  ; to where cursor is
    mov     dx,endpointy  ; get other endpoint coordinates
    mov     ax,34ffh      ; lower pen and draw
    int    5fh
    ret

```

5

```
page
movptr:
    mov     ah,53          ; get cursor
    int     5fh
    mov     endpointx,bx    ; save it
    mov     endpointy,dx
    mov     ax,3600h         ; set cursor, off
    mov     bx,42            ; x for '*' in LINE box
    mov     dx,0              ; y
    int     5fh
    mov     dl,42            ; star
    mov     ah,2              ; print string
    cmp     byte ptr mode,LINE ; is it in LINE mode?
    jz      markline         ; jif yes
    mov     dl,32            ; blank

markline:
    int     21h
    mov     ax,3600h         ; set cursor, off
    mov     bx,96            ; x for '*' in POINT box
    mov     dx,0              ; y
    int     5fh
    mov     dl,42            ; star
    mov     ah,2              ; print string
    cmp     byte ptr mode,POINT ; is it in POINT mode?
    jz      markpoint        ; jif yes
    mov     dl,32            ; blank

markpoint:
    int     21h
    mov     ax,3600h         ; set cursor, off
    mov     bx,150            ; x for '*' in DRAW box
    mov     dx,0              ; y
    int     5fh
    mov     dl,42            ; star
    mov     ah,2              ; print char
    cmp     byte ptr mode,DRAW ; is it in DRAW mode?
    jz      markdraw          ; jif yes
    mov     dl,32            ; blank

markdraw:
    int     21h
    mov     ax,3601h          ; set cursor, on
    mov     bx,endpointx      ; get coordinates
    mov     dx,endpointy
```

```
int      5fh
ret
page
;
asterisk    db      '*'
blank        db      ' '
mode         db      '?'
endpointx   dw      ?
endpointy   dw      ?
;
menu         db      27,'&k0\',27,'&s1A',27,'&j@','$'
;
;
exitesc     db      27,'&s0A','$'
labels       db      ' Line '
              db      ' Point * '
              db      ' Draw '
              db      ' '
              db      ' Clear '
              db      ' '
              db      ' '
              db      ' Exit ','$'
;
;
cseg      ends
end      start
```



# 6

# Built-In Device Drivers

---

## 6.1 Introduction

In addition to the RAM and ROM disc drivers (A: and B:), the Portable PLUS includes device drivers for accessing the built-in devices (AUX, CLOCK, COM1, COM3, and CON) and various HP-IL peripheral devices (COM2, LPT1, LPT2, LST, PLT, PRN, and 82164A). These built-in device drivers are as follows:

- AUX** In the default configuration, equivalent to COM1. Redirectable from PAM.
- CLOCK** The system clock.
- COM1** The built-in serial (RS-232) port.
- COM2** Assigned to the first HP 82164A HP-IL/RS-232-C Interface in the loop (accessory ID 42h).
- COM3** The optional built-in modem.
- CON** The system console (the built-in display for output, the built-in keyboard for input).
- LPT1** Assigned to the first HP-IL printer (accessory ID 2xh) in the loop.
- LPT2** Assigned to the second HP-IL printer (accessory ID 2xh) in the loop.
- LST** Same as LPT2.
- PLT** Assigned to the first HP-IL plotter in the loop (accessory ID 6xh). Redirectable from PAM.

PRN      In the default configuration, equivalent to LPT1. Redirectable from PAM.

82164A    Same as COM2.

Standard MS-DOS device read and write operations will receive and send data through the standard drivers to external devices. The actual transmission of data is affected by several factors, including serial port configuration, software protocol, and hardware handshaking. Most of these factors can be controlled using I/O control commands to the desired device driver. The following paragraphs describe the operation of the data I/O channels of the modem and the serial port.



**Note**

When using MS-DOS Int 21h functions to read bytes from the serial port or modem devices, the device drivers will *not* wait for the specified number of characters to be received before returning to the application. For example, if an MS-DOS *read* command is issued to the COM1 with a character count of 10, and there are only eight characters currently in the buffer, only those eight characters will be returned to the calling program. The returned count will be 8 rather than 10. If a read command is issued when there are no characters in the buffer, the driver will return immediately with no characters read.

6

### 6.1.1 Serial Operation

Serial communications are subject to several types of hardware and software protocols, as well as data configurations. Three hardware handshake lines are used with external devices: Data Set Ready (DSR), Clear To Send (CTS), and Received Line Signal Detect (RLSD), also known as Device Carrier Detect (DCD). In addition, the driver can control two output lines: Data Terminal Ready (DTR) and Request To Send (RTS). Many applications don't use any of these lines. In its default state, the driver doesn't observe any hardware protocol and sets DTR and RTS true whenever the serial port is active.

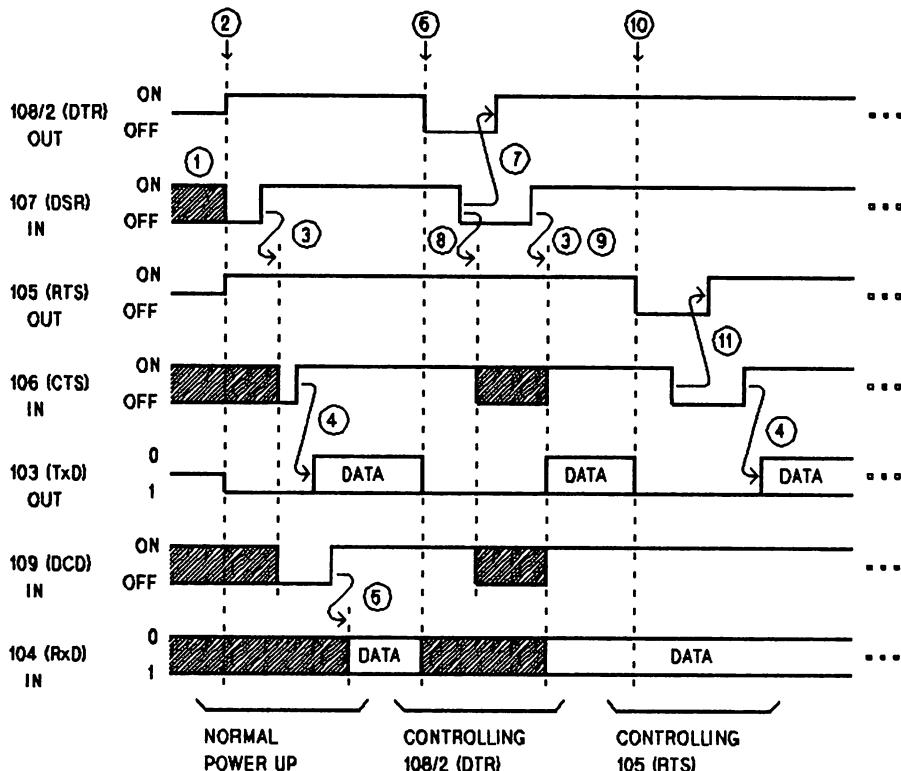
The driver can be set to observe any or all of the handshake lines through I/O control commands or through the PAM Datacom Configuration Menu. When handshaking is in effect, data transmission is governed by the RS-232-C standard:

- Data won't be sent out unless CTS, DSR, DTR, and RTS are true.
- Data won't be received unless RLSD, DSR, and DTR are true.

- If DSR becomes false before DTR is set false, the computer assumes the line is broken and will terminate the connection by setting its DTR and RTS outputs false. DTR is guaranteed to stay false for at least one second.
- If RLSD becomes false and remains false for more than 10 seconds while DTR is true, the computer assumes the line is broken and will terminate the connection by setting its DTR and RTS outputs false.
- After the DTR output has been set false, it won't be set true again until the DSR line is also set to false.
- After the RTS output has been set false, it won't be set true again until the CTS input line is also set to false.

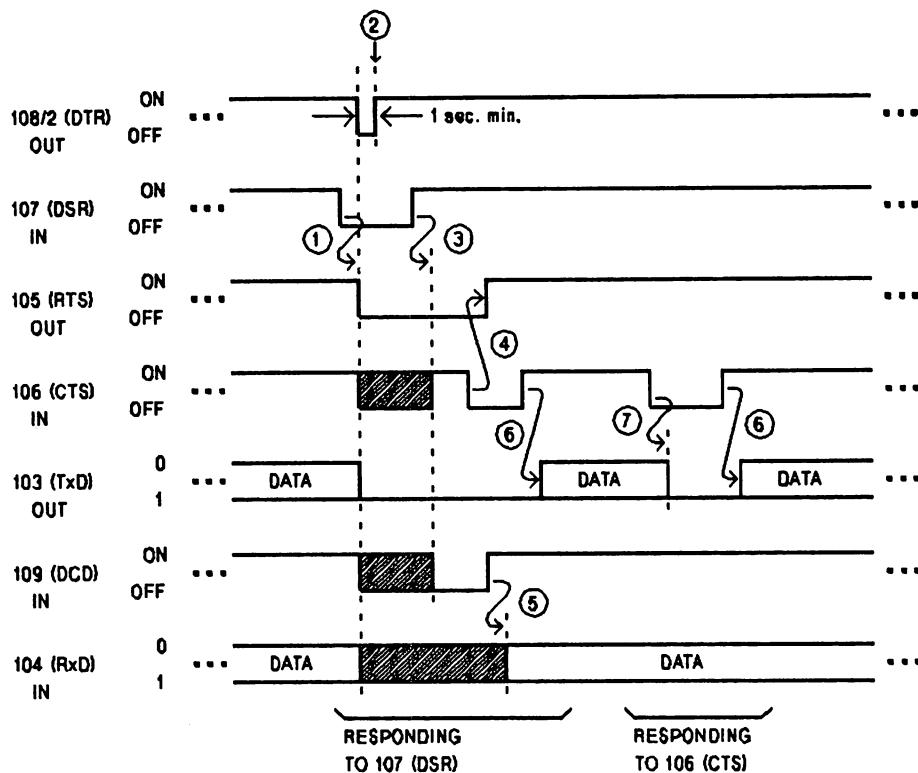
The following diagrams illustrate signal timing for typical situations on the serial interface.

**Figure 6-1. Serial Interface Timing - Sheet 1**



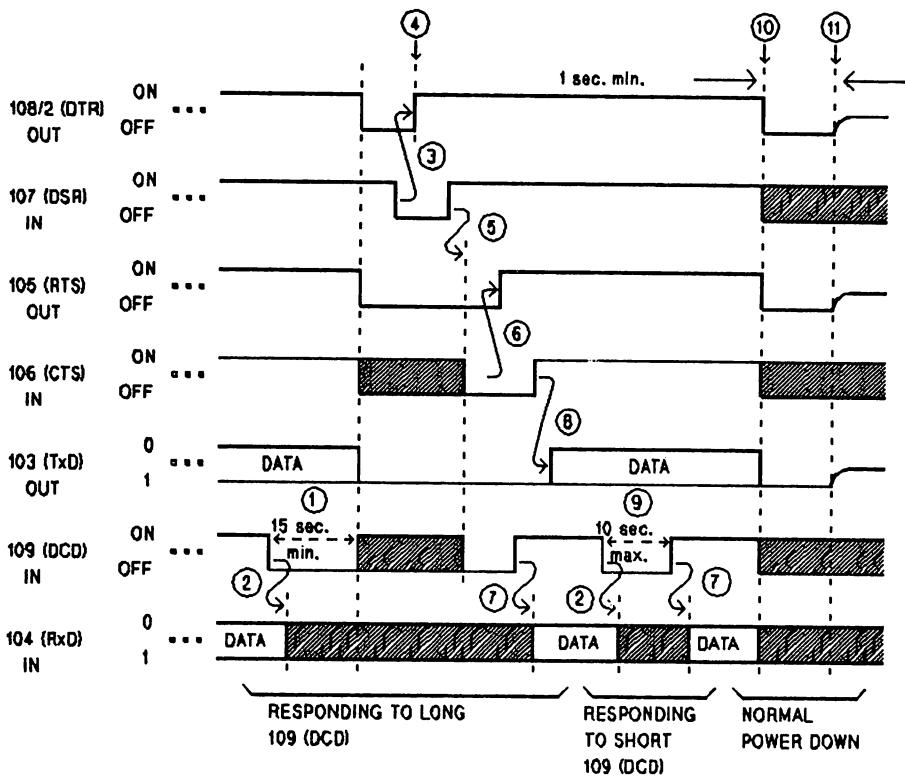
1. Shading indicates the computer ignores the signal.
2. Computer turns on interface power and initiates connection with DCE.
3. 107 (DSR) turning on ends "ignore" regions.
4. 106 (CTS) turning on enables data transmission.
5. 109 (DCD) turning on enables data reception.
6. Computer turns off 108/2 (DTR). This inhibits data transmission and reception.
7. 107 (DSR) turning off enables 108/2 (DTR) to turn on.
8. 107 (DSR) turning off starts "ignore" regions.
9. 107 (DSR) turning on enables data transmission and reception.
10. Computer turns off 105 (RTS). This inhibits data transmission only.
11. 106 (CTS) turning off enables 105 (RTS) to turn on.

**Figure 6-2. Serial Interface Timing – Sheet 2**



1. 107 (DSR) turning off while 108/2 (DTR) is on causes computer to end connection with DCE.
2. Computer initiates new connection with DCE.
3. 107 (DSR) turning on ends "ignore" regions.
4. 106 (CTS) turning off enables 105 (RTS) to turn on.
5. 109 (DCD) turning on enables data reception.
6. 106 (CTS) turning on enables data transmission.
7. 106 (CTS) turning off disables data transmission.

**Figure 6-3. Serial Interface Timing – Sheet 3**



1. 109 (DCD) turning off for greater than 15 seconds before 108/2 (DTR) is turned off causes computer to end connection with DCE.
2. 109 (DCD) turning off disables data reception.
3. 107 (DSR) turning off enables 108/2 (DTR) to turn on.
4. Computer initiates new connection with DCE.
5. 107 (DSR) turning off ends "ignore" regions.
6. 106 (CTS) already being off at end of "ignore" region enables 105 (RTS) to turn on.
7. 109 (DCD) turning on enables data reception.
8. 106 (CTS) turning on enables data transmission.
9. 109 (DCD) turning off for 10 seconds or less does not cause computer to end connection with DCE.
10. Computer ends connection with DCE.
11. Computer turns off interface power.

The serial driver also supports XON/XOFF protocol. This protocol is observed as the default state. In addition, the XON and XOFF characters can be redefined to any eight-bit values, which is valuable for some data services that don't use the default values of ^S (control-S, or XOFF) and ^Q (control-Q, or XON). The serial driver *does not* support ENQ/ACK protocol. If this protocol is desired, it must be implemented by the application.

In normal operation, power isn't applied to the serial port to reduce battery power consumption. Any character that is sent to the serial port (COM1) will automatically power up the interface. If data must be read from the port *before* any characters are transmitted, the port must be turned on in one of the following ways: via the PAM Datacom Configuration screen, the "M1," or "M2;" I/O Control commands, or by sending a character to the port through an MS-DOS Int 21h or Int 14h write function. The serial port remains powered until it is explicitly turned off or a reset occurs.

### 6.1.2 Modem Operation

The optional built-in modem is connected internally to a second serial port, but hardware handshaking is neither implemented nor necessary. The port configuration parameters (such as word length, stop bits, and baud rate) are set by the same I/O Control commands as the serial port. However, modem-specific Hayes-compatible commands are passed to the modem by the driver over the normal read/write channel. The modem is responsible for the separation of data and commands.

XON/XOFF protocol is implemented for the modem data channel. This protocol is observed unless it is disabled from PAM or by an application. ENQ/ACK protocol is *not* supported by the driver.

Like the serial port, the modem isn't usually turned on to conserve power. It is powered up by any character that is sent to it. However, an application must wait for one character transmission period after turning the modem on before sending it any data, or the data may be lost. This includes the "M0;" and "M4;" I/O Control commands, as well as MS-DOS Int 21h and Int 14h write functions. The character transmission period varies with the baud rate according to the following list:

Baud Rate	Character Transmission Period
1200	8.33 ms
300	33.33 ms
150	66.67 ms
110	90.90 ms

Once turned on, the modem remains powered until it is explicitly turned off or a reset occurs.

## 6.2 AUX, COM1, COM2, COM3, and 82164A Devices

The AUX device can address the serial port, the modem, or the HP 82164A HP-IL/RS-232-C Interface. The current device associated with AUX can be set by I/O control commands or by using the PAM System Configuration menu. At power-on or whenever the Edisc is formatted, the AUX device is pointed to the internal serial port. At any subsequent system reboot, the AUX device is associated with the physical device set by the PAM System Configuration menu.

The modem port and the serial port each have a built-in UART that supports the I/O control commands listed in table 6-1. Because follow-on products may use different hardware configurations (particularly for communications), software must use only the I/O Control calls for setting communication parameters. An advantage of using I/O control commands is that they will also work with the HP 82164A HP-IL/RS-232-C Interface (except for any command that begins with an "M".)

Note that all I/O control commands are terminated by a semicolon.



**Note**

In order to conserve battery power, any application that makes use of the modem or serial port should turn off the ports before terminating.

6

**Table 6-1. AUX I/O Control Commands**

Command	Description
<b>Break</b> B0; B1;	Stop sending break. Start sending break.  Sending the string "B1;" will cause the serial device to start sending a break. It will continue to do so until you send the string "B0;".

**Table 6-1. AUX I/O Control Commands (Continued)**

**Handshake Protocol**

- |     |                                     |
|-----|-------------------------------------|
| C0; | Disable XON/XOFF protocol.          |
| C2; | Enable XON/XOFF protocol (default). |
- The built-in modem and RS-232 understand only XON/XOFF protocol. "C0;" turns it off, and "C2;" turns it on.

**DTR/RTS**

- |      |                              |
|------|------------------------------|
| LI1; | Activate DTR line (default). |
| LI2; | Deactivate DTR line.         |
| LI3; | Activate RTS line (default). |
| LI4; | Deactivate RTS line.         |
- DTR and RTS are normally turned on when RS-232 is turned on; off when RS-232 is turned off.

**Modem/RS-232 Power Commands**

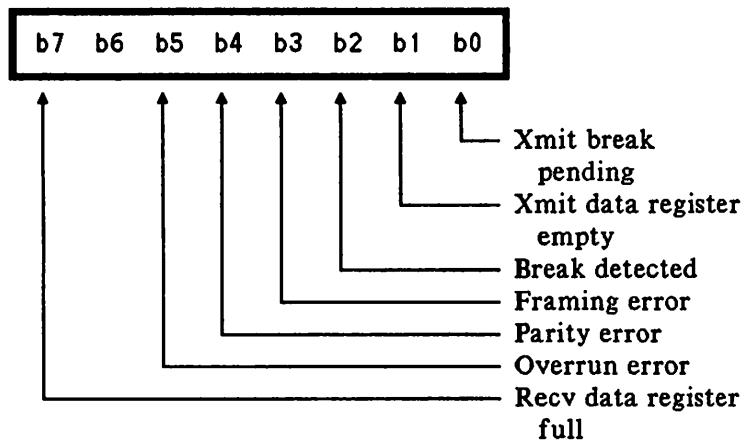
- |     |                       |
|-----|-----------------------|
| M0; | Modem on, RS-232 off. |
| M1; | RS-232 on, modem off. |
| M2; | RS-232 on.            |
| M3; | RS-232 off.           |
| M4; | Modem on.             |
| M5; | Modem off.            |
- These commands are not supported by the HP 82164A HP-IL/RS-232-C Interface. "M0;" and "M1;" switch between the modem and the RS-232 port. "M2;" through "M5;" control these ports individually. Whenever the RS-232 port is turned on, the DTR and RTS lines are set true; when this port is turned off, the lines are set false 1 second before the port is deactivated.

**Table 6-1. AUX I/O Control Commands (Continued)**

**UART Status**

M?;

If a UART status command is issued, the next I/O Control Read command for one character will return a byte of status which reflects the current state of the UART. The bits are as follows:

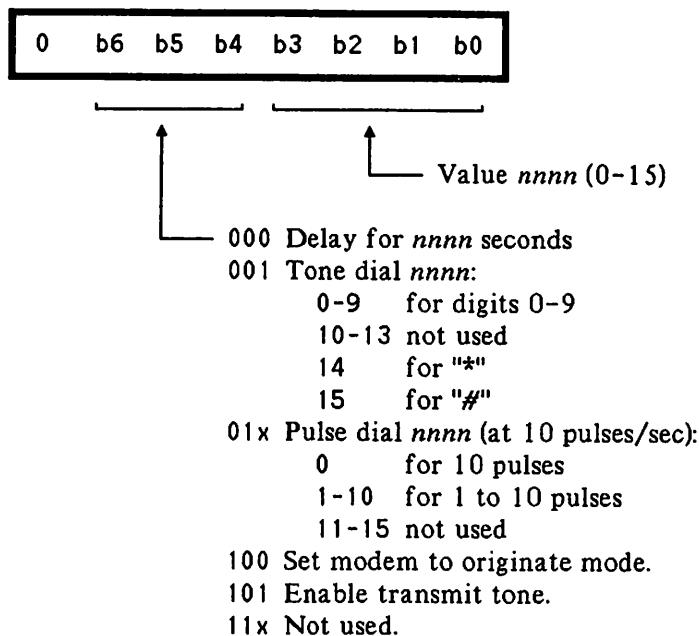


**Table 6-1. AUX I/O Control Commands (Continued)**

**HP110 Compatible Modem Dialing**

MD...;

HP110 compatible modem dial commands are terminated by any byte with the sign bit set. The intervening bytes have the following form:



This command works only when directed to the COM3 device, or to AUX when it is set to the Modem.

It is recommended that applications use the standard modem dialing command if HP110 compatibility is not needed (see chapter 12).

**Modem Hangup**

MH;

Forces the modem to hang up and turn off. This command works only for COM3.

**Table 6-1. AUX I/O Control Commands (Continued)**

**Modem Present Query**

MP; Checks for the optional built-in modem and returns one byte to the I/O control read buffer (to be read by an I/O Control Read command). The byte is zero if there's no modem, nonzero if there is a modem.

**Ring Enable**

MR; Turn on ring enable.  
MZ; Turn off ring enable.

It is possible to enable a software interrupt so that an application can detect the phone ringing. "MR," and "MZ," enable and disable this interrupt. (This function is equivalent to Int 59h).

**Device Status**

MS; If a device status command is issued, the next I/O control read command for one character will return one byte of status for the device the command was sent to (i.e "COM1" gives the serial port status). If the bit is "1" the condition is true.

- Bit 0: XON/XOFF handshake enabled.
- Bit 1: Unused.
- Bit 2: Driver has sent an XOFF.
- Bit 3: XOFF has been received from Host.
- Bit 4: RTS is true.
- Bit 5: DTR is true.
- Bit 6: No internal modem.
- Bit 7: Device is powered.

**Parity**

P0; Set even parity.  
P1; Set odd parity.  
P4; Set no parity.

**Table 6-1. AUX I/O Control Commands (Continued)**

**Set XON/XOFF Characters**

**PCwxyz;** Redefines the XON, XOFF, ENQ, and ACK handshake characters. (The ENQ and ACK characters are ignored by internal devices, but are used by the HP 82164A HP-IL/RS-232 interface.) *All four characters must be included in the command.*

w: New XON character.  
x: New XOFF character.  
y: New ENQ character.  
z: New ACK character.

**Baud Rate**

**SB3;** Set baud to 110.  
**SB4;** Set baud to 134.  
**SB5;** Set baud to 150.  
**SB6;** Set baud to 300.  
**SB7;** Set baud to 600.  
**SB8;** Set baud to 1200 (default for modem).  
**SB9;** Set baud to 1800.  
**SBA;** Set baud to 2400.  
**SBB;** Set baud to 3600.  
**SBC;** Set baud to 4800.  
**SBD;** Set baud to 7200.  
**SBE;** Set baud to 9600 (default for serial port and HP82164).  
**SBF;** Set baud to 19200.

**Table 6-1. AUX I/O Control Commands (Continued)**

<b>Hardware Handshake</b>	
SL0;	Observe all handshake.
SL1;	Ignore CTS.
SL2;	Observe CTS.
SL3;	Ignore RLSD.
SL4;	Observe RLSD.
SL5;	Ignore DSR.
SL6;	Observe DSR.
SL7;	Ignore all handshake (default).
If hardware RLSD is enabled, and if RLSD is not high when the AUX write is attempted, the driver will wait for up to 10 seconds for the line to go high. If it does not, RTS and DTR will be turned off and the driver will return with a "not ready" error.	
If CTS or DSR is being observed, and if one (or both) is not true when the AUX write is attempted, the driver will return a "write fault" error after 10 seconds of retries.	
<b>Stop Bits</b>	
SS0;	Use 1 stop bit.
SS1;	Use 2 stop bits.
<b>Word Length</b>	
SW0;	Word length is 8 bits.
SW1;	Word length is 7 bits.
<b>Check Buffer</b>	
(BFh);	This is a one-character command with a value of 0BFh ("?" with the sign bit set). It returns the number of characters available in the device's buffer (zero if none). To use this, write the command to AUX I/O Control, then read one byte from AUX I/O Control. You can use this returned byte as a character count when performing an MS-DOS read from the AUX device. This call is supported on all HP-IL character device drivers.

**Example:** This example demonstrates how to talk to the serial port and how to configure it using IOCTL calls (DOS function 44h). In this example the configuration is hard-wired into a DB statement, but could be made modifiable with more complex programming. This program opens the AUX and CON devices and puts them in "raw" mode. It then alternates between:

- Doing a status of the AUX port, and if anything has been entered, copying it to the CON device. (If data logging is enabled, it will also get copied to the PRN device.)
- Doing a status of the CON device, and if any keys have been pressed, copying them to the AUX device.

To allow the "data-logging" status to be toggled, the softkey **F1** is defined and the softkey menu is displayed.

This program is structured such that it can be run through the EXE2BIN utility. It also assumes that this has been done, hence, it doesn't set up the DS, SS, or ES registers--if this is a .COM file, those registers will already have been set equal to CS by the system.

```
page 60,132
title GLASS TELETYPE --- terminal emulation example
;
cseg    segment para public 'code'
        assume cs:cseg, ds:cseg
        org     100h
start   proc   far
        call    openfiles           ; open the AUX and CON as a files
        call    cnfaux              ; configure AUX port
loop1:
        call    stataux             ; check AUX & if anything is available
        ;                                echo it to display
        ; now check for a key hit, and if there is one, process it.
        ;
        call    statkey              ; check keyboard & if alpha-numeric
        ;                                ; send out AUX, else handle it.
        jmp    loop1                ; loop forever
;
start   endp
;
; fetch a key from the keyboard. If alpha-numeric simply send it out the
; AUX port. If it's F1, toggle log-bottom. If F8, exit the program.
;
```

```

statkey:
    mov     dl,0ffh          ; set flag for console INPUT
    mov     ah,6              ; DOS function - direct console I/O
    int     21h              ; see DOS function calls
    jz      nokey            ; jif no key
    cmp     al,27             ; possible extended keycode?
    jnz    alphakey          ; jif no
    mov     dl,0ffh          ; get the extended keycode
    mov     ah,6
    int     21h
    jz      esckey            ; jif no 2nd keycode, just escape key
    cmp     al,112             ; 'p' (F1)?
    jnz    noti12            ; jif no
    xor     byte ptr logflag,255 ; toggle log flag
    mov     dx,offset logoff   ; address of 'log off' escape seq
    cmp     byte ptr logflag,0 ; is log off?
    jz      logit             ; jif yes
    mov     dx,offset logon   ; else use 'log on' escape seq

logit:
    mov     ah,9              ; send to console
    int     21h
    ret

noti12:
    cmp     al,119             ; 'w' (F8)?
    jnz    beepit            ; jif no
    mov     ax,4c00h          ; DOS function - terminate program
    int     21h              ; (function call 4C - error level=0)
;
; the 4C function call never returns to us (it terminates us).
;
beepit:
    mov     dl,7              ; bell character
    mov     ah,2              ; send it to "CON"
    int     21h
    ret

esckey:
    mov     al,27             ; reload escape keycode

alphakey:
    mov     tempbuff,al        ; store keycode in memory
    mov     dx,offset tempbuff ; get address of buffer
    mov     cx,1               ; number of bytes to send
    mov     bx,auxhandl        ; get file handle for AUX

```

```

        mov     ah,40h           ; write to file or device
        int     21h

nokey:
        ret
;

; a file handle in MS-DOS is just a 16 bit number that the opsys uses to refer
; to an open file or device. Kind of like Secret Agent Man, they've given you
; a number, and they've taken away your name.....
;

; this routine opens the AUX port and the CON device and puts them into
; raw mode for faster output and so CRs don't get stripped by the AUX driver's
; input routines. It also opens the PRN device in case log bottom
; gets turned on.
;

openfiles:
        mov     ah,3dh           ; DOS function - open file or device
        mov     al,2               ; for read/write
        mov     dx,offset auxname ; point to name of AUX driver
        int     21h
        mov     auxhandl,ax       ; save the file handle
        mov     bx,ax
        mov     ax,4400h          ; read AUX device info function
        int     21h
        mov     bx,auxhandl       ; get aux handle
        mov     dh,0               ; upper byte has to be zero
        or      dl,20h             ; set RAW mode bit
        mov     ax,4401h          ; write device info
        int     21h
        mov     ax,3d02h           ; open file function
        mov     dx,offset connname ; point to name of CON driver
        int     21h
        mov     conhandl,ax       ; save console handle
        mov     bx,ax
        mov     ax,4400h          ; read CON device info function
        int     21h
        mov     bx,conhandl       ; get console handle
        mov     dh,0               ; upper byte has to be zero when write
        or      dl,20h             ; force to raw mode
        mov     ax,4401h          ; write CON device info function
        int     21h
        mov     dx,offset prnname ; point to name of PRN driver
        mov     ax,3d01h           ; open for writing

```

```

        int      21h
        mov      prnhand1,ax           ; save file handle
        mov      dx,offset menu       ; initialize softkey menu
        mov      ah,9                  ; send escape strings to CON
        int      21h
        mov      dx,offset logoff     ; and DATA LOGGING key
        mov      ah,9
        int      21h
        mov      byte ptr logflag,0    ; init DATA LOGGING flag
        ret

;
;

; This piece of code assumes that the AUX port has previously been opened as
; a file, and its file 'handle' has been saved in the RAM location called
; 'auxhand1'.
;

cnfaux:
        mov      bx,auxhand1          ; get the file handle
        mov      dx,offset iostring    ; address of I/O control string
        mov      cx,32                 ; length of I/O control string
        mov      ah,44h                ; DOS function - I/O control
        mov      al,3                  ; write to control channel
        int      21h                  ; call the DOS function
        ret

;
;

; The two lines above that loaded registers AH and AL were done as seperate
; loads for clarity, but could have been written more compactly as:
;

;      mov      ax,4403h
;

; for this example, we'll do a status of the AUX driver to find out if there's
; anything in the input buffer. If there is, we'll read it and echo the bytes
; to the console display (for lack of anything better to do with them).
;

stataux:
        mov      bx,auxhand1          ; get AUX handle
        mov      dx,offset statstring  ; address of ioctl string
        mov      cx,1                  ; only one byte long
        mov      ax,4403h              ; write to dev control channel
        int      21h                  ; do it
        mov      bx,auxhand1          ; handle
        mov      dx,offset tempbuff    ; address of buffer

```

```

        mov    cx,1           ; read only one byte
        mov    ax,4402h        ; read from dev control channel
        int    21h             ; do it
        mov    al,tempbuff     ; get the byte we read
        cmp    al,0             ; anything available?
        jz    saret            ; jif no, return
        mov    bx,auxhandl     ; get handle
        mov    cl,al            ; get count
        mov    ch,0              ; where to read it into
        mov    ah,3fh            ; read from file or device
        int    21h

;

; when reading from a device driver, MS-DOS stops when it encounters a CR.
; hence, we must use the count returned in AX by function 3F as the actual
; number of bytes read.

;

        push   ax              ; save number bytes read
        mov    cx,ax            ; copy actual number read
        mov    dx,offset tempbuff ; get address of data
        mov    bx,conhandl      ; get console handle
        mov    ah,40h            ; DOS function - write to file
        int    21h
        pop    cx              ; recover # bytes read
        cmp    byte ptr logflag,0 ; logging data?
        jz    saret            ; jif no
        mov    dx,offset tempbuff ; address of data
        mov    bx,prnhandl      ; PRN handle
        mov    ah,40h            ; write to file/device
        int    21h

saretn:
        ret

;

logon      db      27,"&f0a1k16d2L Data *Logging ",27,112,"$"
logoff     db      27,"&f0a1k16d2L Data Logging ",27,112,"$"
menu       db      27,"&f0a2k-1d0L"
                db      27,"&f0a3k-1d0L"
                db      27,"&f0a4k-1d0L"
                db      27,"&f0a5k-1d0L"
                db      27,"&f0a6k-1d0L"
                db      27,"&f0a7k-1d0L"
                db      27,"&f0a8k16d2LExit to P.A.M. ",27,119,"$"

```

```
logflag      db      ?
auxhandl     dw      ?
auxname      db      "AUX",0
statstring   db      0bfh
conhandl     dw      ?
conname      db      "CON",0
prnhandl    dw      ?
prnname      db      "PRN",0
;
;
;
iostring      db      "M1;SBE;C2;P0;SW1;SL7;LIO;SS0;MZ;"
;
; This IOCTL call would cause the following to occur:
;
;      M1;      select RS-232 port as active port, disable modem port
;      SBE;      set baud rate to 9600
;      C2;      enable XON/XOFF protocol
;      P0;      set even parity
;      SW1;      set word length to 8 bits
;      SL7;      ignore all hardware handshake (CTS, DSR)
;      LIO;      disable signal lines (DTR)
;      SS0;      set stop bits to 1
;      MZ;      disable ring interrupt
;
tempbuff      db      128 dup (?)
;
cseg      ends
end      start
```

## 6.3 NUL Device

Anything sent to the NUL device will disappear.

---

## 6.4 CLOCK Device

The CLOCK device defines and performs functions like any other character device. When a read or write to this device occurs, exactly six bytes are transferred. The first two bytes are a word representing the number of days since January 1, 1980. The third byte is minutes; the fourth, hours; the fifth, hundredths of seconds; and the sixth, seconds. Reading the CLOCK device returns the date and time; writing to it sets the date and time. I/O Control reads and writes can be used in the same way to read and set alarms. However, you should note that *it is very expensive timewise to read the hardware clock.* Continuous clock reading can degrade system performance.

There are system services available for reading and setting the clock and alarms (refer to "System Services Interrupt 50h" in chapter 5).

---

## 6.5 LPT1, LPT2, LST, PLT, and PRN Devices

The following device drivers allow access to peripherals on the HP-IL interface loop.

- 6      LPT1      Assigned to the first HP-IL printer (accessory ID 2xh) in the loop. If no printer is found, a search is made for a display device (accessory ID 3xh). If neither is found, an error is returned.
- LPT2      Assigned to the second HP-IL printer (accessory ID 2xh) in the loop. If only one printer (or none) is found, a search is made for a display device (accessory ID 3xh). If no display device is found, an error is returned.
- LST      Same as LPT2.
- PLT      Assigned to the first HP-IL plotter in the loop (accessory ID 6xh). Redirectable from PAM.
- PRN      In the default configuration, equivalent to LPT1. Redirectable from PAM.

## 6.6 CONsole Driver

The CONsole driver controls the system console, which consists of two main parts and the corresponding control mechanisms:

- Built-in display for output.

Terminal driver (refer to "Control Sequences" below).

Fast video interrupt 5Fh (refer to "Fast Video Interrupt" in chapter 5).

Video I/O interrupt 10h (refer to "Video I/O Interrupt" in chapter 5).

CON expansion interrupt 5Eh (refer to "CON Expansion Interrupt" in chapter 5).

- Built-in keyboard for input.

Keyboard driver (refer to "Keyboard Operation" at the end of this chapter).

Keyboard I/O interrupt 16h (refer to "Keyboard I/O Interrupt" in chapter 5).

The following overview describes some major aspects of the CONsole driver's operation.

**Display Modes.** In Alpha mode, the display shows 25 rows of 80 characters. In Graphics mode, the display shows 200 rows of 480 pixels. The CONsole driver can control the displayed output in each of these display modes.

**Character Fonts.** Within Alpha display mode there are two character fonts: the HP font and the Alternate font. Only one character font can be displayed at one time. The HP font contains the Roman8 character set (bold and light), a math character set, and a line-drawing character set. The Alternate font contains the Alternate character set (bold and light). Any combination of the current mode's fonts can appear on the screen at the same time. Font and mode selection can be performed via escape sequences sent to the display.

**Keyboard Driver Modes.** The keycodes generated and the actions performed by the CONsole keyboard driver depend upon the driver mode (HP mode or Alternate mode) and the keyboard mode (normal, Scancode mode, or Modifier mode). In Alternate mode, the driver attempts to emulate an IBM-compatible keyboard. In HP mode, the driver provides HP110 compatibility, plus the addition of the various keyboard modes. The keyboard also provides several terminal-like features:

- Programmable function keys.
- Menu display.
- Transmit functions mode.

### 6.6.1 CONsole Control Sequences

The CONsole driver recognizes certain control characters and escape sequences as commands to perform certain functions. Generally, these characters and sequences are interpreted as they are sent to the CONsole display (the LCD); simply typing control characters and escape sequences on the keyboard will usually do nothing special unless the corresponding key codes are being echoed to the display.

**Control Characters.** Table 6-2 lists the control characters handled by the CON driver. The remaining control character that are not in the table are generally ignored (and not displayed) by the driver.

Table 6-2. Control Characters

Control Character	Description
Null ^@	This character is ignored and not printed.
Bell ^G	If the bell is enabled, an audible beep will be issued; otherwise, the character is ignored and not printed.
Backspace (BS) ^H	The cursor backspaces one column.
Tab Forward (HT) ^I	The cursor moves forward to the next horizontal tab stop. If there are no more tab stops on the current line, the cursor moves to the first column of the next line. Tab stops are set at eight-column intervals.

**Table 6-2. Control Characters (Continued)**

<b>Line Feed (LF)</b>	
<b>^J</b>	The cursor moves down to the same column in the next line. If it is initially on the last screen line, a Roll Up is performed.
<b>Carriage Return (CR)</b>	
<b>^M</b>	The cursor moves to the first column of the current line.
<b>Select Alternate Font (SO)</b>	
<b>^N</b>	Subsequent characters appear in the alternate font, starting at the cursor position and continuing until an SI (Shift In) character is received.
<b>Select Primary Font (SI)</b>	
<b>^O</b>	Subsequent characters appear in the primary font, starting at the cursor position and continuing until an SO (Shift Out) character is received.
<b>Escape (ESC)</b>	
<b>^[</b>	The escape character starts an escape sequence.
<b>Start Output (DC1/XON)</b>	
<b>^Q</b>	Resumes data transmission. If typed on the keyboard, the flow of characters to the display may be resumed (depending on the application).
<b>Stop Output (DC3/XOFF)</b>	
<b>^S</b>	Suspends data transmission. If typed on the keyboard, the flow of characters to the display may be suspended (depending on the application).

**HP Two-Character Escape Sequences.** The CONsole driver recognizes the HP two-character escape sequences described in table 6-3.

**Table 6-3. HP Two-Character Escape Sequences**

Escape Sequence	Description
<b>Print Screen</b> ESC 0	Dumps the current screen contents to the printer. Same effect as pressing <b>(Print)</b> or issuing Interrupt 53h.
<b>Cursor Up</b> ESC A	The cursor moves up one line. If the cursor is at the top of the screen, a Roll Down occurs. If the cursor is at the top of display memory, nothing happens.
<b>Cursor Down</b> ESC B	The cursor moves down one line. If the cursor is at the bottom of the screen, a Roll Up occurs. If the cursor is at the bottom of display memory, nothing happens.
<b>Cursor Right</b> ESC C	The cursor moves right one column, stopping when it reaches the right side of the screen.
<b>Cursor Left</b> ESC D	The cursor moves left one column, stopping when it reaches the left side of the screen.
<b>Reset Terminal</b> ESC E	The CONsole output driver is reset as follows: <ul style="list-style-type: none"><li>■ Alpha mode.</li><li>■ Default fonts.</li><li>■ Clear display.</li><li>■ Cursor at home position.</li><li>■ Reset softkeys to default values.</li><li>■ Latin print direction (left to right).</li></ul>

**Table 6-3. HP Two-Character Escape Sequences (Continued)**

<b>Home Down</b> ESC F	The cursor moves to the first column of the blank line just below the last line of text in display memory. If the last line of text is also the last line of display memory, everything moves up one line, discarding the first line of display memory, to provide a blank line at the end.
<b>Home Up</b> ESC H	The cursor moves to the first column of the first line in display memory.
<b>Tab Forward</b> ESC I	The cursor moves forward to the next horizontal tab stop. If there are no more tab stops on the current line, the cursor moves to the first column of the next line. Tab stops are set at fixed eight-column intervals. (Same function as ^I (HT).)
<b>Clear to End of Display Memory</b> ESC J	Blanks everything from the cursor to the end of display memory.
<b>Clear to End of Line</b> ESC K	Blanks characters from the cursor to the end of the current line.
<b>Insert Line</b> ESC L	Inserts a blank line above the line that the cursor is on. The cursor moves to the first column of the inserted line.
<b>Delete Line</b> ESC M	Deletes the line that the cursor is on. The cursor moves to the first column of the next line.
<b>Delete Character</b> ESC P	Deletes the character at the cursor, moving the remainder of the current line to the left (or right, if in Arabic mode).

**Table 6-3. HP Two-Character Escape Sequences (Continued)**

<b>Insert Character On</b>	
ESC Q	Enables insert character mode. Subsequent characters are inserted just before the character that the cursor is on, pushing the remainder of the current line to the right (or left, if in Arabic mode).
<b>Insert Character Off</b>	
ESC R	Disables insert character mode.
<b>Roll Up</b>	
ESC S	The screen moves ahead one line in display memory (text moves up one line).
<b>Roll Down</b>	
ESC T	The screen moves back one line in display memory (text moves down one line).
<b>Next Page</b>	
ESC U	Displays the next "page" of display memory. (A "page" is either 23 or 25 lines long, depending on whether or not softkey labels are displayed.) The cursor moves to the top left corner of the new page.
<b>Previous Page</b>	
ESC V	Displays the previous page of display memory. The cursor moves to the top left corner of the new page.
<b>Display Functions On</b>	
ESC Y	Enables display functions mode. All subsequent control codes and escape sequences are printed and not executed, with these exceptions: <ul style="list-style-type: none"><li>■ ESC Z - Displayed, then executed.</li><li>■ Carriage return - Displayed, then executed with a line feed.</li></ul>
<b>Display Functions Off</b>	
ESC Z	Turns off display functions mode.

Table 6-3. HP Two-Character Escape Sequences (Continued)

**Read Primary Status**

ESC ^

Returns the following 10-byte sequence ("\*" indicates that a bit can be "0" or "1"):

Byte 0: ESC.

Byte 1: \.

Byte 2: Display memory size (always 34h):

0011 0100— 1K Bytes.

  └— 2K Bytes.

  └— 4K Bytes.

  └— 8K Bytes.

Byte 3: Configuration strap A-D status:

0011 0\*0\*— A - Function Key Transmission.

  └— B - Space Overwrite.

  └— C - Inhibit EOL Wrap.

  └— D - Page/Line.

Byte 4: Configuration strap E-H status (always 3Ch):

0011 1100— E - Always "0".

  └— F - Always "0".

  └— G - DC2 Handshake.

  └— H - Inhibit DC2.

Byte 5: Latching key status:

0011 0\*0\*— Caps Lock Key.

  └— Block Mode Key.

  └— Auto LF Mode.

  └— Secondary Status.

Byte 6: Transfer pending flags (always 30h):

0011 0000— Cursor Sense Pending.

  └— Function Key Pending.

  └— Enter Key Pending.

  └— Secondary Status Pending.

Byte 7: Error flags (always 32h):

0011 0010— Datacomm Error.

  └— Self-Test OK.

  └— Device Error.

Byte 8: Device transfer pending flags (always 30h):

0011 0000— Device Status Pending.

  └— Operation Status Pending.

Byte 9: CR.

**Table 6-3. HP Two-Character Escape Sequences (Continued)**

**Read Relative Cursor Position**

ESC `

Returns the following 11-byte sequence:

ESC & a ccc x rrr Y

where *ccc* is the cursor column (000-079), and *rrr* is the cursor row (000-024, relative to the top of the screen). In Arabic mode, the columns are numbered from right to left.

**Read Absolute Cursor Position**

ESC a

Returns the following 11-byte sequence:

ESC & a ccc c rrr R

where *ccc* is the cursor column (000-079), and *rrr* is the cursor row (000-061, absolute line in display memory). In Arabic mode, the columns are numbered from right to left.

**Enter Line**

ESC d

Causes the line containing the cursor to be returned.

**Home Up**

ESC h

The cursor moves to the first column of the first line in display memory. (Same as ESC H.)

**Tab Backward**

ESC i

Moves the cursor back to the previous tab stop. If the cursor is initially at the start of the line, it moves to the last tab stop on the previous line. Tab stops are set at fixed eight-column intervals.

**HP Alpha Escape Sequences.** The CONsole driver uses the alpha escape sequences listed in table 6-4 to control the alpha display. In sequences requiring numeric parameters, each parameter is optional--if it's omitted, it defaults to zero.

Almost every escape sequence listed below ends with an uppercase character. The uppercase character terminates the sequence. If the corresponding lowercase character is used, commands that start with the same three characters can be concatenated. For example,

ESC & a 3 r 19 C

would move the cursor to absolute row 3, column 19. The escape sequences that affect softkey definitions (ESC & f) are often used this way.



The term "HP Alpha Escape Sequences" refers to those sequences that begin with the two characters "ESC &"; the name does not imply that all such sequences necessarily perform an "alpha" function.

**Table 6-4. HP Alpha Escape Sequences**

Escape Sequence	Description
<b>Move Cursor to Absolute Column</b> ESC & a {0...79} C	If the target column is beyond the display bounds, the cursor moves as far as possible. Note that columns are numbered from right to left in Arabic mode.
<b>Move Cursor to Absolute Row</b> ESC & a {0...61} R	Zero specifies the top row in display memory, while 61 specifies the bottom row. If the specified row is beyond display memory bounds, the cursor is moved as far as possible. The screen will be scrolled as much as necessary to keep the cursor visible.
<b>Move Cursor to Screen Column</b> ESC & a {0...79} X	If the target column is beyond the display bounds, the cursor moves as far as possible. Note that columns are numbered from right to left in Arabic mode.

Table 6-4. HP Alpha Escape Sequences (Continued)

**Move Cursor to Screen Row**

ESC & a {0...22/24} Y

Zero specifies the top row on the screen, while 24 specifies the bottom row (22 if softkey labels are being displayed). If the specified row is beyond the screen bounds, the cursor is moved as far as possible.

**Move Cursor to Relative Column**

ESC & a {± value} C

The cursor is moved the specified number of columns from its current position. If the target column is beyond display bounds, the cursor is moved as far as possible. Note that "+" moves left and "-" moves right in Arabic mode.

**Move Cursor to Relative Row**

ESC & a {± value} R

The cursor is moved the specified number of lines from its current position. If the target row is beyond display memory bounds, the cursor is moved as far as possible. The screen will be scrolled as much as necessary to keep the cursor visible.

**Reverse Print (Arabic Mode)**

ESC & a 1 D

Print subsequent text from right to left on the display. All horizontal movement functions are reversed, placing column zero at the right side of the screen and increasing to the left. Additionally, absolute and relative cursor positioning column coordinates are reversed, running from 0 to 79, *right to left*.

**Forward Print (Latin Mode)**

ESC & a 2 D

Print subsequent text from left to right on the display. All horizontal movement functions are standard, placing column zero at the left side of the screen and increasing to the right.

Table 6-4. HP Alpha Escape Sequences (Continued)

**Select Display Enhancement**

ESC & d {@...0}

The required parameter selects the enhancement to be applied to subsequent characters. The enhancement is field oriented, staying in effect until another enhancement is given. The halfbright enhancement is implemented by using the character set in font region 2 (light).

Enhancement	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
-------------	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

Blinking	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
Inverse	**		**		**		**		**		**		**		**
Underline		***											***		***
Halfbright													***	***	***

**Softkey Definition Key Select**

ESC & f {1...8} K

Specifies which key will be affected by this softkey definition escape sequence.

1: **(f1)** softkey.

.

.

8: **(f8)** softkey.

6

**Softkey Definition Attribute Select**

ESC & f {0...2} A

Specifies the attribute for the key being defined in this definition escape sequence.

0: *Normal* softkey. String will be treated just as though it was typed at the keyboard. This is the default attribute.

1: *Local* softkey. String will be displayed, but not entered into the keyboard buffer.

2: *Transmit* softkey. String will be treated as though it was typed at the keyboard with an appended carriage return.

Table 6-4. HP Alpha Escape Sequences (Continued)

**Softkey Definition String Length**

ESC & f value L

Specifies the number of characters in the string for the key being defined in this escape sequence.

-1: The old string is erased.

0: The old string is left unchanged.

>0: The specified number of characters are read into the string (following the end of this escape sequence, after softkey label characters, if any).

**Softkey Definition Label Length**

ESC & f value D

Specifies the number of characters in the label for the key being defined in this escape sequence.

-1: The old label erased.

0: The old label is left unchanged.

>0: The specified number of characters are read into the label (following the end of this escape sequence).

**Softkey Labels Off**

ESC & j @

If softkey labels are currently on, turn them off. In graphics mode, nothing happens.

**Softkey Labels On**

ESC & j B

If softkey labels are currently off, turn them on. In graphics mode, nothing happens.

**Clear/Set Alt Mode Keyboard**

ESC & k 0 \

Clear Alt Mode Keyboard (set HP Mode).

ESC & k 1 \

Set Alt Mode Keyboard (clear HP Mode).

**Clear/Set Auto LF**

ESC & k 0 A

Turn off auto line feed.

ESC & k 1 A

Turn on auto line feed.

**Clear/Set Bell Enable**

ESC & k 0 D

Disable the ^G bell.

ESC & k 1 D

Enable the ^G bell.

**Table 6-4. HP Alpha Escape Sequences (Continued)**

**Clear/Set Keyboard Modes**

**ESC & k {0...3} 0**

Selects keyboard modes.

- 0: Turn off numeric pad, keycode, and Modifier modes.
- 1: Turn on numeric keypad mode.
- 2: Turn on keycode mode.
- 3: Turn on Modifier mode.

**Clear/Set Caps Lock**

**ESC & k {0/1} P**

Permits the caps lock key to be programmatically turned on and off.

- 0: Turn off caps lock.
- 1: Turn on caps lock.

**Clear/Set Transmit Functions (Strap A)**

**ESC & s {0/1} A**

Enables or disables the transmission of key codes for local function keys.

- 0: Turn off Transmit Functions.
- 1: Turn on Transmit Functions.

**Clear/Set Inhibit End-of-Line Wrap (Strap C)**

**ESC & s {0/1} C**

Sets end-of-line wrap.

- 0: Enable end-of-line wrap. Printing a character in column 80 causes an implicit carriage return and line feed to occur.
- 1: Disable end-of-line wrap. The cursor will "stick" in column 80. Subsequent characters will overwrite the character in that column.

**HP Graphics Escape Sequences.** The CONsole driver uses the HP escape sequences listed in table 6-5 to control the graphics display. In sequences requiring numeric parameters, each parameter is optional--if a parameter is omitted, it defaults to zero.



**Note** The term "HP Graphics Escape Sequences" refers to those sequences that begin with the two characters "ESC \*"; the name does not imply that all such sequences necessarily perform a "graphic" function.

**Table 6-5. HP Graphics Escape Sequences**

Escape Sequence	Description
<b>Clear Graphics Memory</b> ESC * d A	Clear all pixels in the display if in graphics mode.
<b>Set Graphics Memory</b> ESC * d B	Set all pixels in the display if in graphics mode.
<b>Graphics On</b> ESC * d C	Switch to graphics mode (if not already there) and turn on the graphics display, making visible existing graphics screen data.
<b>Graphics Off</b> ESC * d D	Turn off the graphics display. Screen data is retained. This sequence is ignored in alpha mode.
<b>Alpha On</b> ESC * d E	Switch to alpha mode (if not already there) and turn on the alpha display, making visible existing alpha screen data.
<b>Alpha Off</b> ESC * d F	Turn off the alpha display. Screen data is retained. This sequence is ignored in graphics mode.

**Table 6-5. HP Graphics Escape Sequences (Continued)**

**Alpha Cursor Block/Graphics Cursor On**

**ESC \* d K**

In alpha mode, set the primary cursor to "block" and turn it on. In graphics mode, turn on the graphics cursor.

**Alpha Cursor Underline/Graphics Cursor Off**

**ESC \* d L**

In alpha mode, set the primary cursor to "underline" and turn it on. In graphics mode, turn off the graphics cursor.

**Position Graphics Cursor (Absolute)**

**ESC \* d x y 0**

Move the graphics cursor to the specified absolute coordinates ( $0 \leq x \leq 479$ ,  $0 \leq y \leq 199$ ).

**Position Graphics Cursor (Relative)**

**ESC \* d x y P**

Move the graphics cursor the specified distance from its current position. (The cursor will not move past the edges of the display.)

**Alpha Cursor On (Primary/Secondary)**

**ESC \* d value Q**

Turn on the alpha cursor.

0: Turn on primary cursor (default).

1: Turn on secondary cursor.

**Alpha Cursor Off**

**ESC \* d R**

Turn off the cursor.

**Set Drawing Mode**

**ESC \* m value A**

Selects the way in which dots, lines, and area patterns are written to the graphics display.

0: No change.

1: Clear (turn off graphic bits).

2: Set (turn on graphic bits).

3: Complement (toggle graphic bits).

4: Jam (turn bits on or off according to data).

Table 6-5. HP Graphics Escape Sequences (Continued)

**Set Line Type**

**ESC \* m value B**

Selects one of 10 line types. Once selected, all subsequent vectors will be drawn using that line type.

- 1: ■■■■■■■ Solid line (default).
- 2: User-defined line pattern.
- 3: Same pattern as 2.
- 4: ■■■□□□□
- 5: ■■■■■□□□
- 6: ■■■■■□□□□
- 7: ■□□□□□□□
- 8: ■■■■■□□□
- 9: ■□□□□□□□
- 10: ■■■□□□□□

**Define Line Pattern and Scale**

**ESC \* m pat scl C**

Defines the dot pattern used to draw vectors of line type 2. The first parameter is in decimal form. The optional second parameter specifies a scale factor that will be applied to the pattern of all subsequently drawn lines. (Each pixel is repeated *scl* times.) If the scale is not specified, it defaults to 1.

6

**Set Relocatable Origin**

**ESC \* m x y J**

Moves the relocatable origin to the specified absolute coordinates.

**Set Relocatable Origin to Pen Position**

**ESC \* m K**

Moves the relocatable origin to the current pen position.

**Set Relocatable Origin to Cursor Position**

**ESC \* m L**

Moves the relocatable origin to the current graphics cursor position.

**Table 6-5. HP Graphics Escape Sequences (Continued)**

**Set Graphics Defaults**

**ESC \* m R**

Selects the default (reset) values for the following graphics parameters:

- Graphics Cursor Off.
- Graphics Display On.
- Relocatable Origin at (0,0).
- "Set Pixels" Drawing Mode.
- Linetype 1 (solid).
- Pen Up.
- Pen to (0,0).
- Graphics Cursor to (0,0).
- User Line Pattern 1.
- Line Scale Factor 1.

**Lift Pen**

**ESC \* p A**

Raises the graphics pen so that subsequent pen movement will not draw lines.

**Lower Pen**

**ESC \* p B**

Lowers the graphics pen so that subsequent pen movement will draw lines.

**Move Pen to Graphics Cursor Position**

**ESC \* p C**

If the pen is down, a line will be drawn as determined by the current drawing mode and line type.

**Draw Single Dot at Current Pen Position**

**ESC \* p D**

The dot will be drawn in accordance with the current drawing mode (set, clear, complement, no change).

**Set Relocatable Origin to Current Pen Position**

**ESC \* p E**

Sets the relocatable origin to the current pen position.  
Relocatable Format parameters will be referenced to this point.

**Use ASCII Absolute Format**

**ESC \* p F**

States that subsequent X and Y parameters are absolute graphics display coordinates. (Default.)

**Table 6-5. HP Graphics Escape Sequences (Continued)**

**Use ASCII Incremental Format**

**ESC \* p G** States that subsequent X and Y parameters are relative to the current pen position.

**Use ASCII Relocatable Format**

**ESC \* p H** States that subsequent X and Y parameters are relative to the current position of the relocatable origin.

**NOP/Synch**

**ESC \* p Z** This command is useful as a graphics sequence terminator. Its only function is to clear the parameter buffer; it can be used to end long pen movement sequences (for example, "ESC \* p a 0,0 5,5 20,10 Z").

**Move Pen**

**ESC \* p x y ...** Upon receipt of any command or parameter delimiter (a space or comma), if two valid parameters have been specified the pen is moved (drawing a line if it is currently lowered) to the appropriate coordinates (determined from the parameters, the current pen position, and the current drawing format).

**Read Device ID**

**ESC \* s value ^** Returns a device specific string into the keyqueue, followed by a carriage return. *This is among the "graphics" sequences because it is typically used, in HP terminals, to return various types of graphics status. Only a minimal subset is included in this computer.*

0: Returns model number and CR.

1: Returns model number and CR.

110: Returns serial number and CR.

**ANSI Escape Sequences.** The CONsole driver uses the ANSI escape sequences listed in table 6-6 to control the display. All ANSI escape sequences begin with "ESC [", followed by one or more optional parameters, and end with an alpha character (usually). Multiple parameters are separated by semicolons (";"). Unspecified or zero parameters are usually treated as "1".

**Table 6-6. ANSI Escape Sequences**

Escape Sequence	Description
<b>Cursor Up n Lines (CUU)</b> ESC [ <i>p1</i> A	Moves the cursor up the specified number of rows; the column remains unchanged. The cursor will not move past display memory bounds. The screen will scroll as much as necessary to keep the cursor visible. The default parameter is 1.
<b>Cursor Down n Lines (CUD)</b> ESC [ <i>p1</i> B	Same as CUU, but the cursor moves down.
<b>Cursor Right n Columns (CUR)</b> ESC [ <i>p1</i> C	Moves the cursor right the specified number of columns; the row remains unchanged. The cursor will not move past the screen bounds. The default parameter is 1.
<b>Cursor Left n Columns (CUL)</b> ESC [ <i>p1</i> D	Same as CUR, but the cursor moves left.
<b>Move Cursor to Absolute Position (CUP)</b> ESC [ <i>p1</i> ; <i>p2</i> H	Moves the cursor to the position specified by the two parameters. The first one specifies the row, while the second specifies the column. <i>p1</i> : Target line number (1...62). <i>p2</i> : Target column number (1...80).
<b>Erase Display (ED)</b> ESC [ 0 J	Clear to end of display memory (default).
ESC [ 1 J	Clear to beginning of display memory.
ESC [ 2 J	Clear entire display memory.

**Table 6-6. ANSI Escape Sequences (Continued)**

**Erase Line (EL)**

- |           |                                 |
|-----------|---------------------------------|
| ESC [ 0 K | Clear to end of line (default). |
| ESC [ 1 K | Clear to beginning of line.     |
| ESC [ 2 K | Clear entire line.              |

**Insert n Lines (IL)**

- |                   |   |
|-------------------|---|
| ESC [ <i>p1</i> L | Inserts one or more blank rows into the row containing the cursor by shifting the contents of the current row and all following rows the specified number of lines. Lines at the end of display memory will be lost. The cursor is placed in the first column of the first newly inserted blank line. |
|-------------------|---|

**Delete n Lines (DL)**

- |                   |  |
|-------------------|--|
| ESC [ <i>p1</i> M | The specified number of lines will be deleted, beginning the with line containing the cursor. Subsequent lines will be moved up to fill the void, scrolling in blank lines at the end of display memory. The cursor is placed in the first column of the first line that moves up into the void. |
|-------------------|--|

**Delete n Characters (DCH)**

- |                   |  |
|-------------------|--|
| ESC [ <i>p1</i> P | Deletes one or more characters from the cursor position onward to the right (or left, if in Arabic mode). As characters are deleted, the remainder of the line shifts as necessary to fill the void. Blanks shift in at the end of the line. |
|-------------------|--|

**Cursor Position Report (CPR)**

- |                               |   |
|-------------------------------|---|
| ESC [ <i>p1</i> ; <i>p2</i> R | This sequence is the proper response to a Device Status Report (DSR) sequence. (Refer to "ESC [ 6 n" below.)<br><i>p1</i> : Cursor row.<br><i>p2</i> : Cursor column. |
|-------------------------------|---|

**Next Page/n Pages (NP)**

- |                   |   |
|-------------------|---|
| ESC [ <i>p1</i> U | Move ahead the specified number of "pages" in display memory. Normally, a page contains 25 lines; a page is only 23 lines long, however, when softkey labels are displayed. |
|-------------------|---|

**Table 6-6. ANSI Escape Sequences (Continued)**

**Previous Page/n Pages (PP)**

**ESC [ p1 V** Same as CNP, but moves back the specified number of pages.

**Move Cursor to Absolute Position (HVP)**

**ESC [ p1;p2 f** Same as CUP.

**Set Mode**

**ESC [ p1 h** Set display mode. Multiple parameters may be included if they're separated by semicolons (";").

- 4: Insert Character On (IRM).
- =0: Alpha Mode On. (*IBM: Set 40x25 b&w*)
- =1: Alpha Mode On. (*IBM: Set 40x25 color*)
- =2: Alpha Mode On. (*IBM: Set 80x25 b&w*)
- =3: Alpha Mode On. (*IBM: Set 80x25 color*)
- =4: Graphics Mode On. (*IBM: Set 320x200 color*)
- =5: Graphics Mode On. (*IBM: Set 320x200 b&w*)
- =6: Graphics Mode On. (*IBM: Set 640x200 b&w*)
- =7: End-of-Line Wrap On.
- =8: Alpha Mode On.
- =10: Graphics Mode On.
- ?7: End-of-Line Wrap On.

**Reset Mode**

**ESC [ p1 l** Reset display mode. Multiple parameters may be included if they're separated by semicolons (";").

- 4: Insert Character Off (IRM).
- =0: Alpha Mode Off. (*IBM: Reset 40x25 b&w*)
- =1: Alpha Mode Off. (*IBM: Reset 40x25 color*)
- =2: Alpha Mode Off. (*IBM: Reset 80x25 b&w*)
- =3: Alpha Mode Off. (*IBM: Reset 80x25 color*)
- =4: Graphics Mode Off. (*IBM: Reset 320x200 color*)
- =5: Graphics Mode Off. (*IBM: Reset 320x200 b&w*)
- =6: Graphics Mode Off. (*IBM: Reset 640x200 b&w*)
- =7: End-Of-Line Wrap Off.
- =8: Alpha Mode Off.
- =10: Graphics Mode Off.
- ?7: End-Of-Line Wrap Off.

**Table 6-6. ANSI Escape Sequences (Continued)**

**Set Graphics Rendition (SGR)**

**ESC [ p1 m**

Specifies the attributes to be used with subsequent output characters. Multiple parameters may be specified if they're separated by semicolons (";").

- 0: All attributes Off.
- 1: Half bright On.
- 4: Underline On.
- 5: Blinking On.
- 7: Inverse On.
- 10: Use HP fonts.
- 11: Use Alt fonts.

**Device Status Report (DSR)**

**ESC [ 6 n**

Returns a Cursor Position Report (CPR) sequence for the current cursor position. (Refer to "ESC [ p1;p2 R" above.)

**Save Cursor Position (SCP)**

**ESC [ s**

The current cursor row and column is saved.

**Restore Cursor Position (RCP)**

**ESC [ u**

Moves the cursor to the row and column that was previously saved with the SCP sequence.

## 6.6.2 Keyboard Operation

The CONsole driver normally recognizes each keystroke and generates one or more codes that it places in the keyqueue. The driver has two modes that determine how it processes keystrokes:

- HP mode (usually associated with the Roman8 character font). Provides these features.
  - "Local function" keys for display control.
  - Programmable function keys.
  - Softkey menu display.
  - Scancode mode.
  - Modifier mode.
  - Transmit functions mode.
- Alternate mode (usually associated with the Alternate character font). Provides compatibility with IBM operation.

Notice that although driver modes are frequently associated with corresponding character fonts, the treatment of keystrokes and display characters are two entirely separate *unrelated* functions. It is perfectly fine to operate the keyboard in Alt mode using HP fonts.

The Alternate mode keyboard is enabled by the HP Alpha sequence "ESC &k1\". The Alternate font is enabled by the ANSI sequence "ESC [11m".

The CONsole driver mode is controlled by the "ESC &kn\" escape sequence and the "H/h" I/O control functions.

**Function Keys.** Each function key (**f1**) through (**f8**) has default keycodes in each of the driver modes (refer to chapter 13). In HP mode, a definition assigned to a function key is generated when that key is pressed, regardless of whether or not the menu is displayed. In Alt mode, the function keys are assigned IBM-compatible keycodes; (**f9**) and (**f10**) are represented by the (Menu) and (User/System) keys, respectively.

Definitions and menu labels are assigned by "ESC & f" escape sequences.

**Keys Affected by Transmit Functions Mode.** Transmit Functions mode is available only in HP mode. When turned off, most of the top row of keys on the keyboard can be used to perform local screen editing. When turned on, these same keys instead add to the keyqueue the escape sequences that correspond to their local functions (the **A** key, for example, would add "ESC A" to the keyqueue rather than actually move the cursor up one line). The following list summarizes the keys that are affected by Transmit Functions mode.

Keys	Operation
	Cursor Left
	Cursor Up
	Cursor Down
	Cursor Right
f3	Clear Line
f4	Clear Display
f5	Insert Line
f6	Delete Line
f7	Insert Character
f8	Delete Character
▲	Home Up
▲	Previous Screen
▼	Next Screen
▶	Home Down
	Enter

**Keys That Generate Interrupts.** Several keys generate software interrupts, which are described in chapter 5. **Menu** and **User/System** have an effect only in HP mode.

**Menu** Turns the function key labels on and off by issuing Interrupt 56h. (Int 56h can be programmatically called to toggle function key labels on and off as well; similarly, taking over Interrupt 56h will cause **Menu** key presses to enter your own interrupt handler.)

**User** Typically used to select user mode function keys. Sets bit 2 of register AH and generates Interrupt 57h.

**System** Typically used to select system mode function keys. Clears bit 2 of register AH and generates Interrupt 57h.

**Print** Copies the current display to the PRN device. Generates Interrupt 53h.

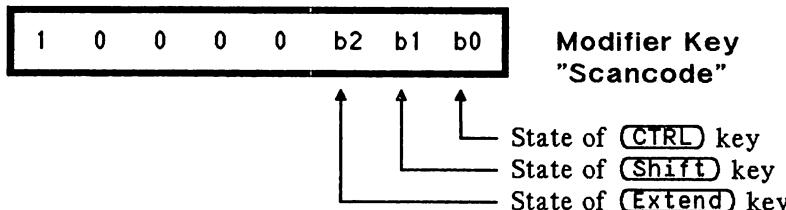
**Keyboard Modes.** The CONsole driver defines four keyboard modes that determine how keystrokes are interpreted and placed in the keyqueue. The four modes are described in the following paragraphs.

- Normal (character code) mode.
- Scancode mode.
- Numeric keypad mode.
- Modifier mode.

**Character Code Mode.** While the keyboard is in character code mode, each keystroke generates one or more character codes that are added to the keyqueue. The character codes depend upon the CONsole driver mode: HP mode or Alternate mode. In addition, normal "character" keys generate character codes that depend upon the keyboard language--the mapping provided by the localized configuration EPROM. This is the normal operating mode for the keyboard. (Refer to chapter 13 for keyboard mapping and character codes.)

**Scancode Mode.** When in scancode mode, pressing a key adds that key's scancode (rather than character code) into the keyqueue. A scancode is simply a number from 0 to 71 that refers to a key's physical position on the keyboard. Nearly every key on the keyboard occupies a unique position in the keyboard matrix and can be sensed in terms of its matrix position; the exceptions are the modifier keys (**Extend**), (**CTRL**), and (**Shift**) and the contrast key (◎).

Contrived scancodes for the modifier keys are constructed and added to the key queue each time any modifier key makes a downward *or upward* transition; the resultant code is 80h plus, in the low three bits, the final states of the three modifier keys after the transition is finished.



The pseudo-scancodes 80h-87h (128-135) can therefore be generated by the modifier keys. Figure 6-4 shows the keyboard and indicates the scancodes associated with each key.

**Figure 6-4. Keyboard Scancodes**

1 f1	0 f2	9 f3	8 f4	17 Menu	16 Syst	24 f5	33 f6	32 f7	41 f8	40 Sel	48 ◀	57 ▲	56 ▼	65 ▶
2 '	10 1	18 2	19 3	25 4	26 5	34 6	42 7	43 8	49 9	50 0	58 -	59 =	66 Back space	
3 Tab	12 Q	11 W	20 E	28 R	27 T	36 Y	35 U	44 I	52 O	60 P	51 [	68 ]	67 \	
4 Caps	CTRL	5 A	13 S	22 D	21 F	30 G	29 H	37 J	45 K	53 L	61 ;	69 ‘	70 Return	
15 ESC	Shift	6 Z	14 X	23 C	31 V	38 B	46 N	47 M	55 .‘	54 .‘	62 /	Shift	71 Stop	
7 Ent	Ext char	39									Ext char	71 Stop		

Scancodes are shown in the upper left corner of the keys.

Scancodes not used: 63, 64.



In Scancode mode, the **Tab** key generates the code 03h, which appears to be ^C to the operating system. This can cause the program to terminate during printing or file operations.

**Numeric Keypad Mode.** The keyboard of the Portable PLUS includes an embedded numeric keypad (indicated by blue numbers and symbols on the front faces of the keycaps). The numeric keypad is activated by pressing **(Extend) (Select)**, and can be used in both HP and Alt modes. When turned on, the embedded numeric keyboard generates *only* the indicated numeric pad keycode, regardless of whether or not any modifier keys are depressed (with one Alt mode exception, which will be discussed shortly). Pressing **(Extend) (Select)** a second time turns off the numeric pad, and the keyboard reverts to its normal format.

In Alt mode, the numeric keypad can be used to generate *any* keycode: with the numeric keypad turned on, hold down the **(Extend)** key and type, on the numeric pad, the *decimal value* of the keycode you wish to generate. When you release the **(Extend)** key, the keycode you have specified will be added to the keyqueue.

The numeric keypad will not function in Scancode mode.

**Modifier Mode.** Modifier Mode combines certain features of Scancode Mode and normal keyboard mode. Generally, the keyboard functions as it normally does in Character Code mode. Modifier keys, however, add pseudo-scancode values to the keyqueue on their upward and downward transitions, as they do in Scancode mode. Certain local function keys add special one-byte values to the keyqueue and generally *do not* perform their usual function (the **(Caps)** key is an exception). Also, Transmit Functions mode is always active, but the initial escape character normally generated by any key that issues a two-character escape sequence will be added to the keyqueue with the sign bit set (9Bh rather than 1Bh).

### 6.6.3 CONsole I/O Control Functions

The CONsole driver provides several I/O control functions. Refer to the *MS-DOS Programmer's Reference Manual* for information about using these functions. Table 6-7 lists the write control functions.

**Table 6-7. CONsole Write I/O Control Functions**

Function	Description
<b>Request Keyqueue Lookahead</b> ?	Requests that a subsequent CON read control function return the next character from the keyqueue. (Refer to the next table for details.)
<b>Enable HP Mode (Disable Alt Mode)</b> H	Puts CONsole driver in HP mode.
<b>Enable Alt Mode (Disable HP Mode)</b> h	Puts CONsole driver in Alt mode.
<b>Request Console Mode Information</b> M	Requests that subsequent CON read I/O control return information about the current state of the CONsole driver. (Refer to the next table for information.)
<b>Enable Numeric Keypad</b> N	Turns on the numeric keypad.
<b>Reset Keyqueue</b> Q	Flushes the type-ahead buffer.
<b>Enable Modifier Mode</b> X	Turns on Modifier mode. In this mode, every key on the keyboard adds a one- or two-character keycode to the keyqueue. Modifier keys ( <b>CTRL</b> ), ( <b>Shift</b> ), and ( <b>Extend</b> ) add keycodes at both their upward and downward transitions.

**Table 6-7. CONsole Write I/O Control Functions (Continued)**

<b>Disable Special Modes</b>	
Y	Cancels all of the special keyboard modes: Numeric Keypad mode, Modifier mode, and Scancode mode.
<b>Enable Scancode Mode</b>	
Z	Turns on Scancode mode. In this mode, every matrix key adds its scancode, rather than normally processed keycode, into the keyqueue. Modifier keys generate pseudo-scancodes on both their upward and downward transitions. Note that both left and right shift keys have the same scancode.

Table 6-8 lists the read control functions. The function request character is specified by the most recent write control function call (refer to the previous table).

**Table 6-8. CONsole Read I/O Control Functions**

Function	Description
<b>Request Keyqueue Lookahead</b>	
?	Returns the next byte in the keyqueue. If the keyqueue is empty, -1 is returned. If the I/O control call sets CX to a value greater than one, the same next byte (or -1) will be returned the number of times specified by CX.

**Table 6-8. CONsole Read I/O Control Functions (Continued)**

**Request Console Mode Information**

M

Returns up to four bytes of current CONsole driver status information. CX specifies the number of bytes to read; fewer than four can be read, but if more than four are requested, only the first four are meaningful. The four status bytes are:

**Byte 0**

- Bit 0: Alpha cursor is on.
- Bit 1: End-of-line wrap enabled.
- Bit 2: Softkeys are being displayed.
- Bit 3: Insert Character mode is on.
- Bit 4: CapsLock is on.
- Bit 5: Transmit Functions is on.
- Bit 6: HP mode is active.
- Bit 7: Modifier mode is active.

**Byte 1**

- Bit 0: Scancode mode is active.
- Bit 1: Softkey label capture enabled.
- Bit 2: Softkey string capture enabled.
- Bit 3: Auto Linefeed is on.
- Bit 4: Bell is enabled.
- Bit 5: Graphics mode is on.
- Bit 6: Output is stopped.
- Bit 7: Primary cursor is "block".

**Byte 2**

- Bit 0: Numeric pad is active.
- Bit 1: Graphics cursor is on.
- Bit 2: Graphics pen is raised.
- Bit 3: (Reserved.)
- Bit 4: Numeric keypad is generating a keycode.
- Bits 5-7: (Reserved.)

**Byte 3**

- Bit 0: Blinking enhancement enabled.
- Bit 1: Inverse enhancement enabled.
- Bit 2: Underline enhancement enabled.
- Bit 3: Halfbright enhancement enabled.

# Low-Level Hardware Interface

## 7.1 Introduction

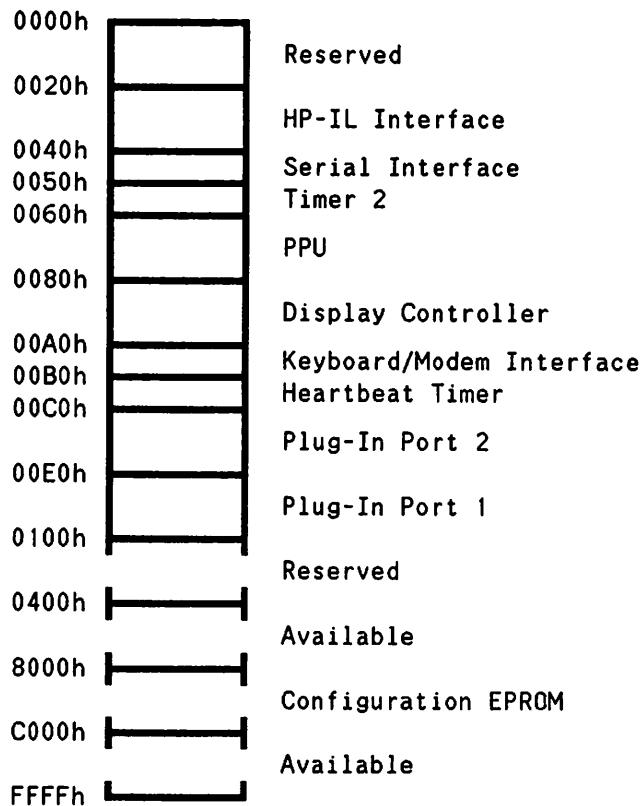
For applications that don't require compatibility with other computers, certain operations may be performed using a low-level hardware interface. That is, the application may directly control the individual circuits performing the operations, rather than using "standard" system functions. This chapter describes the low-level hardware interface for these circuits:

- Multi-controllers (keyboard, serial and modem interfaces, timers).
- HP-IL controller.
- Display controller.

## 7.2 I/O Memory Map

The CPU has a 1M-byte system memory address space and a 64K-byte I/O address space. The I/O address space contains the read and write registers that control the hardware circuits. Figure 7-1 illustrates the I/O address space. High bytes have odd addresses; low bytes have even addresses.

Figure 7-1. I/O Address Space



Detailed information about the registers used by each circuit is included in the following sections.

## 7.3 Multi-Controllers

The computer contains *two* 1LK5 multi-controller ICs: one associated with the modem interface and keyboard, and one associated with the serial interface.

Each multi-controller has the following capabilities:

- Keyboard interface.
- Interval timer.
- Serial port (for built-in serial interface or optional modem).
- Multi-purpose port.

All of these functions share a common eight-bit data bus. Timing is provided by an external clock signal. The keyboard interface and multi-purpose port can generate or propagate an interrupt while the system is in sleep mode (with the external clock turned off).

### 7.3.1 Keyboard Interface

The system keyboard interface is provided by only one of the multi-controllers. (This capability in the other multi-controller isn't used.) It serves two purposes:

- Generating an interrupt if enabled whenever the state of the keyboard changes (whenever a key is depressed or released). Key debounce is provided by delaying the interrupt by an appropriate time period.
- Latching the state of the keyboard so that the row and column information can be read from the data bus. Function-modifier keys (Shift, CTRL, and Extend char), which are intended to be depressed while other keys are pressed, have a separate input port with a single input line for each key.

When a key is depressed or released, row and column comparators detect the change in the state of the keyboard lines. When this occurs, the keyboard interface starts a timer to give the line enough time to settle (debounce time).

After the debounce time has elapsed, the interface circuitry senses the row and column comparators to check if the key change is still present. If the key change is no longer present at the end of the debounce time, the interface circuitry resets the debounce timer and continues sampling the keyboard lines. If a key change is still present at the end of the debounce time, an interrupt is generated. At the same time, the state of the keyboard is latched, and sampling of the keyboard is discontinued. The interface circuitry remains in this state until the interrupt is cleared.

One bit in a status register indicates whether a matrix key generated the interrupt. If this bit is set when the register is read, the state of the row and column buffers must then be read in order to identify the depressed key.

The process for detecting a change of state of a nonmatrix key is similar to the process just described, except that the key that caused the interrupt is immediately identified by a bit in the status register (reading from the buffer is unnecessary). The states of matrix and nonmatrix keys are latched at the end of the debounce time, regardless of whether a matrix or nonmatrix key initiated the interrupt process.

Two registers are used to disable or clear interrupt requests from the keyboard or the system interrupt lines. When an interrupt is cleared, the interrupt source is then able to generate another interrupt immediately following this bus cycle. In addition, when an interrupt is cleared, the latch associated with the corresponding comparator is updated, so that the comparator is set-up to detect the next change of state.

The debounce timer is automatically reset whenever the system is in sleep mode and at the time when the last pending keyboard interrupt is cleared. If a key is pressed while the machine is in sleep mode, an interrupt is immediately generated without any debounce delay. In this situation, no status bit is set until the system wakes up. Then the debounce timer begins counting, and at the end of the debounce time another interrupt is generated and the appropriate status bit is set.

### 7.3.2 Interval Timer

Each of the two multi-controllers provides an interval timer. Each interval timer is a 24-bit binary counter with a resolution of 2.25 microseconds and a range of 37 seconds. The timers can generate automatically repeated interrupts at a software-defined interval.

Timer 1 provides the system "heartbeat" timer (18 interrupts per second). These interrupts generate the timer tick (interrupt 1Ch), and control key repetition and timeout. This timer is devoted to the BIOS.

Timer 2 controls the "Return to Command" sequence timing for the optional modem whenever the modem is turned on.

### 7.3.3 Serial Port

Each multi-controller provides a serial port (UART): one for the built-in serial interface, and one for use with the optional modem. Each serial port contains a parallel-to-serial transmitter and an independent serial-to-parallel receiver. These are double buffered, programmable ports with variable baud rates, word lengths, stop bit lengths, and parities. The receivers enable the CPU to detect the incoming baud rate.

Data transmission and reception may be interrupt driven or accomplished by means of software polling. The transmitter can be programmed to send either CPU data or breaks (all zero transmission).

The multi-controller enables testing of transmitter and receiver logic at operating speed. This is accomplished by internally routing transmitted data to the receiver serial input line and allowing the CPU to access the receive shift register.

The serial port consists of a transmitter, a receiver, and associated control and status registers. Data transmission is asynchronous with respect to data reception.

**Transmitter Operation.** When the system is awake, the transmitter is idle if no break is pending and if no data is waiting to be sent out.

Break transmission is initiated by setting a bit in the control register. Breaks are continuously transmitted as long as the bit is set. Note that a break transmission has precedence over a data transmission.

Data transmission is initiated by writing to the transmit data register. This action clears the "transmit data register empty" flag. The register contents are first loaded into the transmit shift register. Then, the empty flag is set, signaling that the transmit register is empty. If enabled, the transmitter interrupt is asserted. New data can now be written to the transmit data register. Data transmission proceeds. When the stop bits have finished, the transmitter returns to the idle state.

**Receiver Operation.** The receiver input data stream has two sources, which are controlled by the loopback option. When loopback is inactive, the serial input is gated

to the receiver's input data line. When loopback is active, the transmitter's output is selected as the receiver's input.

The state of the receiver input data line can be read from a bit in the status register. The CPU can sample this bit to determine the incoming baud rate.

The receiver is idle as the system wakes up. It remains in this state until a valid start bit is detected. When this occurs, the receiver enters its receive mode and shifts in the programmed number of data bits. If parity is enabled, the parity bit is sampled next. The receiver then samples the first stop bit.

After the incoming frame has been processed, the receiver is ready to update its status and data register. If "receive data register full" flag is clear, the receiver assumes the old register contents and status have been read, and it loads the new data word. The parity error, framing error, and break detect flags are set accordingly, and the overflow error is cleared. The full flag is then set, and (if enabled) the receiver interrupt is also asserted.

If full flag was initially set, only the overrun error flag is set. The receive register and the remaining status bits remain unchanged. The new data word and its status information are discarded. To clear the "receive data register full" flag, the CPU simply writes a "1" to bit 1 of the control register (0048h).

**Initializing the Serial Port.** The receiver and transmitter enter their idle states as the system wakes up. The proper baud rate, word and stop bit length, and parity selection must be programmed before data can be sent or received reliably.

The transmitter remains idle until a break is forced or until the "transmit data" register is loaded.

Because the multi-controller places the serial port in loopback mode when the system is reset, the state of the receive line has no effect on the receiver until loopback is deactivated. This allows the control registers to be initialized to the proper settings before the receiver enters the receive mode.

### 7.3.4 Multi-Purpose Port

Each multi-controller provides a multi-purpose port, which consists of four interrupt lines. Operation of the multi-purpose interrupt lines is asynchronous and is unaffected by the system clock or the state of the keyboard interface processing. These interrupt lines are negative edge-triggered, to trigger on either the rising or falling edge of the interrupt input line.

### 7.3.5 Registers - Overview

Each multi-controller contains 16 internal registers. These registers exist in I/O memory at even addresses, low byte only. Table 7-1 summarizes their functions.

**Table 7-1. Multi-Controller Registers**

I/O Address	Function
0040h	Interrupt status
0042h	Interrupt control
0044h	Write: Serial baud select (low byte)
0046h	Write: Serial baud select (high byte)
0048h	Serial control/status
004Ah	Serial data
004Ch	Serial interrupt control
004Eh	Read: Serial/Timer 2 interrupt status (Reserved)
0050h	Timer 2 reference (high byte)
0054h	Timer 2 reference (middle byte)
0056h	Timer 2 reference (low byte)
0058h	Timer 2 control
005Ah	Timer 2 counter (high byte)
005Ch	Timer 2 counter (middle byte)
005Eh	Timer 2 counter (low byte)
00A0h	Keyboard/multi-purpose interrupt status
00A2h	Keyboard/multi-purpose interrupt control
00A4h	Read: Keyboard column latch
00A6h	Write: Modem baud select (low byte) Read: Keyboard row latch
00A8h	Write: Modem baud select (high byte) Modem control/status
00AAh	Modem data
00ACh	Modem interrupt control
00AEh	Read: Modem/Heartbeat Timer interrupt status Write: Keyboard debounce disable (Reserved)
00B0h	Heartbeat timer reference (high byte)
00B2h	Heartbeat timer reference (middle byte)
00B4h	Heartbeat timer reference (low byte)
00B6h	Heartbeat timer control
00B8h	Heartbeat timer counter (high byte)
00BAh	Heartbeat timer counter (middle byte)
00BCh	Heartbeat timer counter (low byte)
00BEh	

The following paragraphs describe the operation of each register. Registers are grouped functionally: keyboard function, serial/modem function, and timer function. Where appropriate, registers are discussed in pairs--one from each of the two multi-controllers.

### 7.3.6 Registers - Keyboard Function

The keyboard interface has five address locations associated with it. The first address provides access to the read-only Interrupt Status register and the write-only Clear Interrupt register. The two other address locations provide read-only access to the eight bits of the row and column lines of the keyboard matrix. The fourth address location accesses the inverted multi-purpose port lines, which consist of one keyboard matrix column line, three modifier-key lines (for shift-type keys), and four falling-edge triggered interrupt lines. The fifth address location controls keyboard debounce.

The input clock signal is stopped while the system is in sleep mode. In sleep mode the sampling latches for the keyboard column lines and the multi-purpose lines are driven to a transparent state, and the row line drivers are turned on. An active transition of a column line or multi-purpose line is thus capable of generating an interrupt (if not disabled). Under this condition the debounce timer is bypassed (no delay for a key-generated interrupt). No other latches are affected by sleep mode.

When the system is reset, all interrupt source and status lines are cleared and disabled.

**Keyboard Interrupt Status (OOAOh Read).** The Interrupt Status register is the access to the latches that store the interrupt requests for the keyboard and the multi-purpose interrupt lines. This register is unaffected during sleep mode and is cleared to all zeros when the system is reset.

Bit 7 set to "1" indicates a change of state in a matrix key line.

Bits 6 through 3 give status information about interrupts from the multi-purpose port, which are discussed separately later.

Bit 2 set to "1" indicates that Ext char is pressed.

Bit 1 set to "1" indicates that Shift is pressed.

Bit 0 set to "1" indicates that CTRL is pressed.

**Keyboard Clear Interrupt Request (OOAOh Write).** The Clear Interrupt Request register allows the user to clear each individual interrupt independently. The Clear Interrupt register is different from the Disable Interrupt register in that the interrupt lines become active immediately upon completion of the bus cycle that writes a clear to the individual interrupt sources. When an interrupt occurs, the interrupt service routine must clear the interrupt using this register.

In addition to clearing the interrupt latches, a "1" bit written to a bit position corresponding to the keyboard matrix or any extra column lines will update the comparator latches and set up the keyboard to generate an interrupt on the next change of state.

This register is not affected by sleep mode, but it is cleared when the system is reset.

Bit 7 set to "1" clears interrupt requests for all matrix keys.

Bits 6 through 3 clear interrupt requests for the multi-purpose lines, which are discussed separately below.

Bit 2 set to "1" clears interrupt requests for the Ext char key.

Bit 1 set to "1" clears interrupt requests for the Shift key.

Bit 0 set to "1" clears interrupt requests for the CTRL key.

**Keyboard Disable Interrupt Control (OOA2h Write).** The latch at this address provides control over which interrupt sources are disabled. Setting a bit in this register disables the corresponding interrupt source and clears any pending interrupt from that source. All interrupt sources are unaffected in sleep mode, and are disabled when the system is reset.



This register is *not* Read/Write. You're advised to use the "Alter Interrupt Control Register" system service (Int 50h) to avoid disabling BIOS functionality.

Bit 7 set to "1" disables interrupts from all matrix keys.

Bits 6 through 3 disable interrupts for multi-purpose lines, which are discussed separately below.

Bit 2 set to "1" disables interrupts from the (Ext char) key.

Bit 1 set to "1" disables interrupts from the (Shift) key.

Bit 0 set to "1" disables interrupts from the (CTRL) key.

**Keyboard Control Status (OOA2h Read).** This register provides access to the status of lines. All lines are sampled and latched to provide synchronous access from the data bus. In sleep mode the sampling latches are driven to a transparent state.

Bit 7 set to "1" indicates that a key on matrix column line C9 is pressed.

Bits 6 through 3 indicate the states of the multi-purpose lines, which are discussed separately below.

Bit 2 set to "1" indicates that (Ext char) is pressed.

Bit 1 set to "1" indicates that (Shift) is pressed.

Bit 0 set to "1" indicates that (CTRL) is pressed.

**Keyboard Column Latch (OOA4h Read).** The Column Latch is an eight-bit read-only address location containing the inverted state of the latches that sample the column lines of the keyboard matrix.

Bits 7 through 0 individually set to "1" indicate that keys on the corresponding column lines (C8 through C1) are pressed.

**Keyboard Row Latch (OOA6h Read).** The Row Latch is an eight-bit read-only address location containing the state of the latches that sample the row lines of the keyboard matrix. These latches stop sampling during sleep mode (holding the state of the rows at the time the system went to sleep).

**Keyboard Debounce Disable (OOAEh Write).** The Keyboard Debounce timer can be bypassed by setting bit 3 to a "1". Resetting the bit to a "0" allows the debounce timer to delay interrupt generation. Bit 3 set to "1" causes any change in the state of the keyboard to be held in the sampling latches; an interrupt is generated within one clock period of when the key transition is sampled. Note that only a single row or

column transition is reflected in the state of the sampling latches since the row and column lines are not sampled simultaneously.

Other bits have no effect.

This register is not affected during sleep mode. Bit 3 is set to "1" when the system is reset.

### 7.3.7 Registers - Serial Port

Both multi-controllers can use their serial-port registers. One IC interacts with the modem interface; the other IC interacts with the serial interface. In the descriptions that follow, the first address corresponds to the serial interface, the second address corresponds to the modem interface.



**Note** There are two serial interface inputs which are not handled by a multi-controller. DSR\* and CTS\* are stored in bits 6 and 7 respectively of a register at I/O address 2Fh (the auxiliary input register of the HP-IL controller).

**Serial/Modem Baud Select Low (0044h/00A4h Write).** This register, combined with six bits of the next register, determine the baud rate for the corresponding port. The receive and transmit clock frequencies are derived from the 2.667-MHz input clock. The binary value in the Baud Select registers (14 bits) is used to divide the clock down to the sample rate for the baud clocks. The value is the binary representation of the following decimal equation:

$$\text{value (decimal)} = (1,333,500 / \text{baud}) - 1$$

For example, the value 138 (008Ah) produces 9600 baud; the value 4444 (115Ch) produces 300 baud.

Bits 7 through 0 are the eight least significant bits of the binary value. (The next register contains the six most significant bits.)

The 14 bits of the baud rate divisor (contained in this and the following register) take the value of 5966 (174Eh) when the system is reset (giving 223 baud).

### **Serial/Modem Baud Select High and Loop Back (0046h/00A6h Write).**

This register contains the six most significant bits of the value that determines the baud rate (refer to the previous register). In addition, this register enables loopback operation of the serial port. Bit 7 is ignored.

Bit 6 set to "1" enables serial loopback operation. When enabled, serial loopback causes transmitter output to be gated to the receiver input. It also causes the receive data register to mirror the receive shift register, allowing the CPU to test the receiver at operating speed. When bit 6 is "0", the receiver input is gated from the external input line. The bit is set to "1" when the system is reset.

Bits 5 through 0 define the most significant six bits of the baud rate select value. (Refer to the previous register.)



#### **Note**

There are several ways of setting the baud rate, parity, word length, etc. that are preferable to writing directly to the hardware. They can be set with Int 14h (refer to chapter 5), I/O Control commands (refer to chapter 6), or the PAM Datacom Config function.

### **Serial/Modem Status (0048h/00A8h Read).** This register indicates the status of the serial port. Most bits can be read using Int 5Dh (the AUX Expansion Interrupt).

Bit 7 set to "1" indicates that the receive data register is full (contains a byte of data). It is set when a received data word is transferred into the receive data register. To allow another word to be loaded into the receive data register, the CPU must clear this bit by setting bit 1 at write address 0048h/00A8h (described below). Bit 7 set to "0" indicates that the receive data register is empty. Bit 7 is set to "0" when the system is reset or in sleep mode.

Bit 6 indicates the current receiver input state. When loopback operation is active, bit 6 samples the transmitted output. When loopback is inactive, bit 6 samples the state of the serial input line. The CPU can read this bit to determine the incoming baud rate. Bit 6 is set to "1" in sleep mode and when the system is reset.

Bit 5 set to "1" indicates that an overrun error occurred. It is set when a new data word is received, but the receive data register has not been read (bit 1 at write address 0048h/00A8h has not been set since bit 7 above was last set). Bit 5 is set to "0" when bit 1 at write address 0048h/00A8h is set to "1", indicating that data has been read. Bit 5 is set to "0" in sleep mode and when the system is reset.

Bit 4 is active only while parity is enabled (bit 3 at write address 0048h/00A8h is set to "1" as described below). Bit 4 set to "1" indicates that a parity error was detected in received data. This bit is set to "0" when bit 1 at write address 0048h/00A8h is set to "1", indicating that data has been read. Bit 4 is set to "0" in sleep mode and when the system is reset.

Bit 3 set to "1" indicates that a framing error occurred--a zero is detected when the stop bit is sampled. Bit 3 is set to "0" when bit 1 at write address 0048h/00A8h is set to "1", indicating that data has been read. Bit 3 is set to "0" in sleep mode and when the system is reset.

Bit 2 set to "1" indicates that the receiver detected a break signal (an all-zero signal from the start bit to the first stop bit). Bit 2 is set to "0" when bit 1 at write address 0048h/00A8h is set to "1", indicating that data has been read. Bit 2 is set to "0" in sleep mode and when the system is reset.

Bit 1 set to "1" indicates that the transmit data register is empty. This bit is set when the transmit data register contents are passed to the transmit shift register. Bit 1 set to "0" indicates that the transmit data register is full--when data is loaded into the transmit data register. Bit 1 is set to "1" in sleep mode and when the system is reset.

Bit 0 set to "1" indicates that the transmitter is sending a break signal. (The transmitted break signal is controlled by bit 0 at write address 0048h/00A8h, discussed below.) Bit 0 set to "0" indicates that no break signal is being transmitted. This bit is set to "0" in sleep mode and when the system is reset.

**Serial/Modem Format Control (0048h/00A8h Write).** This register controls the format of the serial port.

7

Bits 7 and 6 specify the word length (the number of data bits) for both the receiver and transmitter:

"00" = 5 bits.

"01" = 6 bits.

"10" = 7 bits.

"11" = 8 bits.

Bits 7 and 6 are set to "11" when the system is reset.

Bits 5 and 4 specify the number of stop bits sent by the transmitter at the end of each word:

- "00" = 1 bit.
- "01" = 1 1/2 bits.
- "10" = 2 bits.
- "11" = 2 bits.

Bits 5 and 4 are set to "11" when the system is reset.

Bit 3 set to "1" enables parity. Parity is disabled if the bit is set to "0". Bit 3 is set to "1" when the system is reset. Bit 2 set to "1" selects odd parity (if parity is enabled by bit 3). If bit 2 is set to "0", then even parity is selected. This bit is set to "1" when the system is reset.

Bit 1 set to "1" clears these status flags: receive register full, overrun error, parity error, framing error, received break detect (bits 7, 5, 4, 3, 2 at read address 0048h/00A8h). Setting this bit to "0" has no effect. Bit 1 is set to "0" in sleep mode and when the system is reset.

Bit 0 set to "1" forces the transmitter to continually send a break signal (low voltage). The signal continues until the bit is set to "0", enabling normal data transmission. Bit 0 is set to "0" in sleep mode and when the system is reset.

**Serial/Modem Received Data (004Ah/00AAh Read).** This register contains the data bits from the receiver. Data is right-justified; if fewer than eight data bits are used, higher bits are set to "0". All bits are set to "0" in sleep mode and when the system is reset.

**Serial/Modem Transmitted Data (004Ah/00AAh Write).** This register contains the data bits to be sent by the transmitter. Data should be right-justified; if less than eight data bits are to be used, higher bits are ignored. The register is set to 209 (D1h) when the system is reset.

**Serial/Modem Interrupt Control (004Ch/00ACh Read/Write).** This register clears interrupt conditions or disables those interrupts. Note that disabling an interrupt also clears that interrupt.

Bit 7 set to "1" disables the "transmit data register empty" interrupt and clears that flag (bit 7 at read address 004Eh/00AEh). Setting bit 7 to "0" enables that interrupt. Bit 7 is set to "1" when the system is reset.

Setting bit 6 to "1" clears the "transmit data register empty" interrupt flag (bit 7 at read address 004Eh/00AEh). Setting bit 6 to "0" has no effect. Bit 6 always reads as "0".

Bit 5 set to "1" disables the "receive data register full" interrupt and clears that flag (bit 6 at read address 004Eh/00AEh). Setting bit 5 to "0" enables that interrupt. Bit 5 is set to "1" when the system is reset.

Setting bit 4 to "1" clears the "receive data register full" interrupt flag (bit 6 at read address 004Eh/00AEh). Setting bit 4 to "0" has no effect. Bit 4 always reads as "0".

Bits 3 through 0 are ignored when written, and always read as "0".

**Serial/Modem Interrupt Status (004Eh/00AEh Read).** This register indicates the conditions that have occurred and caused an interrupt.

Bit 7 set to "1" indicates that the contents of the transmit data register have been loaded into the transmit shift register and are no longer needed. The bit is set only if this interrupt is enabled (by bit 7 at address 004Ch/00ACh set to "0"). Bit 7 set to "0" indicates that data hasn't been shifted out of the transmit register. This bit is set to "0" when either bit 7 or 6 at address 004Ch/00ACh is set to "1". Bit 7 is set to "0" in sleep mode and when the system is reset.

Bit 6 set to "1" indicates that data has been received in the receive data register. The bit is set only if this interrupt is enabled (by bit 5 at address 004Ch/00ACh set to "0"). Bit 6 set to "0" indicates that data hasn't been received in the receive register. This bit is set to "0" when either bit 5 or 4 at address 004Ch/00ACh is set to "1". Bit 6 is set to "0" in sleep mode and when the system is reset.

Bits 5 through 1 are always "0".

Bit 0 indicates the status of the timer interrupt, which is discussed separately below.

### 7.3.8 Registers – Interval Timer

Each of the two multi-controllers has an interval timer. The interval timer has eight registers associated with it, including the shared interrupt status register. One register provides the means to initialize, start, and stop the interval counter and to control interrupts from this timer. Another register provides read-only access to the high byte of the counter, and two other registers provide access to the latched buffers for the middle and low bytes of the counter. Three other registers provide read/write access to

the three bytes of the the interval reference register. This register contains the three-byte number that defines the interval at which the interval counter generates repeated interrupts.

The interval counter counts downward. When activated, it automatically loads itself with the contents of the interval reference registers. When it counts down to zero, it can generate an interrupt. The counter can also be explicitly loaded from the interval reference register.

If an application modifies timer 2 registers, it must restore the previous values in order for the modem "stop" action to operate properly. If an application modifies the heartbeat timer registers, system operation will change accordingly.

**Timer 2/Heartbeat Timer Interrupt (004Eh/00AEh Read).** Bit 1 of this register indicates whether a timer interrupt has occurred. (This register is shared with the serial port.)

Bits 7 and 6 indicate the status of serial port interrupts, which are discussed separately above.

Bits 5 through 1 always read as "0".

Bit 0 set to "1" indicates that a timer interrupt is pending. This means that the interval counter has reached zero. An interrupt can be generated only while the interval counter is counting and while its interrupt line is enabled (bits 2 and 1 at address 0058h/00B8h are set to "0"). Bit 0 set to "0" indicates that either the counter isn't counting or it hasn't reached zero. This bit is set to "0" in sleep mode and when the system is reset.

**Timer 2/Heartbeat Timer Control (0058h/00B8h Read/Write).** This register contains four bit locations to control the operation of the interval timer. Two bits are used to disable or clear an interrupt request generated by the timer. One bit is used to stop and start the counter. One bit causes the interval counter to be loaded with the contents of the interval timer reference registers.

Bits 7 through 4 are ignored when written, and always read as "0".

Setting bit 3 to "1" loads the contents of the three interval reference registers into the interval counter. This data transfer occurs in the middle of the bus cycle that writes this bit to the control register. This action can be executed at any time, whether or not

the counter is enabled to count. A spurious interrupt will not be generated as a result of this action, so it isn't necessary to disable the timer interrupt. Setting bit 3 to "0" has no effect. Bit 3 always reads as a "0".

Bit 2 set to "1" stops the counter from counting and deactivates clock lines, reducing power consumption. While this bit is "1", interrupts are disabled, but not cleared. (Setting the interval counter to zero doesn't cause an interrupt if this bit is "1".) Bit 2 set to "0" enables the counter to count and generate interrupts. Bit 2 is set to "1" in sleep mode and when the system is reset.

Bit 1 set to "1" disables timer interrupts and clears a pending interrupt. Bit 1 set to "0" enables the interval timer to generate interrupts. This bit is set to "1" in sleep mode and when the system is reset.

Setting bit 0 to "1" clears a pending interrupt generated by the interval timer. The interrupt is cleared in the middle of the bus cycle that writes this bit to the control register. Setting this bit to "0" has no effect. Bit 0 always reads as "0".

**Timer 2/Heartbeat Timer reference (0052-0056h/00B2-00B6h Read/Write).** These three registers (at even addresses only) make up the interval reference register, which indicates the starting count value for the interval counter. The interval counter is a 24-bit counter. The value stored in the register (binary representation) is related to the desired interrupt frequency (interrupts per second) by the following decimal equation:

$$\text{value (decimal)} = (444,500 / \text{frequency}) - 1$$

The contents of this register are automatically loaded into the interval counter immediately following the initiation of an interrupt (when the counter reaches zero during continuous interval counting), and also when the load counter bit is set (bit 3 at write address 0058h/00B8h is set to "1"). The low-address register contains the most significant bits; the high-address register contains the least significant bits. Each register is set to 255 (FFh) when the system is reset.

**Timer 2/Heartbeat Timer counter (005A-005Eh/00BA-00BEh Read).** These three registers (at even addresses only) represent the latched value of the 24-bit interval counter. The high byte (at the low address) is read directly from the register. At the time the high byte is read, the middle byte and low byte are latched at the middle and high addresses, so that they can be read next. (The middle and low counter registers aren't read directly.) This is the only way to set these two latches. The counter can be

read at any time (with the counter counting or with it stopped). The three counter registers and the two latches are set to 255 (FFh) when the system is reset.

### 7.3.9 Registers - Multi-Purpose Port

Each multi-controller has four external lines that can be used for generating interrupts or monitoring status. The port has two address locations associated with it. The first address provides access to the read-only Interrupt Status register and the write-only Clear Interrupt register. The other address location accesses the read-only status of the four falling-edge triggered interrupt lines and the write-only disable interrupt register.

The input clock signal is stopped while the system is in sleep mode. In sleep mode the sampling latches for the multi-purpose lines are driven to a transparent state. An active transition of a multi-purpose line is thus capable of generating an interrupt (if not disabled).

When the system is reset, all interrupt source and status lines are cleared and disabled.

In the register descriptions that follow, the association between addresses, bits, and multi-purpose signal lines are defined below:

0040h,0042h - Bit 6: PPUBUSY (from PPU)  
0040h,0042h - Bit 5: IRQ\* (from HP-IL controller)  
0040h,0042h - Bit 4: RING\* (from serial interface)  
0040h,0042h - Bit 3: RLSD (from serial interface)

00A0h,00A2h - Bit 6: EXTINT1\* (from plug-in port #1)  
00A0h,00A2h - Bit 5: EXTINT2\* (from plug-in port #2)  
00A0h,00A2h - Bit 4: MRING\* (from modem interface)  
00A0h,00A2h - Bit 3: MCARRIER (from modem interface)

**Interrupt Status (0040h/00A0h Read).** This register is accessed by the latches that store the interrupt requests for the keyboard and the multi-purpose interrupt lines. This register is unaffected during sleep mode and is cleared to all zeros when the system is reset.

Bit 7 indicates keyboard status, which is discussed separately above.

Bits 6 through 3 when set to "1" indicate that an interrupt is pending in response to a falling edge on a multi-purpose line. When a bit is set to "0", no interrupt is pending.

Bits 2 through 0 indicate keyboard status, which is discussed separately above.

**Clear Interrupt Request (0040h/00A0h Write).** This register allows the user to clear each individual interrupt independently of the others. The Clear Interrupt register is different from the Disable Interrupt register in that the interrupt lines become active immediately after the bus cycle that writes a clear to the individual interrupt sources has been completed. When an interrupt occurs, the interrupt service routine must clear the interrupt by writing to this register.

This register is not affected by sleep mode, but it is cleared when the system is reset.

Bit 7 affects keyboard interrupts, which are discussed separately above.

Setting bit 6 through 3 to "1" clears the corresponding interrupt bit in the interrupt register (read address 0040h/00A0h). Setting any bit to "0" has no effect.

Bits 2 through 0 affect keyboard interrupts, which are discussed separately above.

**Disable Interrupt Control (0042h/00A2h Write).** The latch at this address provides control over which interrupt sources are disabled. Setting a bit in this register disables the corresponding interrupt source and clears any pending interrupt from that source. All interrupt sources are unaffected in sleep mode, and are disabled when the system is reset.

Bit 7 affects keyboard interrupts, which are discussed separately above.

Setting bit 6 through 3 to "1" disables interrupts from the corresponding multi-purpose line. Bits 2 through 0 affect keyboard interrupts, which are discussed separately above.



**Note** This register is *not* Read/Write. You're advised to use the "Alter Interrupt Control Register" system service (Int 50h) to avoid disabling BIOS functionality, possibly crippling the system.

**Control Status (0042h/00A2h Read).** This register provides access to the inverted state of the multi-purpose port lines. All lines of this port are sampled and latched to provide synchronous access to this port from the data bus. Note that the indicated states of the lines are the inverse of the states of the input lines to that port.

In sleep mode and when the system is reset, the sampling latches are driven to a transparent state. Bit 7 indicates keyboard status, which is discussed separately above.

Bits 6 through 3 indicate the *inverted* states of the multi-purpose lines. If a bit is set to "1", the corresponding input signal is low.

Bits 2 through 0 indicate keyboard status, which is discussed separately above.

## 7.4 HP-IL Controller

The operation of the 1LB3 HP-IL controller IC is fully defined in the following manuals:

- Hewlett-Packard Company. *The HP-IL Integrated Circuit User's Manual*. HP part number 82166-90016, c1982.
- Hewlett-Packard Company. *The HP-IL Interface Specification*. HP part number 82166-90017, c1982.

HP-IL control can be performed using BIOS interrupt 54h. It is highly recommended that you use this interrupt rather than going to the hardware directly. Refer also to the HP-IL IRQ interrupt (Int 4Ch).

The 1LB3 controller is turned off in sleep mode. It is reset as the machine wakes up.

The first manual listed above describes the eight internal registers that control the operation of the HP-IL circuit. Each register contains one byte of information. The I/O addresses and functions of these registers are summarized in table 7-2 and following.

Table 7-2. HP-IL Registers

I/O Address	Function
0021h	Status register
0023h	Interrupt register
0025h	Data register
0027h	Parallel poll register
0029h	Loop address register
002Bh	Scratchpad register
002Dh	Scratchpad register
002Fh	Auxiliary input register

**Status Register (0021h).** This register controls the automatic message responses performed by the IC.

	bit 7	bit 0
Read	SC CA TA LA SSRQ RFCR CLIFCR MCL	
Write	SC CA TA LA SSRQ SLRDY CLIFCR MCL	

SC: System controller.

CA: Controller active.

TA: Talker active.

LA: Listener active.

SSRQ: Send service request.

RFCR: RFC received.

SLRDY: Set local ready.

CLIFCR: Clear IFCR.

MCL: Master clear.

**Interrupt Register (0023h).** This register maintains five interrupt flags and their enable bits, three control bits for the message to be received and for the message to be sent.

	bit 7	bit 0
Read	C2in C1in C0in IFCR SRQR FRAV FRNS ORAV	
Write	C2out C1out C0out ← interrupt enable bits →	

C2in-C0in: Control bits of received message.

C2out-C0out: Control bits for transmission.

IFCR: IFC received.

SRQR: Service request received.

FRAV: Frame (message) available.

FRNS: Frame (message) received not as sent.

ORAV: Output register available.

**Data Register (0025h).** This register contains the data bits of the message received and the message to be sent.

	bit 7	bit 0
Read	D7in D6in D5in D4in D3in D2in D1in D0in	
Write	D7out D6out D5out D4out D3out D2out D1out D0out	

D7in-D0in: Data bits of received message.

D7out-D0out: Data bits for transmission.

**Parallel Poll Register (0027h).** This register maintains the status of HP-IL output and input, and controls the automatic response to parallel polls.

	bit 7	bit 0
Read	ORE RERR PPST PPEN PPPOL P2 P1 P0	
Write	- - PPST PPEN PPPOL P2 P1 P0	

ORE: Output register empty.

RERR: Receiver error.

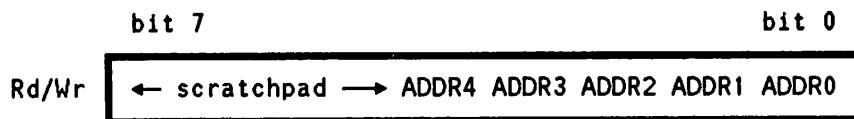
PPST: Parallel poll status.

PPEN: Parallel poll enable.

PPPOL: Parallel poll polarity.

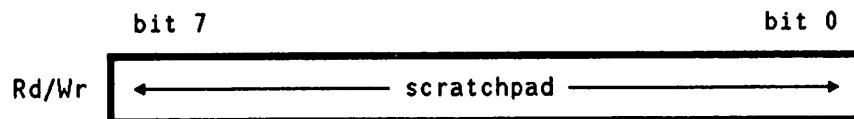
P2-P0: Parallel poll bit designation.

**Loop Address Register (0029h).** This register maintains the primary HP-IL address and provides scratchpad space.

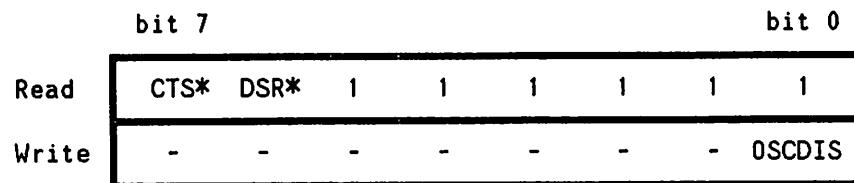


ADDR4-ADDR0: HP-IL address.

**Scratchpad Registers (002Bh,002Dh).** These two registers provide storage for user information.



**Auxiliary Input Register (002Fh).** This register maintains the state of the two input lines from the serial receptacle and controls the HP-IL oscillator.



**CTS\***: Inverted CTS input signal from serial receptacle.

DTR\*: Inverted DSR input signal from serial receptacle.

**OSCDIS:** Oscillator disable.

## 7.5 Display Controller

The 1LM3-0001 is the controller for the 80-character x 25-line (480 dots wide by 200 dots high), four-quadrant liquid crystal display. The 1LM3 is packaged in a 48-pin DIP ceramic hybrid with two 8Kx8 static CMOS RAM die. With the RAM in the hybrid, the 1LM3 provides both display control and display memory in a single package. The hybrid sits directly on an unbuffered 80C86 bus and drives the LCD without buffering.

Also included on the 1LM3 is a circuit (the PPU interface) to eliminate random logic from the computer system. This circuit is totally independent of the LCD control functions.

The 1LM3 generates four data signals and four clock signals for the liquid-crystal display. The four data signals and two of the clock signals are sent to the video interface. *The clock signals must be active whenever power is applied to the LCD.* The clock signals are active while the display is blanked (refer to control register 0 below). All signals are turned off during sleep mode or when the system is reset.

The internal registers of the display controller are not affected during sleep mode. All registers are cleared when the system is rebooted.

Although access to display RAM by the CPU takes place through the display controller, the display RAM logically occupies a block of system address space starting at 80000h. Any CPU access to this space will be directed to the display RAM.

Two modes of display RAM interpretation are provided by the 1LM3. In graphics mode, the display RAM is bit-mapped to the display. In alpha mode, the display RAM is interpreted from a character code and attribute bits through a font definition space to dynamically generate the bits for the screen. In the topics that follow, display RAM mapping is defined *relative to* 80000h in system memory.

### 7.5.1 Display RAM Mapping - Graphics Mode

In graphics mode, the section of display RAM pointed to by the top-of-page pointer (register 2/3) is mapped directly (one bit to one dot) to the display. Because the display is 480 dots wide (not 512 dots), only the 60 least significant bytes of each 64-byte

block of display RAM are displayed. The four most-significant bytes of each 64-byte block of display RAM are don't-cares in graphics mode.

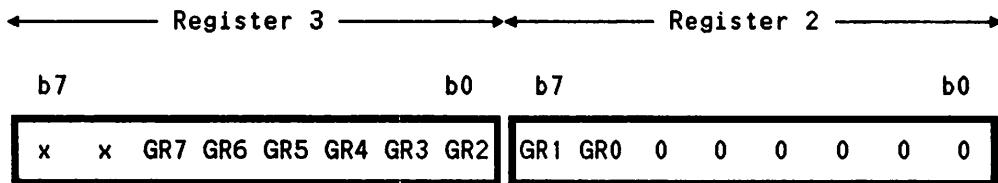
With 16K bytes of display RAM and 64 bytes used per dot row, 256 dots rows of graphics can be stored. This translates to 1.25 screens of graphics display for the 25-line (200 dot-row) display.

The least significant byte of each 64-byte block of display RAM maps to the left-most eight dots on the display. Bit 7 of the least significant byte maps to the left-most dot on the display. A "1" turns its corresponding dot on to black. A "0" turns its corresponding dot off to white. A 9Bh value will map to the display as:

<9 >< B>

■■■■■■■■

Any of the 256 dot-row blocks in display RAM can be put to the top of the display screen by writing the appropriate (16-bit) value to register 2/3. To distinguish 256 dot-rows, the row specifier must be 8 bits (GR7-GRO). Because each dot-row block of RAM is 64 bytes wide, the value written to register 2/3 must be (GR7-GRO)(64), as shown:



In graphics mode, the two most-significant bits of register 2/3 are don't-cares. It is the responsibility of the graphics driver to ensure that bits 0 thru 5 of register 2/3 are "0". Non-zero values in these bits can put the display controller into states that require a hard reset of the chip in order to recover.

Each dot in the display RAM can be addressed by an 8-bit row specifier (GR7-GRO) and a 9-bit column specifier (GC8-GC0). (The left edge of the display screen is dot-column 0 and the right edge of the display screen is dot-column 479 (1DFh).) The byte to be accessed is addressed (relative to display RAM space) by:

a15

a8 a7

a0



The bit within this byte is specified by subtracting the binary values:

$$(1 \quad 1 \quad 1) - (GC2 \quad GC1 \quad GC0)$$

The display RAM dot-row block pointed to by the top-of-page register (register 2/3) will appear at the top of the display screen. Display RAM dot-row blocks of increasing address will be displayed successively down the screen. The display RAM is treated as a continuous cylinder, wrapping from dot-row block 255 (FFh) to dot-row block 0 as necessary. Figure 7-2 shows the relative addresses of all bytes in the graphics display.

**Figure 7-2. Display RAM Mapping - Graphics Mode**

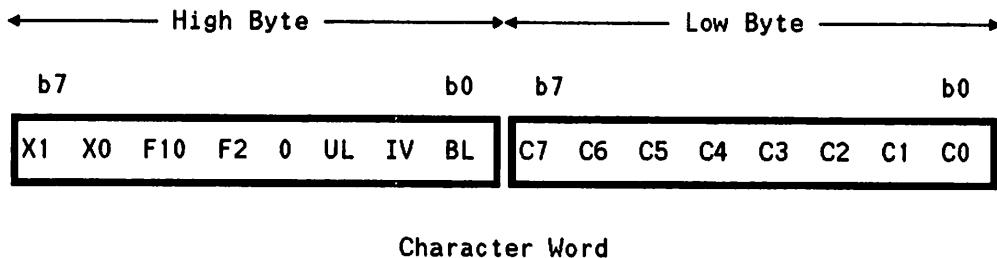
	Columns								Not Displayed					
	0-7	8-15	464- 472-				471 479							
Row 0	0000	0001	.	.	.	003A	003B	003C	003D	003E	003F			
Row 1	0040	0041	.	.	.	007A	007B	007C	007D	007E	007F			
Row 2	0080	0081	.	.	.	00BA	00BB	00BC	00BD	00BE	00BF			
.	.	.	.	.	.	.	.	.	.	.	.			
.	.	.	.	.	.	.	.	.	.	.	.			
.	.	.	.	.	.	.	.	.	.	.	.			
Row 254	3F80	3F81	.	.	.	3FBA	3FBB	3FBC	3FBD	3FBE	3FBF			
Row 255	3FC0	3FC1	.	.	.	3FFA	3FFB	3FFC	3FFD	3FFE	3FFF			

← 60 Bytes → ← 4 Bytes →

## 7.5.2 Display RAM Mapping – Alpha Mode

In alpha mode, the dot pattern displayed on the screen is continuously created "on the fly" by the controller's interpretation of display RAM. The display RAM is divided into text storage and font storage areas.

In the text storage area, each character is represented by a 16-bit word. The low byte of this word contains an 8-bit character code (C7/C0) specifying the character ("A", "7", "#", ...). The high byte contains attribute bits that define how the character will be displayed.



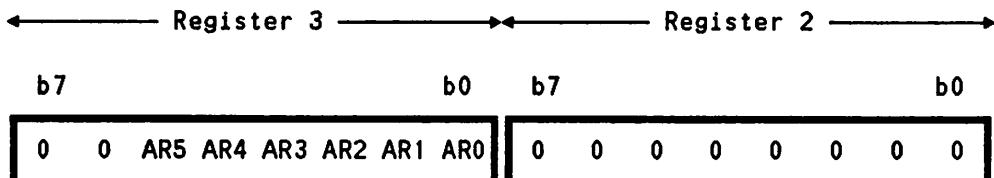
Bit 3 of the high byte of the character word must be "0" to prevent long-term degradation of the LCD.

The font storage area contains the bit patterns for forming characters on the display screen. The mapping for text and font storage areas is illustrated in figure 7-3.

Each display line in alpha mode will be 80 characters wide. The character codes and attribute bits will occupy the 160 least significant bytes in each 256-byte block of display RAM. The least-significant word in each 256-byte block will define the left-most character on the display line.

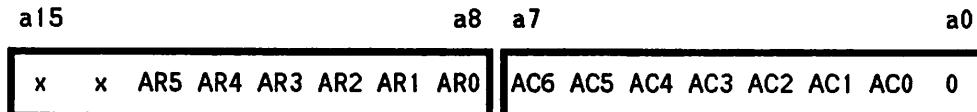
With 16K bytes of display RAM and 256 bytes per alpha line, 64 alpha lines can be stored in display RAM. This will constitute 2.5 pages for the 25-line display. The BIOS normally reserves the first 2 lines for softkey labels.

Any of the 64 alpha lines can be put at the top of the display screen by writing the appropriate (16 bit) value to register 2/3, the top-of-page register. A six-bit row specifier is required for 64 lines (ARS\AR0). Because 256 bytes separate the start of successive lines, the value written to register 2 must be (ARS\AR0)(256).



It is the responsibility of the alpha driver to see that bits 14 and 15 are "0" and that bits 0 thru 7 are "0". Non-zero values in these bits will lead to improper operation of the controller.

The address for a character word is derived from a six-bit alpha-row pointer (AR5\AR0) and a seven-bit character column pointer (AC6\AC0) by:



Column 0 will be displayed at the left edge of the screen.

The display RAM is treated as a continuous cylinder, wrapping from the bottom of display RAM to the top of display RAM. (Refer to "Softkey Menu Display" below for a complete description of the top of display RAM.)

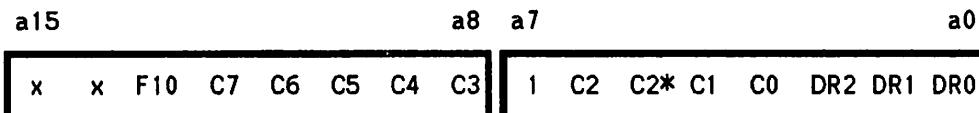
A character font block on the display screen is 6 dots wide and 8 dots high. The block is meant to contain a 5x7 dot character font, with the bottom dot row for descenders and underlines. No character separation is provided in hardware. Character separation must be provided by leaving one edge of the font block blank--usually, the left edge is used as the character separator. Hardware is indifferent to the choice made.

Each character font will be stored in eight adjacent bytes of display RAM. The two least-significant bits of each byte are don't cares. Bit 7 of each byte maps to the left edge of the font block. The least significant byte will map to the top of the font block. A "1" will turn its corresponding dot on to black. A "0" will turn its corresponding dot off to white. The font for a "P" would be stored as:

ooooooo	→	011110xx = 78h	→	Byte 0
ooooooo	→	010001xx = 44h	→	Byte 1
ooooooo	→	010001xx = 44h	→	Byte 2
ooooooo	→	011110xx = 78h	→	Byte 3
ooooooo	→	010000xx = 40h	→	Byte 4
ooooooo	→	010000xx = 40h	→	Byte 5
ooooooo	→	010000xx = 40h	→	Byte 6
ooooooo	→	000000xx = 00h	→	Byte 7

Character/attribute storage uses only 160 bytes of each 256-byte block of display RAM. This leaves RAM space for three blocks of font definitions, 256 characters (2048 bytes) in each block. A character code can be interpreted through any of the three font blocks, depending on the attribute bits F10 and F2.

Font block 0 will be selected by attribute bits F10=0 and F2=0. This font block will be stored as eight fonts (64 bytes) in each 256-byte block of display RAM. The entire font block will be contained in the 32 least-significant 256-byte blocks. The eight fonts in each 256-byte block will occupy addresses xxA0h thru xxDFh. The top font byte for character code 0 will occupy address 00A0h. The bottom font byte for character code FFh will occupy display RAM address 1FDFh. (See figure 7-3.) Given an eight-bit character code (C7\CO), a three-bit font dot-row pointer (DR2\DR0), and the attribute bits F10=0 and F2=0, the appropriate font byte is located at the address:



Font block 1 is selected by F10=1 and F2=0. This block is mapped to display RAM in the same manner as font block 0 with the exception that it occupies space in the 32 most significant 256-byte blocks of RAM. (See figure 7-3.) The mapping from character code, dot-row pointer and attribute bits is of the same form as for font block 0. The top font row for character code 0 is stored at display RAM address 20A0h. The bottom font-row for character code FFh is stored at display RAM address 3FDFh.

Font block 2 is specified by F2=1. (F10 is a don't care when F2=1.) The third font block is stored as four fonts (32 bytes) in each 256-byte block of display RAM. This font block occupies addresses  $xxE0h$  thru  $xxFFh$  in all 64 display RAM blocks. The top font row for character code 0 is at display RAM address  $00E0h$ . The bottom font row of character code FFh occupies address  $3FFFh$ . (See figure 7-3.) The mapping from character code and dot row pointer to font byte address for F2=1 is:

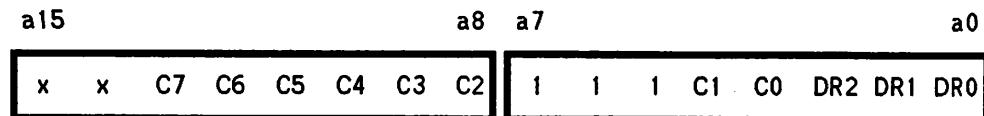


Figure 7-3. Display RAM Mapping - Alpha Mode

	Alpha Character 0 1 ... 79					Fonts 0/1		Font 2	
Row 0	0000	0002	...	009E		00A0	...	00DF	00E0 ... 00FF
Row 1	0100	0102	...	019E		01A0	...	01DF	01E0 ... 01FF
.	.	.		.		.		.	.
.	.	.		.		.		.	.
.	.	.		.		.		.	.
Row 31	1F00	1F02	...	1F9E		1FA0	...	1FDF	1FE0 ... 1FFF
Row 32	2000	2002	...	209E		20A0	...	20DF	20E0 ... 20FF
.	.	.		.		.		.	.
.	.	.		.		.		.	.
.	.	.		.		.		.	.
Row 62	3E00	3E02	...	3E9E		3EA0	...	3EDF	3EE0 ... 3EFF
Row 63	3F00	3F02	...	3F9E		3FA0	...	3FDF	3FE0 ... 3FFF

### 7.5.3 Alpha Attribute Bits

Three bits in each character word affect the manner in which that character will be displayed. Any given character on the display can be independently displayed with any combination of inverse video, underlining, and blinking.

Bit 8 of each character word controls blinking, bit 9 controls inverse video and bit 10 controls underline. For each of the three bits, a "1" is the active state and a "0" is the inactive state. All eight combinations of the attribute bits are both valid and unique.

When blinking alone is asserted, the affected character will be normal for 2/3 of the blink period and will be turned off (all dots in the 6x8 font block are white) for 1/3 of the blink period. The blink period will be about 1 second, depending on the input clock to the chip. (The blink period will be exactly  $(4,608,000)(T)$ , where T is the period of the input clock.) When inverse video alone is asserted, all dots in the 6x8 font block are reversed in sense. A normally black character on a white background will appear as a white character on a black background.

An inverse-video blinking character will be inverse video for the "normal" 2/3 of the cycle and will blink to all dots black in the "blinking" 1/3 of the cycle.

When underline alone is asserted, the bottom dot-row in the 6x8 font block is displayed as all on (black). This will overwrite any descenders in the row.

The underline will be affected in the same way as the rest of the character when blinking is also asserted. It will turn off during the blanking portion of the cycle.

Inverse video will force the underline to appear as a white bar on a black field.

### 7.5.4 Alpha Cursors

Cursor display is handled by hardware in the IC. The display RAM location of the character to be cursoried is specified by registers 4 and 5. Register 4 contains the column of the cursoried character, and so must be a value in the range 0 thru 79 decimal (0h thru 4Fh). Values in the range 80 thru 255 (50h thru FFh) will turn off the cursor.

Register 5 contains the display RAM row of the character to be cursoried. The value in register 5 must be in the range 0 thru 63 (0h thru 3Fh) or the cursor will not be displayed.

The column in register 4 always corresponds to the cursor column on the display. (Column 0 is the left edge of the display.) However, the cursor row on the display will

not match the value in register 5 unless the top-of-page is at display RAM address 0. Pointing the cursor to an alpha row that is not currently on the screen will result in no cursor being displayed.

Either of two cursor displays can be selected by setting or clearing bit 2 in register 0. Clearing the bit causes the underline cursor to be displayed. Setting the bit causes the box cursor to be displayed.

The underline cursor affects only the bottom dot-row in the 6x8 font block. At the cursor position, the bottom dot row will alternate between normal and inverse video with a period of about 0.7 seconds and a 50% duty cycle. In the "normal" period of the cycle, the character appears according to the three attribute bits only. During the active period of the cycle, the inversion is asserted on top of the attribute bit effects. For example, inverse video will be re-inverted and a "blanked" character during blink will have an underline asserted. The block cursor affects the entire 6x8 font block. At the cursor position, the font block will alternate between inverse-video and inverse-video blank. The period is about 0.7 seconds and the duty cycle is 50%. The blinking attribute is entirely overridden by the block cursor. The inversion impressed by the block cursor is asserted after the inverse-video attribute bit, so a double inversion to "normal" is possible.

### 7.5.5 Registers

The 1LM3 contains five register bytes. All five bytes are read/write accessible. The registers are all cleared to "0" when the machine reboots. They are unaffected by sleep mode. Any register can be accessed with either a byte or word operation from the 8086.

Registers 0 through 5 are located at I/O addresses 0080h through 0085h, as listed in table 7-3. Within the 1LM3, the registers are accessed via addresses 0 thru 5 when CSIO\* is asserted (only the three least significant address bits are internally decoded). The byte at address "1" is unused. Writing to this location has no effect. Reading this location will, in general, return meaningless data.

**Table 7-3. Display Controller Registers**

I/O Address	Function
0080h	Control (register 0)
0081h	(Reserved)
0082h	Top of page - low byte (register 2)
0083h	Top of page - high byte (register 3)
0084h	Cursor column position (register 4)
0085h	Cursor row position (register 5)

**Control Register (0080h).** The bits in register 0 have the following effects:

Bit 0 cleared to "0" will blank the display. All four LCD data streams are held low, but the 4 LCD clocks are unaffected. Furthermore, no RAM accesses for LCD refresh will occur. This will allow faster access of display RAM by the CPU. Bit 0 set to "1" will allow data to flow to the LCD. The chip powers on in blanked mode at boot time.

Bit 1 cleared to "0" will cause the data in display RAM to be interpreted in alpha mode. Bit 1 set to "1" will cause the display RAM data to be interpreted in graphics mode. The chip powers on in alpha mode at boot time.

Bit 2 selects the type of cursor to be displayed in alpha mode (this bit is a don't care in graphics mode). Bit 2 cleared to a "0" causes the underline cursor to be displayed. Bit 2 set to a "1" causes the box cursor to be displayed. The power on state is underline cursor mode at boot time.

Bits 3 and 4 specify the number of lines of alpha memory to be locked to the bottom of the display. Zero to three lines may be specified, with bit 3 the least significant bit of the count. These two bits are don't cares in graphics mode. The "no memory lock" option is the power on state at boot time.

Bit 5 set to a "1" causes the LCD M-clock output to become an input clock to the internal counter controlling the cursor and attribute blink rates. This feature is implemented for the benefit of automatic testing only. The assertion of this bit in normal operation can result in permanent damage to the LCD. This bit is cleared to a "0" at power on and at boot time.

Bits 6 and 7 are don't cares. They are both accessible via read and write and can be used by the system at will.

**Top of Page Registers (0082h/0083h).** Registers 2 and 3 are normally accessed at I/O address 0082h by a *word* access. They contain the address of the line in display RAM to be displayed at the top-of-page. In graphics mode, the valid line numbers range from 0 through 255 (FFh). Because each line uses 64 bytes of display RAM, the address in these registers must be (line #)(64). The six least significant bits of register 2 must be "0".

In alpha mode, the valid line numbers range from 0 through 63 (3Fh)--each line needs 256 bytes of display RAM. The value in register 2/3 must be (line #)(256). All eight bits in register 2 must be "0" in alpha mode. Non-zero values in these bits can get the chip into a state recoverable only by rebooting.

Bits 6 and 7 of register 3 are don't cares in graphics mode. In alpha mode, on the other hand, a "1" in either of these two bits will suppress the cursor.

**Cursor Position Registers (0084h/0085h).** These registers specify the location of the cursor on the display. Register 4 specifies the display screen column in which the cursor appears. Valid values for register 4 range from 0 through 79 (4Fh). Any value larger than 79 will effectively turn off the cursor. Note that bit 7 of register 4 can be used as a cursor disable without affecting the column number in the other 7 bits.

The value in register 5 specifies the display RAM alpha row in which the cursor appears. Valid values in register 5 range from 0 thru 63 (3Fh). Any value above 63 will suppress the cursor. Note that the top-of-page pointer must be subtracted from the register 5 alpha row pointer (modulo 64) to find the physical display row the cursor will appear on. (Physical display row 0 is the top of the display.) If register 5 points to an otherwise valid alpha row that is not currently on the display screen, no cursor will appear. No other adverse effects will occur.

In graphics mode, both registers 4 and 5 are don't cares and may be used by the system at will.

## 7.5.6 Softkey Menu Display

The system can lock one, two, or three alpha rows at the bottom of the display screen to serve as fixed menu labels. The alpha lines locked will be the lines at address 0000h, or 0000h and 0100h, or 0000h and 0100h and 0200h. The row at address 0000h will be the top of the locked display. The top-of-page pointer (registers 2/3) will have no effect on the screen position of the locked rows.

The number of alpha rows locked at the bottom of the screen is determined by the 2-bit binary value in bits 3 and 4 of register 0 (bit 3 being least significant).

The normal end-of-display-RAM wrapping will not occur with softkey menu lock in effect. Display RAM row 3F00h will be followed by alpha row 0100h, 0200h or 0300h, as appropriate, if a wrap is necessary. However, it is the responsibility of the software driver to jump the top-of-page pointer over the softkey display rows. Pointing the top-of-page register into the softkey memory area with softkey lock selected will cause two copies of the locked RAM space to be displayed.



# 8

# Memory Management

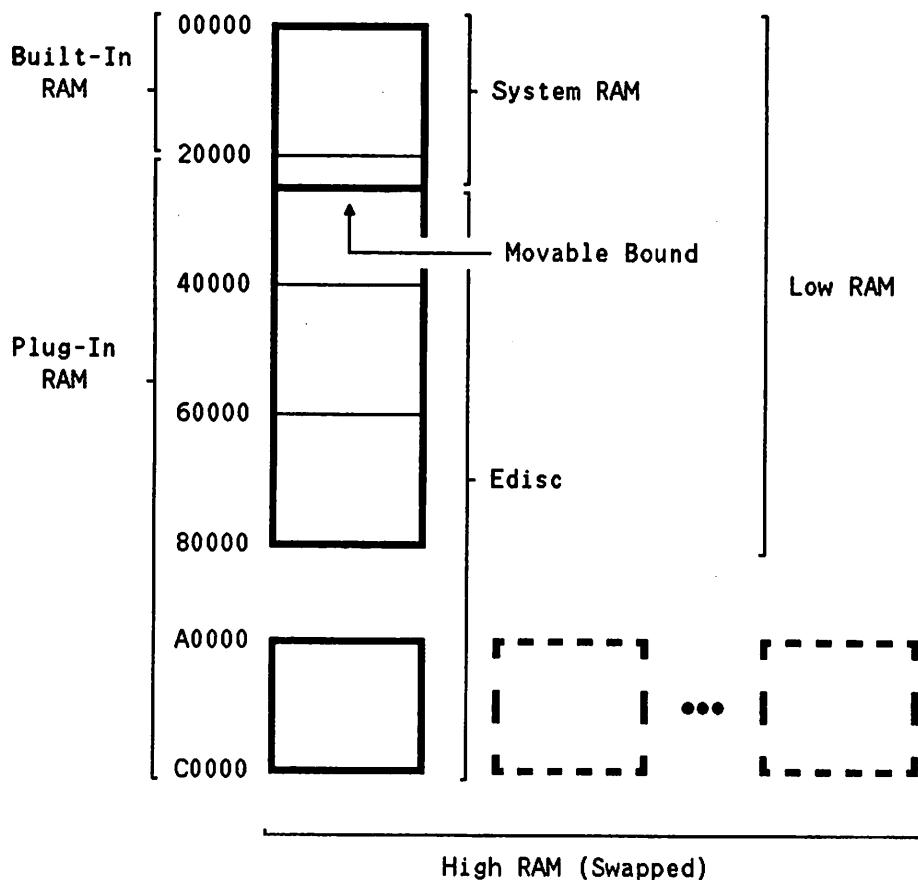
---

## 8.1 Introduction

Memory consists of a variable amount of both ROM and RAM. The ROM portion consists of ROM-resident code (including the boot code, diagnostics, and parts of the BIOS), and a ROM disc. The ROM disc is a logical entity that consists of data extracted from both fixed and plug-in ROMs. The RAM portion consists of up to 512K bytes of contiguous memory as well as a noncontiguous 128K space in which additional RAM can be enabled (128K bytes at a time). See figure 8-1. The RAM contains a portion allocated for use by the operating system (*system RAM*, or *main memory*) and a portion that is allocated to an Edisc (sometimes called *RAM disc*). Only memory in the first 512K contiguous block (called *low RAM*) can be allocated to the operating system. The Edisc can use the low RAM as well as the 128K blocks of noncontiguous memory (called *high RAM*). A separate block of RAM exists for use by the display hardware. This display memory (discussed under "Display Controller" in chapter 7) is *not* manipulated by the memory management functions.

In addition to being allocated to system RAM and Edisc, part of RAM is allocated to Edisc checksums. PAM includes the checksum space as part of the Edisc when displaying the size of the Edisc.

**Figure 8-1. RAM Organization**



The memory management code handles the following tasks:

- Identifying if the data contained in the Edisc is valid (via checksum).
- Identifying the amount of total RAM in the system and the portion that is dedicated to system memory.
- Formatting the Edisc.

- Performing I/O on the Edisc (including enabling and disabling the RAM appropriately as required by the access).
- Identifying plug-in ROMs and constructing the necessary internal tables to configure the logical ROM disc.
- Performing I/O on the ROM disc.
- Enabling and disabling plug-in ROMs.
- Passing control to (and returning control from) plug-in ROMs.
- Identifying the amount of contiguous unused space on the Edisc.

A system memory driver handles all of the disc I/O functions. The remaining functions are handled by system service routines. Formatting the Edisc can be done through either a system service routine (refer to "BIOS Interrupts" in chapter 5) or through an I/O control write to the driver.

---

## 8.2 Edisc

The Edisc occupies all the RAM that is not dedicated to use by the operating system. (See figure 8-1.) This includes RAM beyond the first 512K bytes (if any) as well as any of the first 512 Kbytes beyond the end of the operating system. (The operating system consumes a contiguous block of memory beginning at address 0.) The bound between the operating system and the Edisc is controlled by the Personal Applications Manager (PAM). The bound may be set at any 4K boundary in the first 512K bytes of RAM so long as the minimum sizes for operating system memory and Edisc memory are not violated. The minimum size for the operating system is 80K bytes. The minimum size of the Edisc is between 3K and 8.5K bytes, depending upon the total amount of RAM installed. (The Edisc must be large enough to contain the boot sector, the file allocation table (FAT), and the root directory.)

Although the Edisc is logically a single contiguous block of memory, the Edisc is physically composed of two distinct types of RAM:

- RAM that is addressed in the first 512K of the address space above the operating system memory. This memory is contiguous from the 512K point down to the end of the operating system memory.

- RAM that is on plug-in RAM cards, which are enabled one block (128K bytes) at a time at a fixed location in the high half of the address space. (None of this second type of RAM can be used as operating system memory since it is not contiguous with the lower memory space.)

As plug-in memory is added, it is used to fill out the lower 512K bytes. If 512K bytes of RAM are already in the system, additional plug-in RAM is placed in the pool of swappable memory blocks.

The sectors of the Edisc are assigned to physical memory beginning with the swappable plug-in blocks (if any). The end of the Edisc is that portion resident in the low 512K bytes of memory space (if any). Sectors within this space begin at the high end and work towards lower addresses; thus the last sector on the Edisc (assuming it is in the low 512K-byte space) is adjacent to the top end of operating system memory.

Memory blocks that are dedicated solely to the Edisc are disabled when the memory management code exits as a safety precaution to minimize the risk of a program inadvertently overwriting the Edisc memory. Since system memory must always be enabled, a block of memory that contains both the last sector of Edisc and the high end of system memory will always be left enabled and be vulnerable to program errors. This danger can be avoided by setting the system memory size to a multiple of 128K bytes.

When the memory management code is first entered, it enables all of the low 512K bytes of RAM and the first block of the swappable RAM. Other blocks of RAM are swapped into the address space as required. Since the first swappable block will contain the boot sector, the file allocation table, the root directory, and any space reserved for checksums, it is likely to be the most frequently accessed and is therefore enabled by default when the memory management code is accessed.

Checksums are automatically provided on Edisc access. Checksums are generated and saved whenever a sector is written and checked whenever a checksum is read. The checksum for a sector is a single byte that is generated by summing all of the bytes in the sector with a wrap-around carry (the sum is incremented whenever a carry occurs in summing the bytes). The checksum is the complement of this value (it is the value that when added to the sum produces a value of OFFh).

The checksums are stored in the boot sector and (if more space is required) in memory reserved above the boot sector (which is otherwise at the high end of its memory bank). The boot sector can store 384 checksums (enough for a 192K-byte Edisc)--additional checksum space is allocated in blocks of 512 bytes with enough total space allocated to cover the maximum Edisc size for the amount of memory present. This is independent

of the amount that is actually allocated for Edisc at the time. Note that the checksum for sector 0h (the boot sector) is computed on only the first 128 bytes of the sector (it excludes the checksum data).

As for the checksums, the amount of space required by the file allocation table (FAT) is dependent on the total amount of memory. With one sector per cluster, each sector of FAT has enough entries for a little over 170K bytes of Edisc. Enough sectors of FAT are allocated to cover the maximum sized Edisc that can exist given the total amount of memory.

The root directory of the Edisc is defined to have 64 allowable entries and occupies four sectors immediately following the FAT sectors. This information, together with the rest of the Edisc configuration data, is contained in the BIOS parameter block (BPB) in the boot sector (sector 0h).

Table 8-1 describes the boot sector (sector 0h) of the Edisc.

**Table 8-1. Edisc Sector 0h**

Byte Number	Description (Value)
0h-2h	Dummy Jump to Boot Code (0EBh, 01Ch, 090h)
3h-Ah	OEM Name and Version (45711)
Bh,Ch	Bytes per Sector (512)
Dh	Sectors per Cluster (1)
Eh,Fh	Reserved Sectors (1)
10h	Number of FATs (1)
11h,12h	Number of Root Directories (64)
13h,14h	Total Number of Sectors Depends upon the Edisc size that is set by PAM
15h	Media Descriptor (0FAh)

**Table 8-1. Edisc Sector 0h (Continued)**

16h,17h	<b>Number of Sectors per FAT</b> Depends upon total memory size, and is large enough to cover the maximum Edisc size (1 to 12).
18h-7Fh	<b>Reserved</b> Ignored by the operating system. Can contain miscellaneous system information.
80h-1FFh	<b>Checksums</b> Allocated one byte for each RAM disc sector starting at offset 80h and proceeding forward, possibly overflowing into additional checksum area above the boot sector.

Bytes 0Bh thru 17h of this sector contain the BPB for the disc. The Edisc has 512 bytes per sector, 1 sector per cluster, 1 reserved sector (sector 0h), 1 file allocation table (FAT), 64 allowable entries in the root directory (two sectors worth), and a media descriptor of 0FAh.

The number of sectors per FAT depends on total memory size and is a value from 1 to 12. The total number of sectors in the Edisc is determined from a variable set by PAM. The default is the value that results from setting the system size to its minimum value (maximum Edisc).

The last 384 bytes of sector 0h are dedicated to checksum information for the RAM. Each sector of the Edisc has a checksum contained in a byte within this area (or in the area immediately above the sector, if needed). The mapping of checksums into this region is such that the checksum for sector 0 is at offset 80h in sector 0h; the checksum for sector 1 is at offset 81h; and so forth. After the checksum for sector 383 (which is offset 1FFh in sector 0h), the checksums overflow into the area above sector 0h, which is allocated as required in blocks of 512 bytes for this purpose.

Since cluster numbers are 12 bit quantities and each cluster of the Edisc occupies 512 bytes, there is a maximum of 4K clusters (2 megabytes) that can belong to the Edisc (independent of the amount of physical memory in the machine).

### 8.3 ROM Disc

The ROM disc appears to the operating system as the second unit (B:) of the block device defined by the memory driver. It appears to have a reserved boot sector at sector 0h, 12 sectors allocated to a FAT, a root directory having a maximum of 16 entries (one sector), followed by file data. All files in the root directory are subdirectories. The data for these subdirectories appears to immediately follow the root directory. Following the data for the subdirectory files is the data for the files in the first subdirectory. This will be a system subdirectory containing system files. Last is file data for any remaining subdirectories. This structure is shown in figure 8-2.

**Figure 8-2. ROM Disc**

Sector	Contents
0h	Boot Sector
1h	
.	
.	FAT
.	
Ch	
Dh	Root Directory
Eh	
.	
.	Subdirectory
.	Files
2Dh	
2Eh	
.	System
.	Subdirectory
.	File Data
1E9h	
1EAh	
.	Plug-In
.	Subdirectory
.	File Data
1FE9h	

Actually, the data that appears in the boot sector, the FAT, the root directory, and some of the data in the root's subdirectory files is not actually present in a ROM, but rather is computed on the fly from data in the system ROM, plug-in ROMs, and in previously initialized tables in system RAM. The ROM disc appears permanently write-protected--all attempts to write to the ROM disc will result in a write-protect error.

The first 24 bytes of sector 0h are formatted as a boot sector as specified by the operating system. This includes a 3-byte jump instruction, followed by an 8-byte field for OEM name and version, followed by a 13-byte BIOS parameter block. The BIOS parameter block (BPB) contains information defining the format of the logical ROM disc. This structure is described in table 8-2.

**Table 8-2. ROM Disc Sector 0h**

Byte Number	Description (Value)
0h-2h	Dummy Jump to Boot Code (0EBh, 01Ch, 090h)
3h-Ah	OEM Name and Version (45711)
Bh,Ch	Bytes per Sector (512)
Dh	Sectors per Cluster (2)
Eh,Fh	Reserved Sectors (1)
10h	Number of FATs (1)
11h,12h	Number of Root Directories (16)
13h,14h	Total Number of Sectors Clusters are numbered from 2h to OFEFh beginning at sector 0Eh. With two sectors per cluster, this gives 8170 sectors total.
15h	Media Descriptor (0FAh)
16h,17h	Number of Sectors per FAT (12)
18h-1FFh	Undefined Ignored by the operating system. All bytes are zeros.

The file allocation table (FAT) in sectors 1h through Ch defines the apparent structure of the ROM disc. Each of these sectors is computed on the fly when requested. The entire range of valid cluster numbers from 2h to FEFh is mapped into the ROM disc's logical address space. The first two entries define the media descriptor. Starting with

cluster 2 (the first data cluster) the FAT allocates 16 clusters for subdirectory files of the root directory. Following this, 222 clusters are allocated for the files in the system subdirectory.

Following the system subdirectory files are 15 blocks of 256 clusters each allocated for plug-in ROMs. Every plug-in ROM is mapped into one of these 15 blocks of 256 clusters. This effectively puts an upper limit of 15 ROM applications that can be plugged in at once (independent of whether they are full-bank or half-bank). The number of slots available in the ROM drawers may impose additional limits. Figure 8-3 shows the logical layout of the FAT sectors. (Each FAT entry contains 12 bits.)

**Figure 8-3. ROM Disc FAT**

Sector	Bytes	Contents
1h	0h-2h 3h-1Ah 1Bh-167h 168h-1FFh	First Two FAT Entries (FAh, FFh, FFh) Cluster Allocation for Root Subdirectories Cluster Allocation for System File Data Cluster Allocation for Plug-In #1 Data
2h	0h-E7h E8h-1FFh	]
3h	0h-67h 68h-1E7h 1E8h-1FFh	]
4h	0h-167h 168h-1FFh	]
5h	0h-E7h E8h-1FFh	]
6h	0h-67h 68h-1E7h 1E8h-1FFh	]
7h	0h-167h 168h-1FFh	]
8h	0h-E7h E8h-1FFh	]
9h	0h-67h 68h-1E7h 1E8h-1FFh	]
Ah	0h-167h 168h-1FFh	]
Bh	0h-E7h E8h-1FFh	]
Ch	0h-67h 68h-1E7h 1E8h-1FFh	]
		Cluster Allocation for Plug-In #F Data Not Used

The root directory resides in sector Dh of the ROM disc. When needed, this sector is created on the fly. The sector contains enough room for 16 directory entries. Each directory entry takes 32 bytes and defines a subdirectory file. The first entry defines the subdirectory for the system files. The data for this entry is extracted from system

ROM. Each of the following entries (up to 15 maximum) defines a subdirectory for one of the occupied plug-in ROM slots. The variable data for these entries is extracted from the corresponding plug-in ROM. The ordering of the subdirectories depends upon the number of plug-in applications in "lower" slots. All entries in the root directory are contiguous (all vacant entries are grouped following the occupied ones). The root directory is shown in figure 8-4.

**Figure 8-4. ROM Disc Root Directory**

Sector	Bytes	Contents
Dh	0h-1Fh	Directory Entry for System Subdirectory
	20h-3Fh	Directory Entry for Plug-In #1 Subdirectory
	40h-5Fh	Directory Entry for Plug-In #2 Subdirectory
	60h-7Fh	Directory Entry for Plug-In #3 Subdirectory
	80h-9Fh	Directory Entry for Plug-In #4 Subdirectory
	A0h-BFh	Directory Entry for Plug-In #5 Subdirectory
	C0h-DFh	Directory Entry for Plug-In #6 Subdirectory
	E0h-FFh	Directory Entry for Plug-In #7 Subdirectory
	100h-11Fh	Directory Entry for Plug-In #8 Subdirectory
	120h-13Fh	Directory Entry for Plug-In #9 Subdirectory
	140h-15Fh	Directory Entry for Plug-In #A Subdirectory
	160h-17Fh	Directory Entry for Plug-In #B Subdirectory
	180h-19Fh	Directory Entry for Plug-In #C Subdirectory
	1A0h-1BFh	Directory Entry for Plug-In #D Subdirectory
	1C0h-1DFh	Directory Entry for Plug-In #E Subdirectory
	1E0h-1FFh	Directory Entry for Plug-In #F Subdirectory

Each entry in the "root directory" defines a subdirectory file that is one cluster in length and resides in one of the 16 clusters that immediately follows the "root directory". These files therefore occupy the first 32 data sectors of the ROM disc (clusters 2h through 11h; sectors Eh through 2Dh). When needed, these sectors are generated on the fly from data in the "root directory" of the corresponding plug-in ROM. The specified cluster numbers within the "root directory" of the plug-in ROM are mapped into the corresponding locations within the logical sector space dedicated to that plug-in. The subdirectory files in the "root directory" of the ROM disc have a fixed size of one cluster; therefore, the "root directory" in each of the plug-in ROMs must be fixed at one cluster (two sectors) and can therefore have a maximum of 64 entries. The layout of the subdirectory files is shown in figure 8-5.

**Figure 8-5. ROM Disc Fixed Subdirectory Files**

Cluster (Sectors)	Content
2h (Eh-Fh)	Subdirectory File for System Files
3h (10h-11h)	Subdirectory File for Plug-In #1
4h (12h-13h)	Subdirectory File for Plug-In #2
5h (14h-15h)	Subdirectory File for Plug-In #3
6h (16h-17h)	Subdirectory File for Plug-In #4
7h (18h-19h)	Subdirectory File for Plug-In #5
8h (1Ah-1Bh)	Subdirectory File for Plug-In #6
9h (1Ch-1Dh)	Subdirectory File for Plug-In #7
Ah (1Eh-1Fh)	Subdirectory File for Plug-In #8
Bh (20h-21h)	Subdirectory File for Plug-In #9
Ch (22h-23h)	Subdirectory File for Plug-In #A
Dh (24h-25h)	Subdirectory File for Plug-In #B
Eh (26h-27h)	Subdirectory File for Plug-In #C
Fh (28h-29h)	Subdirectory File for Plug-In #D
10h (2Ah-2Bh)	Subdirectory File for Plug-In #E
11h (2Ch-2Dh)	Subdirectory File for Plug-In #F

Plug-in ROMs can contain 256K of data, which is separated into 512 sectors of 512 bytes each. There is a root directory in the plug-in ROM that describes the organization of its files. The file data itself is not altered by the memory driver except for subdirectories, which specify cluster numbers (within the plug-in ROM) that must be remapped into the appropriate values for the ROM disc. Each of the plug-in ROMs has a reserved bank of 512 sectors (256 clusters) in the ROM disc for its file data. Figure 8-6 shows the allocation of the ROM disc clusters to the plug-in file data. (The format for the data on the plug-in ROM itself is covered in chapter 9, "Plug-In ROM Design.")

**Figure 8-6. ROM Disc Plug-In File Data**

Clusters (Sectors)	Contents
F0h-1EFh (1EAh-3E9h)	Plug-In #1 File Data Area
1F0h-2EFh (3EAh-5E9h)	Plug-In #2 File Data Area
2F0h-3EFh (5EAh-7E9h)	Plug-In #3 File Data Area
3F0h-4EFh (7EAh-9E9h)	Plug-In #4 File Data Area
4F0h-5EFh (9EAh-BE9h)	Plug-In #5 File Data Area
5F0h-6EFh (BEAh-DE9h)	Plug-In #6 File Data Area
6F0h-7EFh (DEAh-FE9h)	Plug-In #7 File Data Area
7F0h-8EFh (FEAh-11E9h)	Plug-In #8 File Data Area
8F0h-9EFh (11EAh-13E9h)	Plug-In #9 File Data Area
9F0h-AEFh (13EAh-15E9h)	Plug-In #A File Data Area
AF0h-BEFh (15EAh-17E9h)	Plug-In #B File Data Area
BF0h-CEFh (17EAh-19E9h)	Plug-In #C File Data Area
CF0h-DEFh (19EAh-1BE9h)	Plug-In #D File Data Area
DF0h-EEFh (1BEAh-1DE9h)	Plug-In #E File Data Area
EF0h-FEFh (1DEAh-1FE9h)	Plug-In #F File Data Area

## 8.4 Summary of ROM Disc Access

The memory driver accesses ROM disc sectors in the following ways:

- If sector 0h is requested, then the boot sector is composed from fixed data in the system.
- If a sector in the FAT is requested (sectors 1h through Ch), then the sector range being mapped is determined and the FAT table entry for those sectors is computed and returned on the fly. The FAT entry for sectors in the plug-in ROM directories or in the system file data is extracted from fixed data within the system. The FAT entry for sectors within a plug-in ROM's file data is computed from the mapping of that plug-in ROM together with data in the root directory of that plug-in.
- If sector Dh is requested, then the root directory of the ROM disc is computed on the fly from data within the root directory of each plug-in ROM and from fixed data within the system.

- Sectors Eh and Fh specify the subdirectory for the system files, which is generated on the fly by the memory management code.
- Sectors 10h through 2Dh specify subdirectory files for plug-in ROMs. These sectors are composed on the fly from data within the root directory of the corresponding plug-in ROM.
- Sectors 2Eh through 1E9h contain the file data for the system files. The contents (and mapping) of these sectors are fixed and are returned directly.
- Sectors 1EAh through 1FE9h are mapped into the data files of the plug-in ROMs. If the sector is from a subdirectory file within the plug-in ROM (which can be determined from data within the plug-in ROM's boot sector), then the cluster specifications (which are at fixed offsets in the sector) must be remapped by adding an offset appropriate to the particular plug-in. If the sector is not part of a subdirectory, then the data is returned as is.



# 9

# Plug-In ROM Design

## 9.1 Introduction

A plug-in ROM can contain up to 256K bytes of data. This data can be divided into three distinct parts:

- ROM boot code.
- A logical ROM disc.
- ROM-executable code.

The logical ROM disc occupies the low memory within the plug-in ROM.

ROM-executable code is placed at the high end. ROM boot code occupies the first sector within the plug-in ROM. The memory driver does not check for this division; the entire plug-in ROM is treated as a 256K-byte partition of its ROM disc.

The plug-in ROM can be implemented as either a half-bank ROM or a full-bank ROM. In a half-bank implementation, only one ROM is provided (instead of two). Because the ROM is 8 bits wide and the data bus is 16 bits wide, a half-bank implementation will occupy either the even bytes or the odd bytes of the address space--depending upon which socket the ROM occupies in the ROM drawer. (A different half-bank ROM may occupy the other half, or the other half may be unoccupied.) The memory driver automatically handles the special requirements of ROM disc accesses to half-bank ROMs.

The address ranges and byte numbering conventions used in the following descriptions assume a full-bank implementation. A half-bank implementation operates in the same way, except that every other location contains an undefined byte (which is discarded by the routine that reads the ROM bank). Thus, the address range for a half-bank implementation is effectively half that of a full-bank implementation. Half-bank ROMs may not (in general) have any ROM-executable code.

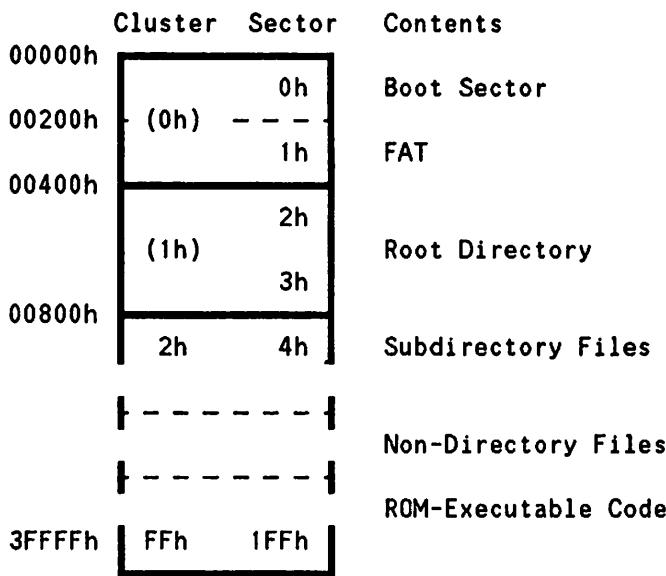
## 9.2 Plug-In ROM Format

The plug-in ROM resides in a 256K address space--some of which may not be occupied, depending on ROM size and whether the implementation is a half-bank or full-bank. The address space is divided into 512 logical sectors of 512 bytes each. The sectors are paired into clusters of 1024 bytes each. On a normal disc, the cluster numbering begins at 2h immediately following the root directory. In the plug-in ROM, the boot sector is in sector 0h, sector 1h is occupied by a one-sector FAT, and the root directory is in sectors 2h and 3h. Therefore, cluster 2h begins in sector 4h, and the first sector of all clusters is twice the cluster number.

The information in these first four sectors is used by the ROM-disc driver to make the files in the plug-in ROM appear as a subdirectory in the ROM disc. As a result, all of the explicit cluster specifications in the plug-in ROM must be remapped into the cluster space of the ROM disc that has been allocated to the particular plug-in. The cluster numbers that occur in the FAT and root directory are at fixed points and easily handled by the memory driver.

Cluster numbers are also specified in subdirectory files. To facilitate the mapping of these cluster specifications, the plug-in ROM format requires that all subdirectory sectors occur before any non-subdirectory sectors (a word in the boot sector is defined to identify this boundary). If the plug-in ROM contains any ROM-executable code, then this code should be placed after the last sector of file data. This format is shown in figure 9-1.

**Figure 9-1. Plug-In ROM Format**



Sector 0h occupies addresses 0h through 1FFh of the plug-in ROM. It is not really a boot sector in that the system does not attempt to boot from it--however, it does have a name and version field and a BIOS parameter block that are defined the same as on a disc boot sector. The sector also contains some information for configuring the plug-in ROM into the ROM disc, and it can contain some ROM boot code that executes during the boot sequence. The format of sector 0h is described in table 9-1.

**Table 9-1. Plug-In ROM Sector 0h**

Byte Number	Description (Value)
0h-1h	<b>ROM-Existence Bytes</b>
2h	<b>Plug-In Status</b> Only one bit is defined. Bit 1 is set to indicate that there is boot code provided starting at byte 30h. (Refer to "ROM Boot Code" below.)
3h-Ah	<b>OEM Name and Version</b> Set up by ROM developer for identification purposes. Ignored by the system.
Bh,Ch	<b>Bytes per Sector</b> (512)
Dh	<b>Sectors per Cluster</b> (2)
Eh,Fh	<b>Reserved Sectors</b> (1)
10h	<b>Number of FATs</b> (1)
11h,12h	<b>Number of Root Directories</b> (32)
13h,14h	<b>Total Number of Sectors</b> (512)
15h	<b>Media Descriptor</b> (0FAh)
16h,17h	<b>Number of Sectors per FAT</b> (1)
18h-1Dh	<b>Extended Device Parameters</b> Ignored by the operating system.
1Eh,1Fh	<b>Last Directory Sector</b> The sector number of the last sector of the last directory file. Because subdirectories must precede all other files, this value identifies the boundary between sectors that contain explicit cluster specifications that must be remapped and those that do not.

**Table 9-1. Plug-In ROM Sector Oh (Continued)**

20h-27h	Subdirectory Name	Identification name for the plug-in ROM. This name is given to the subdirectory that is created by the memory driver for this plug-in ROM. The root directory of the ROM will appear in the ROM disc as a subdirectory with this name.
28h-2Bh	Time and Date Stamps	Used while setting up the subdirectory entry for this plug-in ROM in the ROM disc root directory and in the "." and ".." subdirectories in the plug-in ROM's subdirectory file. The format for these bytes is the same as for the corresponding entries in a directory entry.
2Ch-2Fh	Checksum	
30h-1FFh	Boot Code (Optional)	

The first two bytes of sector Oh contain the ROM-existence bytes, which are used to determine if a valid ROM exists and whether it is a full-bank ROM (B2h,B1h), a half-bank ROM occupying the even addresses (B3h,XX), a half-bank ROM occupying the odd addresses (XX,B3h), or two separate half-bank ROMs (B3h,B3h). To implement this, the first byte in the actual ROM that contains the even address range of a full-bank implementation must be 0B2h; the first byte of the ROM that contains the odd addresses of a full-bank implementation must be 0B1h; and in the ROM that defines a half-bank implementation, the first byte must be 0B3h and the second byte is undefined. (The first byte in a ROM is at address 0 if the ROM is plugged into the even-address socket, or at address 1 if the ROM is plugged into the odd-address socket.)

Bytes 0Bh through 17h contain a BIOS parameter block as defined by the operating system. All of these values are ignored by the system, but the data must be formatted as described in table 9-1.

Bytes 2Ch through 2Fh contain the ROM checksum. The checksum is computed in two parts. One checksum is a 16-bit quantity in 2Ch and 2Eh that covers the even-addressed bytes. The other checksum is a 16-bit quantity in 2Dh and 2Fh that covers the odd-addressed bytes. In a half-bank implementation the two values should be equal. In all cases, the checksum is the value that will produce a -1 (0FFFFh) when all of the bytes in the given half of the ROM are added together in 16-bit pairs with a

wraparound carry. When the BIOS verifies the checksum of a ROM it checks at the 8K, 16K, 32K, 64K and 128K boundaries and stops when it gets a valid checksum.

If bit 1 of the status byte (byte 2) is set, then boot code begins at byte 30h. Boot code is ROM-executable code that is invoked during the boot process. (Refer to "ROM-Executable Code" and "ROM Boot Code" below.)

The file allocation table (FAT) in the plug-in ROM occupies sector 1h. The plug-in ROM maps into a logical address space of 256 clusters (512 sectors). If the plug-in ROM contains ROM-executable code, or if it is a half-bank implementation, then the space available for files may be considerably less, though the files may not use all of the available space. All bytes in the FAT sector beyond those that actually map clusters of file data should be set to zero.

The root directory is a normal directory as defined by the operating system. Three limitations are placed on the directory data of plug-in ROMs:

- All subdirectory data must occur in the ROM before any non-directory file data. This limitation applies only to the file data itself, not to the ordering within the root directory.
- A plug-in ROM's root directory may not have a volume label. The root directory of the plug-in ROM is mapped into a subdirectory within the ROM disc, and a subdirectory can't have a volume label.
- The root directory of a plug-in ROM may have a maximum of 30 entries. The subdirectory size is fixed at two sectors, which can contain a maximum of 32 directory entries, and "." and ".." directories must be added to create a subdirectory.

---

### 9.3 ROM-Executable Code

ROM-executable code is placed in full bank ROMs following the file data. This code is accessed by a user program through the System Services Interrupt 50h functions. (Refer to "System Services Interrupt" in chapter 5.) The plug-in ROM developer may lay out this code as desired, provided it all follows the ROM disc portion of the plug-in ROM. This code will reside in some logical sector of the plug-in ROM file space; however, since the sector is not part of a file, it will be marked by the FAT as an unused sector, and since the sectors are logically (as well as physically) write-protected, the sectors containing ROM-executable code should never be accessed by the file system.

## 9.4 ROM Boot Code

The ROM status byte contains one bit indicating whether or not the ROM has boot code that is to be executed when the system is booted. The boot code resides in sector 0, bytes 30h - 1ffh. At boot, this code is copied into display RAM and called. The ROM may be either a full bank or half bank. This code is called only if the ROM is in the special ROM slot 7. (Refer to chapter 11, "Boot Sequence Options.")

## 9.5 Constraints on Plug-In ROM Software

Though a plug-in ROM is formatted in the standard disc format defined by the operating system, there are several constraints imposed on its contents. While these constraints should not be particularly burdensome, it is important that they be recognized and followed.

- The low memory locations of a plug-in ROM must appear as disc media to the system. This is true even if no files are being supplied. This requires a boot sector, FAT, and root directory, followed by the file data. The boot sector and FAT must define the media as 512 bytes/sector, 2 sectors per cluster, 1 reserved sector, 1 FAT, 32 entries in the root directory, a number of sectors appropriate to the amount of data contained, a media descriptor of OFAh, and 1 sector/FAT. The boot sector, FAT, and root directory must be configured appropriately for the specified disc format and the files therein (as described under "Plug-In ROM Format" above).
- The boot sector must contain special information necessary to determine various attributes of the ROM and the subdirectory being defined. Included here are explicit values to determine that the ROM exists, the optional checksum value, the name of the subdirectory, the total number of sectors of directory files, and a status byte (as described under "Plug-In ROM Format" above).
- The files in the plug-in ROM must be arranged so that the data for all directory files occurs on the ROM before any data for non-directory files.
- Programs may not access data in plug-in ROMs through the use of explicit sector numbers. This is required because the sector numbers within the plug-in ROM are remapped into different numbers within the system ROM disc.

- The explicit pathname for accessing files is the "B:" unit with the subdirectory name defined in bytes 20h to 27h of the ROM. Use of explicit pathnames may limit the portability of applications: there is no guarantee that all machines that will accept the ROM will provide the same pathname configuration.
- The subdirectory name must be unique for every plug-in ROM. Developers should therefore exercise care in selecting the subdirectory name. Names that incorporate a company name or a trademarked product name are good choices. Names such as "games", "user", "etc", "data", "code", "lib", "sys", "lang", "edit", and "dict" should be avoided. The system subdirectory is "BIN".
- The root directory may not contain a volume label. This occurs because the plug-in ROM is mapped into a logical subdirectory file of the ROM disc. However, the developer does have complete freedom in the use of the OEM name and version field and the special version number field. In addition, the time and date stamps can indicate different versions.
- The plug-in ROM is limited to a maximum of 30 entries in the root directory. These together with ":" and ".." entries will total 32 entries, which fill the two sectors allotted to the plug-in ROM's subdirectory file. Subdirectories aren't limited in size or number.

---

## 9.6 PAM Interface to Plug-In ROMs

PAM scans the plug-in ROM directories for a PAM.MNU file. This enables applications to be invoked from PAM in the same manner as disc-based applications. (Refer to chapters 10 and 11.) The PAM.MNU file must be in the root directory of the plug-in ROM. The root directory of the plug-in ROM is set up in the PATH variable by PAM. All programs in the root directory are directly executable.

# PAM – The Personal Application Manager

The Personal Application Manager (PAM) is a "user-friendly" shell that provides the user with an alternate interface to MS-DOS. It also provides additional functionality to the system, allowing system parameters and configurations to be easily set and changed, alarms, and some extended datacom capability in the form of auto-answer batch files. This chapter explains how PAM interfaces to the system BIOS, and how some of its features are implemented.

## 10.1 Power-Up Sequence

Every time the system is booted via the reset button or the **(CTRL)** **(Shift)** **(Break)** combination, PAM is invoked by MS-DOS as the default shell after all of the internal and installable drivers have been initialized. (This assumes that the system has not been taken over by some alternate boot method, such as a boot ROM in slot 7, and that the shell hasn't been changed. Refer to chapter 11, "Boot Sequence Options".)

## 10.2 The PAM Environment

When PAM is started, its first operation is that of building the environment that it will use and pass to any program that is executed under it. The MS-DOS environment consists of a list of strings of the form **NAME=parameter**. Some common environment variables are **PATH**, which defines the path to search for commands; and **PROMPT**, which specifies the DOS prompt. At the minimum, the environment must have the string "**COMSPEC=B:\BIN\COMMAND.COM**". This tells MS-DOS where to reload the transient part of COMMAND.COM from. The string can be changed if you want to use an alternate command processor.

When building the environment, PAM will scan drive A (the Edisc) for a file named **PAM.ENV**. If this file is found, the contents will be read into the PAM environment

space and passed to other programs. This file must have the COMSPEC line described above, and the line must be in upper case.

You can also define a new path, prompt, and any other environment variables you need. The PAM.ENV file can be a maximum of 255 characters. If the file is longer than this, the message "PAM.ENV too big" will appear, and the default PAM environment will be used.

Lines in the PAM.ENV file are normally limited to 80 characters in length--additional characters are normally ignored. However, if a line must be longer than 80 characters, you can include a control-J (^J) or control-Z (^Z) character in the line before the 80th character. Each time PAM encounters one of these characters, it accepts 80 additional characters in that line.

If there is no PAM.ENV file on drive A, or if it is invalid, PAM will build a default environment. This consists of the standard COMSPEC line, the standard prompt (defined by "PROMPT=\$t\$h\$h\$h\$h\$h\$h[\$p]: \$s"), and a path. The PAM path is defined by the plug-in ROMs in the system. In the minimal system, with no external ROMs, the path is set to PATH=A:\;B:\BIN. When external ROMs are present, they are added to the path in the order in which they appear in the directory. Also, they are inserted between the A:\ and the B:\BIN, so that a plug-in ROM can replace a system program if desired. For example, if there is a plug-in ROM with the directory name of LOTUS123, the PAM path would be A:\;B:\LOTUS123;B:\BIN.

In addition to occurring at boot time, this environment construction process occurs every time PAM is restarted.

### 10.2.1 PAM and AUTOEXEC.BAT Files

At boot time, after the environment has been constructed, PAM looks for AUTOEXEC.BAT files on the ROM Disc and Edisc. First, if there is a ROM in slot 7, PAM will execute any AUTOEXEC.BAT file that resides on that subdirectory. If there is no AUTOEXEC.BAT file on the slot 7 ROM, PAM will attempt to execute AUTOEXEC.BAT from the root directory of drive A. The batch file can do the usual commands, and it can either do a normal termination and return to PAM, or invoke some alternate program that will take control of the system.

**Note**

The AUTOEXEC.BAT file is executed before PAM configures the system as specified in the Datacom Configuration and System Configuration menus.

### 10.2.2 PAM Internal State

When PAM finally begins its main execution phase, it puts the console in a known state so that it can process softkeys and display information. The state of the console and the commands required to set it up is given in table 10-1.

Table 10-1. PAM Internal State

State	Command
Insert Char Off	ESC R
Transmit Functions On	ESC &s1A
EOL Wrap Disabled	ESC &s1C
Underscore Cursor	ESC *dL
HP Primary Font	ESC [10m
HP Console Mode	ESC &k0\
Display Functions Off	ESC Z
MS-DOS Raw Mode	MS-DOS Function 44h

---

## 10.3 PAM And Application Programs

### 10.3.1 Installing Applications in PAM

At boot, and every time PAM is restarted (after an application or command terminates), the internal and external discs are scanned for PAM.MNU files. These files specify what appears in the application labels (inverse video blocks) of the PAM Main screen, as well as the command to be executed when that application is selected. The format of a PAM.MNU file is as follows:

```
Title  
Command  
Title  
Command  
#Comment  
...  
...
```

Where "Title" is the wording that will appear in the application label, and "Command" is the actual command to be executed. The line containing "Title" can be up to 80 characters long, but only the first 14 are used. "Command" can be up to 80 characters and must be a valid MS-DOS command. If any line in the PAM.MNU file begins with the "#" character, that line is regarded as a comment and ignored. The PAM.MNU file must be in the top-level directory of a disc or plug-in ROM.

### 10.3.2 The "DOS Commands" Application

Even if there are no PAM.MNU files in the system, PAM will generate an application label with the title "DOS Commands" which, when selected, runs B:\BIN\COMMAND.COM. This label will *always* be the first application label on the screen; it cannot be removed or altered.

### 10.3.3 PAM Execution of a Program

Once PAM displays its main command screen, an application program can be invoked either by the selection of the appropriate application label on the screen, or by typing the program name on the command line. If the application is selected from a screen label, the current default drive is changed to be the disc that the corresponding PAM.MNU file was found on. (Note that this is *not* true for ROM-based applications--the default drive is left unchanged for applications that reside on drive B.)

In either case, PAM then issues various configuration commands to set the machine into a known, user-specified state. All of the Datacom Config settings are invoked so that the communication port parameters (such as baud rate and data bits) are configured properly. The System parameters for Disc Write Verify, Power-Save Mode, Display Timeout, Console Mode, Cursor Type, and Tone Duration are set to their current choices. The printer configuration commands for Line Spacing, Pitch, and Skip Perf are sent to the current PRN device. For more detailed descriptions about the configuration commands that are issued, see the following topic, "The PAM Configurations."

After the system is properly configured, the application or command will be executed. First, PAM releases all but 2K bytes of the memory allocated to itself in order to give the application as much room as possible. Then, the program or command is executed by invoking the command interpreter (COMMAND.COM) with the program name passed as a parameter. For example, if "MemoMakr" is typed on the command line, the actual invocation is "COMMAND.COM /C MemoMakr". When an application terminates, control returns to the resident part of PAM, which then reloads the full PAM.COM from the ROM disc and restarts itself. When PAM restarts, it behaves exactly as if a power-up occurred, except that AUTOEXEC.BAT files are *not* executed.

There is the capability in MS-DOS for an application to terminate and at the same time stay resident in the system memory. When control returns to PAM to restart itself, PAM will find that it can not get all of the system memory back. In this case, PAM will force a re-boot to get the system memory that it requires. This can be demonstrated by going into DOS Commands and running the application PRINT. PRINT terminates but keeps a small amount of its program resident. When exiting to PAM, PAM will re-boot instead of simply restarting.

Note that it is possible to invoke PAM from PAM. This is not recommended, however, since PAM has no way of terminating its execution. If PAM is invoked as an application from PAM, it will "stack up" the resident part of the first PAM as well as the resident portion of COMMAND.COM, removing about 5K bytes from available system memory.

## 10.4 The PAM Configurations

PAM allows the user to configure several aspects of the system. There are three categories: System Configuration, which defines the memory size, external peripherals and other parameters of the system; Datacom Configuration, which give the user a simple means of altering the system data communication settings; and Time and Date Configuration, which defines the system clock.

### 10.4.1 The System Configuration

The fields of the system configuration menu are described below. When the System Config menu is exited, the computer will go through the reboot process before returning to the main PAM screen. This is necessary due to the nature of certain parameters, such as the Main Memory/Edisc partition, which must be set before MS-DOS and PAM gain control of the system.

### 10.4.2 Main Memory and Edisc

The Main Memory/Edisc field defines the partition between system RAM (main memory), which is the amount of RAM available to applications, and the electronic disc. The first number in the field is the number of kilobytes allocated to the system RAM, and the second number is the computed Edisc allocation. The sum of system RAM and the Edisc equals the total RAM in the computer (*user RAM*). The default state, and the minimum state, is 80K system RAM, with all other available RAM allocated to the Edisc. The 80K represents the minimum amount of RAM in which PAM can run and provide its full functionality.

The upper limit of this field is variable, depending on the amount of space currently free on the Edisc. PAM will allow the user to increase the system RAM until there is one sector free on the Edisc. This prevents the accidental loss of data. If additional space is required for system memory, files must be deleted from the Edisc, or it must be packed (via the PACK command) to close any "holes" on the disc. The actual setting of the new partition is done when the "Exit" key is pressed in the System Configuration screen. PAM calls the system memory management function to set the Edisc size, and then re-boots. When the system restarts, the new partition is in effect.

### 10.4.3 External Disc Drives

The number of CS80 protocol disc drives is specified here. This number does *not* include any other drives, such as the Amigo family. At system boot time, the BIOS CS80 driver looks at this number and allocates that many units in its device header. The drive letters for the external drives begin at "C", and continue to a maximum of "J", since the CS80 driver can access a total of eight drives.

If the number of external drives is set to 0, the CS80 block driver is removed from the BIOS device list. This is sometimes useful for speeding up the operation of PAM, since PAM will try to find PAM.MNU files on every external drive. If there are no external drives in the system, PAM will return from its search quickly, since it does not have to wait for the disc driver to time-out or return data. If there are no external drives specified, however, you *cannot* access any CS80 disc drives at any time, unless you have installed your own driver.

### 10.4.4 Disc Write Verify

This option allows the user to set and clear the MS-DOS Verify flag. If this parameter is set to "On", MS-DOS will call the disc driver "Write With Verify" routine instead of the normal "Write with No Verify" code. A verified write is a write that performs a read after each disc write and compares the two blocks of data. If there is an error, the user is given the option to abort, retry or ignore the error. This is normally used only when you are writing critical data to a disc, or if you are unsure of the integrity of the disc (such as a disc that is flashing the "Media Monitor" light on an HP Disc Drive). Usually, since disc errors are infrequent, and since the read after each write slows the disc operation, this parameter is set to "Off." Also, this setting does not affect the operation of the internal drives (Drives A: and B: will never be verified).

### 10.4.5 Power-Save Mode

If an application frequently checks for keyboard activity, battery life can be improved significantly by allowing the system to enter a low-power halt known as Power-Save Mode. In this mode, the CPU is halted; the PPU is made aware of the halt and adjusts its power consumption calculations to account for the reduced system activity. Subsequent hardware interrupts force the CPU out of the halt; the interrupt is serviced as usual, and a check is made to determine if the system should again be halted.

Two situations can cause the low-power halt to occur. The first situation is a keyboard read request when the key queue is empty; the application wants a key, and since the

queue is empty the system halts until a key becomes available. The second situation is an application checking keyboard input status more than a specified number of times within one second.

In the first case, the application requires a key in order to continue; if no key is available, the only logical thing to do is wait for the user to press one. The keyboard driver could wait for a key by continuously testing its own input buffer, waiting for a key to appear; with power-save mode, however, as soon as the driver determines that there is nothing in the buffer, it halts the processor. Whenever a subsequent interrupt occurs, the interrupt is processed, the buffer is rechecked and, if it is still empty, the processor halts again.

In the second case, the application expects a key, but rather than letting the keyboard driver wait for it, the application does the waiting by repetitively reading keyboard input status from the keyboard driver. If power-save mode is enabled, the keyboard driver keeps count of the number of input status requests made within a one second interval; if that count reaches a set limit, the keyboard driver forces the processor into the low power halt. When a subsequent interrupt occurs, the interrupt is processed, the status request counter is reset, and the application again gets control.

System service functions 14h and 15h allow the application to read and write the current value of the keyboard input status request limit. If the limit is set equal to zero, power-save mode *and system timeout* are both disabled. Keyboard input status requests-per-second will not be counted; if the timeout interval elapses, the system will not go to sleep. The low power halt, however, will still be entered if a keyboard read request is attempted when the key queue is empty. If the limit is set above 2000h, power-save mode is disabled. If the limit is set to any other value, it specifies the minimum number of keyboard status requests that must be made within a one second interval in order to enter the low power halt.

#### 10.4.6 Determining a Reasonable Status Limit Value

If an application frequently checks keyboard input status without actually requesting a key from the key queue, it is possible for the system to enter the power-save mode low power halt at times when the halt is unwanted (the limit is set too low); similarly, although the user wants the system to halt and eventually time out (go to sleep), the application might actively wait for a key forever (the limit is set too high). By altering the input status limit, such situations can be avoided; choosing an appropriate limit, however, can be somewhat tricky.

When PAM runs an application (or enters DOS Commands) and Power-Save Mode is enabled, it sets the input status limit to 300h. For most applications, this will be a reasonable value. In some cases, however, it may be too high, causing a program which makes regular but far apart keyboard status checks to never enter the halt (and consequently never allow the computer to go to sleep). It "appears" to the system that the application is doing some intense computing, and so it is permitted to keep on running. PAM disables Power-Save Mode by setting the input status limit to 3000h.

In most programs, the only indication to the user that the status limit value is reasonable is in seeing whether or not the system goes to sleep (if it's supposed to) after the specified timeout interval. The general rule of thumb, then, for determining a new value for the Power-Save mode keyboard input status limit is:

- If the system never goes to sleep when the application is running (but it is supposed to time out), lower the limit.
- If the system goes to sleep (or appears to slow down) at unexpected moments while the application is running, raise the limit.

#### 10.4.7 Display Timeout

Display Timeout is the computer's main means of conserving battery power. It is similar to Power-Save Mode in that it is based on the duration of keyboard inactivity, but instead of simply entering a low-power halt, most of the system is electrically shut down, or *put to sleep*. The CPU, modem, LCD display, and serial and HP-IL interfaces are turned off; only display RAM, built-in RAM, plug-in RAM, keyboard hardware, and the PPU remain on. To the user, the computer will appear to be completely turned off.

Timeout occurs when the system's *time counter*, which increments once each second, reaches a preset *timeout limit*. The timeout limit can be set from PAM's System Configuration menu, or by an application through a system service function. The time counter is only allowed to increment when the system is in Power-Save Mode or keyboard-wait low-power halt (see "Power-Save Mode" above); any I/O activity other than keyboard input status checks resets the time counter to zero.

System service functions 12h and 13h allow an application to read and write the current value of the timeout limit. If the timeout limit is set to zero, display timeout is disabled. Any other value specifies the timeout interval in seconds.

The time counter is a one-word value at address 40h:70h. All built-in device drivers zero out this counter each time they are accessed; *installed drivers should be written to do the same.*

If a recharger is plugged into the system and the battery level is greater than 80%, or if a serial or modem carrier signal is present, then the display timeout feature is disabled. It can also be disabled for long periods of use by selecting the "Off" choice in the configuration.

#### **10.4.8 Cursor Type**

This field controls the shape of the cursor outside of PAM. Within PAM, the cursor is always an underscore. When an application or command is executed, however, PAM sends out the appropriate escape sequence to the console to force the cursor type to match the current PAM selection. The sequences and corresponding choices are shown in the following list:

<u>Selection</u>	<u>Escape Sequence</u>
Underscore	ESC *dL
Box	ESC *dK

#### **10.4.9 Console Mode**

The Console Mode specifies the operation of the console during the execution of an application or command. While PAM is running, the console is always in HP mode, which means that the base font is HP Roman 8, and the special function keys generate HP sequences. In Alt mode, the base font is the Alternate font, and the keyboard generates IBM-style key codes. The escape sequences sent to the console for each mode are shown in the following list:

<u>Selection</u>	<u>Escape Sequence</u>
HP	ESC &k0    ESC [10m
ALT	ESC &k1    ESC [11m

#### 10.4.10 Tone Duration

Whenever an ASCII ^G (character code 7) is sent to the console, the computer will beep. The duration of this beep is selectable. The frequency of the beep is always constant, set by writing a Beep Frequency command to the PPU followed by a parameter of 58h. When a "Long" beep is selected, PAM writes a Beep Duration command to the PPU, followed by an A0h. In "Short" mode, the duration is set to 15h. For further information, refer to the "PPU Communications" (Int 50h) system service description.

#### 10.4.11 Plotter Interface

This field selects the actual communication interface associated with the MS-DOS PLT device. The choices are HP-IL, the HP 82164A HP-IL/RS-232-C Interface, HP-IB (addresses 0-31, accessed through an HP-IL/HP-IB interface), and the built-in serial port. The actual redirection is accomplished by the system at boot time, when the MS-DOS device list is modified to associate the name "PLT" with the currently selected interface.

#### 10.4.12 Printer Interface

The PRN device can be associated with one of several physical devices, and is redirected in the same manner as the PLT device, described above.

#### 10.4.13 Printer Mode

The printer mode variable affects only the operation of the Print Screen function, which is accessed by pressing the Print key, by sending "ESC 0" to the display, or by issuing an Int 5h or Int 53h from an application program. In Alpha Only mode, only alpha (non-graphics) screens will appear on the printer if a Print Screen is invoked. In Alpha and HP Graphics modes, alpha screens will be dumped as alpha data to the printer, and graphic screens will be dumped as graphic data in HP Raster Graphic format (this means the printer must understand the ESC \*rA, ESC \*rB, and ESC \*bW commands). In HP Graphics Only mode, all screens will be dumped as graphic data. A dump of an alpha screen will produce an exact copy of the screen, including the current font, and any inverse video and underlined text.

#### **10.4.14 Printer Pitch, Line Spacing, and Skip Perforation**

The printer can be configured by the user to several combinations of these three parameters. Initially, all of these choices are set to "No Configuration", which implies that PAM simply does not send a configuration to the PRN device. Otherwise, the following escape sequences are sent to the PRN device whenever an application or command is executed from PAM:

<u>Parameter</u>	<u>Escape Sequence</u>
<b>Printer Pitch:</b>	
Normal	ESC &k0S
Expanded	ESC &k1S
Compressed	ESC &k2S
Expanded-Compressed	ESC &k3S
<b>Print Line Spacing:</b>	
6 Lines/inch	ESC &16D
8 Lines/inch	ESC &18D
<b>Printer Skip Perforation:</b>	
On	ESC &l1L
Off	ESC &l0L

#### **10.4.15 Datacom Interface**

The MS-DOS AUX device can be associated with either the built-in RS-232 port, the HP 82164A interface, or the internal modem (if present). This field allows the user to specify which device AUX will address. The redirection is done at boot time in the same way as the PRN and PLT devices.

#### **10.4.16 The Datacom Configuration**

The PAM Datacom Configuration screen allows the user to set the UART parameters for the three serial communication devices in the system: the built-in RS-232C interface (COM1), the HP 82164A HPIL/RS-232-C Interface (COM2), and the internal 1200/300 BPS modem (COM3). All of the choices in this menu are implemented by sending the appropriate IOCTL commands to the COM1, COM2 and COM3 devices (An

IOCTL command is sent to a device via the MS-DOS Int 21h function 44h--for more information about these commands, refer to "AUX Device Driver" in chapter 6.)

#### **Transmission Rate (BPS):**

<u>Choice</u>	<u>Command Sent to Device</u>
110	SB3;
134.5	SB4;
150	SB5;
300	SB6;
600	SB7;
1200	SB8;
1800	SB9;
2400	SBA;
3600	SBB;
4800	SBC;
7200	SBD;
9600	SBE;
19200	SBF;

#### **Word Length:**

<u>Choice</u>	<u>Command Sent to Device</u>
7 bits	SW1;
8 bits	SW0;

#### **Stop Bits:**

<u>Choice</u>	<u>Command Sent to Device</u>
1 bit	SS0;
2 bits	SS1;

#### **Parity:**

<u>Choice</u>	<u>Command Sent to Device</u>
Even	P0;
Odd	P1;
None	P4;

**XON/XOFF Pacing:**

<u>Choice</u>	<u>Command Sent to Device</u>
On	C2;
Off	C0;

**CTS Line Handshaking:**

*This field does not apply to the internal modem.*

<u>Choice</u>	<u>Command Sent to Device</u>
Observe	SL1;
Ignore	SL2;

**DSR Line Handshaking:**

*This field does not apply to the internal modem.*

<u>Choice</u>	<u>Command Sent to Device</u>
Observe	SL5;
Ignore	SL6;

**DCD Line Handshaking:**

*This field does not apply to the internal modem.*

<u>Choice</u>	<u>Command Sent to Device</u>
Observe	SL3;
Ignore	SL4;

### Power to Interface:

In addition to controlling the power to the modem and serial ports, this field serves as an indicator of the current state of the interface. This is useful for conserving battery power, since applications (or the user) can turn the ports on, and then forget about them. For example, selecting the serial port as the system printer interface can cause the port to be powered, increasing battery consumption. The Power field can then be used to turn off the port.

There is no entry for the HP 82164A, since it is an HP-IL device that has no significant power requirements from the mainframe.

<u>Choice</u>	<u>Command Sent to Device</u>
On	M2; (Serial) M4; (Modem)
Off	M3; (Serial) M5; (Modem)

### 10.4.17 The Time and Date Configuration

The system clock can be set with this configuration menu. Everything is fairly straightforward. While in the Time and Date Config menu, the "Current Choices" are actually the settings that were in effect when the screen was activated. If the "Exit" key is pressed without changing any parameters, the time and date continue running unchanged. If any changes have been made with the Next/Previous Choice keys, the system clock will be set to the new values after the "Exit" key is pressed.

Also at exit, the softkey clock numbers (displayed in the PAM screens) are synchronized to the closest *minute*. This means that if the clock was set to 12:00:29, the softkeys will show 12:00 for one minute, advance to 12:01, and continue to advance at one minute intervals. If the softkey clock is to correspond exactly to the system clock, the seconds must be set to 0 before exiting.

After the softkey clock is synchronized, alarms are rescheduled as necessary to account for the new time. For more information on alarms, refer to the next topic.

## 10.5 PAM And Alarms

The PPU chip in the Portable PLUS allows the scheduling of interrupts that will be generated at a given date and time in the future. PAM gives the user a simple means of accessing this feature, along with the capability to display a message or perform a command when an alarm occurs.

### 10.5.1 The PAM.ALM File

When PAM is started, re-entered, or changes the clock, drive A: is searched for a file named PAM.ALM in the root directory. If this file exists, PAM will attempt to read it. Each line in the file must be of the form:

MM/DD/YY HH:SS Text

The hours must be specified in 24-hour time. PAM will only process the first eight alarms in the file, and they must be in chronological order, with the first alarm on the first line. The time and date are decoded and, if they are in the future, sent to the PPU in a SET ALARM command. The text of the line is copied to a file named PAM.MSG, which is created on the Edisc. If the text begins with the ">" character, the rest of the line will be interpreted as an MS-DOS command to be executed when the alarm occurs. Otherwise, the text will be displayed to the user.

### 10.5.2 When an Alarm Occurs

When the PPU generates an alarm interrupt, the system issues an Int 46h, which in turn calls the PAM alarm interrupt service routine. This routine does the "warbling" sound that is heard when an alarm occurs. This noise will continue for about ten seconds or until a key is pressed. After this, the routine sets a flag for the main part of PAM which indicates that an alarm has occurred. If there are any more alarms to be set (defined in the PAM.ALM file), the interrupt service routine sets the next one before exiting.

If an alarm occurs while the machine is in PAM, the alarm screen will immediately appear. If the alarm had a message associated with it in the PAM.ALM file, the message is read from the PAM.MSG file and displayed to the user. PAM will wait for a key to

be pressed before resuming execution. If the alarm had a command, it is executed, and PAM is re-started when the command or program terminates.

If the alarm occurs when an application other than PAM is executing, the ringing will happen, but no other action will be taken until PAM is re-entered. When PAM restarts, it will see that an alarm has occurred, and behave as described in the above paragraph.

If an alarm occurs when the machine is in the sleep state, it will wake up as if a key had been pressed, and then proceed as described above.



**Note** When the PAM.ALM file is created, PAM must be restarted in order for the alarms to be set. If the file is created in an editor that is invoked from the PAM command line or by selecting an application label, then the file will be seen since PAM will restart as soon as the editor terminates. But, if you invoke MS-DOS, and then create the PAM.ALM file, you must "exit" to PAM so it can see the file.

Also, the alarm must be set for at least 2 seconds in the future. If you set an alarm for 12:00, and set the time to 11:59:59, you will get an alarm at 12:01, since PAM takes more than a second to set the alarms. If you set the time to 11:59:58, you will get an alarm at 12:00, as expected.

## 10.6 Autoanswering: PAM and Ring Interrupts

Both the internal modem and an external modem connected to the serial port can generate "ring" interrupts when they are called by another phone or computer. PAM handles these interrupts, and allows a batch file to be run when a ring interrupt is generated. When the system detects a ring interrupt, an Int 4Bh (serial) or Int 42h (modem) is performed, depending on which device is generating the ring signal. PAM's Ring Interrupt Service routine is then invoked. The routine makes a ringing sound with the beeper, and sets a flag for the rest of PAM indicating that a ring has occurred.

If a ring occurs while PAM is running, PAM will attempt to execute a file named AUTOANSR.BAT. This file must be in the root directory of drive A (the Edisc), and if it is present it is executed just as if it had been typed on the command line. It can do anything a standard MS-DOS batch file is allowed to do. When it terminates, PAM is restarted.

If the ring interrupt occurs when PAM is not running, such as during the execution of an application, the ringing sound will occur, but no other action will be taken. PAM will ignore any rings that occur while it is not running. If a ring interrupt occurs while the machine is in the sleep state, it will wake up. If it is in PAM, then the AUTOANSR.BAT file will be executed if possible. If PAM is not running, the machine will act as if it had been awakened by a key.

---

## 10.7 The Battery Fuel Gauge

The current status of the "Battery Fuel Gauge" is displayed for the user on the main PAM screen. A value of 100% represents 2.5 amp-hours, the full capacity of the battery. The PPU estimates and maintains this variable. No actual measurement of the battery capacity is made by the PPU; instead, a value is calculated based on the needs of particular circuits and their status. When no recharger is connected, the PPU decrements the "Battery Fuel" value at various rates that correspond to the estimated current that the computer is drawing from the battery. A series of constants is provided which represent battery current drain for the computer when it is in sleep mode, awake mode, when the serial interface is on, when the modem is on, when the recharger is plugged in, and when a particular plug-in module is installed. (For additional information on the battery fuel gauge, refer to appendix B of *Using the Portable PLUS*.)

There is a system service (Int 50h) that should be used by a plug-in module driver to set the power consumption level of the plug-in.

## 10.8 PAM Help Facility

### 10.8.1 Installing PAMHELP.COM

PAM allows an optional on-line help file to be installed in the system. Whenever the environment is built or the discs are searched for PAM.MNU files, PAM will look for a file named PAMHELP.COM. Not all discs are searched for this file, however.

PAMHELP can reside on only a plug-in ROM or the Edisc (drive A). The plug-in ROMs are searched in the order they appear in the directory of B: so if there are two or more PAMHELP files on the ROMs, the last one found will be the one used by PAM. After the ROM disc is searched, and whenever the external discs are read, drive A (the Edsic) is examined. If there is a PAMHELP file on the Edisc, it will take precedence over any ROM-based help programs.

### 10.8.2 After PAMHELP.COM Is Installed

When a PAMHELP file has been found, the **(F7)** function key, which is normally blank, will have a "Help" label. On almost every PAM screen, the Help key will be visible. If the user presses this key, the last PAMHELP.COM file found will be executed by PAM. The execution of PAMHELP differs slightly from the normal application or command invocation. PAM does not "shrink"; rather, it remains resident and executes COMMAND.COM as a subprocess, with a parameter of PAMHELP.COM. Note that this imposes a restriction on the minimum system memory size in which the help program can run - the system must have at least 92K of RAM allocated to memory. The 80K, 84K, and 88K systems will *not* allow PAMHELP to run--the execution will fail even before PAMHELP.COM is loaded.

When PAMHELP.COM is executed, PAM will pass a parameter to the help program. This parameter will be an ASCII string of numbers, and represents the particular screen that PAM was displaying when help was requested. The parameter can be obtained by the PAMHELP program by examining the PSP parameter area which is located at address CS:80h. The byte at CS:80h is a count of the number of bytes in the parameter, and the bytes themselves begin at address CS:81h. (For more information about the PSP and parameters, refer to the MS-DOS programmer's reference). Table 10-2 lists all possible values for the parameter and the screen that corresponds to each value.

**Table 10-2. PAMHELP.COM Parameters**

Parameter	Screen
0	The Main PAM screen
4	File Manager Print File/Dir
5	File Manager Delete
6	File Manager Make Dir
7	File Manager Choose Dir
8	File Manager Format
9	File Manager Copy
10	File Manager Rename
11	Time and Date Configuration
13	Datacom Configuration
14	System Configuration

Codes 1, 2, 3, and 12 are currently unused.

The PAMHELP.COM program is responsible for setting up its own user interface. It can do whatever it likes to the system, *but it must restore the system to the state listed above under "PAM Internal State"*

When the help program is executed, the system is in the PAM internal state, and it must be in the same state when the program terminates. If it isn't, PAM will not function correctly until it is restarted.

## 10.9 Bypassing PAM With COMMAND.COM

If you wish to replace PAM with COMMAND.COM so that the Portable PLUS will power up as an elementary MS-DOS machine, create a CONFIG.SYS file on drive A that contains the following line:

```
SHELL=B:\BIN\COMMAND.COM /P B:\BIN
```

This instructs MS-DOS to use the COMMAND.COM (located on drive B:) as the new shell. To use something other than PAM or COMMAND.COM as your shell, create a CONFIG.SYS that has the appropriate SHELL command in it. (Refer to "The CONFIG.SYS File" in chapter 11.) Note that any program that is to be used as a shell has to provide some additional functionality that is not usually found in applications. Consult the MS-DOS programmer's reference for more details.



Whenever the shell is changed from the default (PAM), the system will not be configured as specified in the PAM System Configuration and Datacom Configuration menus.



# 11

# Boot Sequence Options

---

## 11.1 Introduction

The Portable PLUS was designed as flexibly as possible to allow for changes in the future without requiring a new BIOS. Several hooks were placed in the boot process to provide additional features or to replace the boot process completely. The various hooks into the system are summarized below in the order in which they occur during the boot sequence.

- Boot into the built-in diagnostics if the **(Shift) (Extend char) (F8)** key combination is pressed.
- Recover from sleep mode.
- Execute boot code from the ROM in ROM Slot 7 of the drawer underneath the **(F1)** key (either full or half bank) before any system RAM is changed. (AX = 0)
- Execute boot code from the Configuration EPROM before any system RAM is changed. (AX = 0)
- Execute boot code from the ROM in ROM Slot 7 (either full or half bank) after the hardware initialization is completed by the BIOS. (AX = 1)
- Execute boot code from the Configuration EPROM after the hardware initialization is completed by the BIOS. (AX = 1)
- Search for the file CONFIG.SYS on the ROM in ROM Slot 7 and if found, boot MS-DOS 2.11 using this file.
- If a CONFIG.SYS file was not found in ROM Slot 7, boot MS-DOS 2.11 searching for a CONFIG.SYS on the default drive.
- PAM searches and executes the file AUTOEXEC.BAT if found on ROM Slot 7.

- If an AUTOEXEC.BAT file was not found on ROM Slot 7, PAM searches and executes the file AUTOEXEC.BAT if found on drive A:.
- 

## 11.2 Boot Sequence

### 11.2.1 Built-In Diagnostics

The built-in Diagnostic routines are accessed by the user by pressing the **Shift Extend char f8** key combination with the display turned off. The instructions to exit the built-in diagnostics are provided on the display and will result in a reset to the CPU.

### 11.2.2 Recover From Sleep

The Portable PLUS has a power-save mode which will turn off the display and cut power to the CPU. This is initiated by calling the Sleep Interrupt (Int 55h) or the Conditional Sleep Service (Int 50h). When waking from sleep, the BIOS recovers from a sleep state to the state that it was in before the sleep was initiated. The word stored in RAM at 40h:0 is loaded into the Stack Pointer and the word at 40h:2 is loaded into the Stack Segment before a far return is executed.

### 11.2.3 ROM Slot 7 Boot Code Before Changing RAM

There is a designated ROM position in the HP 82982A Software Drawer which has special considerations by the BIOS and PAM. This ROM position is referred to as ROM Slot 7 and can be identified by the user by holding a ROM drawer while facing the connector. ROM Slot 7 is in the lower-right-hand corner of the drawer. (Refer to chapter 9 for a description of the plug-in ROMs.)

The BIOS sets up a stack at the end of display RAM, disables all plug-in cards and reads the card IDs for future use. The display RAM is used for several steps of this process and must be initialized before the LCD is turned on. System RAM has not been touched since the reset.

If the HP 82984A is plugged into the slot under the **F1** key and the ROM in ROM Slot 7 contains executable boot code (indicated by the ROM status byte) the ROM address 90030h through 901FFh is copied into display RAM at address 80000h. This

code is called via a far call with AX = 0 to indicate that the system RAM has not been modified since the reset. If the ROM in ROM Slot 7 is only a half bank, the code is copied into the display RAM byte by byte, throwing away the unused low or high bytes. If there are two half bank ROMs in ROM Slot 7, the low address ROM (in the corner) is treated as the ROM in ROM Slot 7.

The executable code may then take over the boot process completely or it may do a far return to the BIOS boot code which will continue booting the system.

#### 11.2.4 Config EPROM Boot Code Before Changing RAM

The configuration EPROM has a boot code status byte which indicates if it contains executable boot code and the length and starting address of the boot code. (Refer to appendix C for a listing of the configuration EPROM.) If there is boot code, it is copied into display RAM at address 80000h. This code is called via a far call with AX = 0 to indicate that the system RAM has not been modified since the reset. The stack pointer is still positioned to the end of display RAM.

The code may then take over the system or do a far return to the BIOS to complete the boot.

#### 11.2.5 ROM Slot 7 Boot Code After Some Initialization

The boot code in ROM Slot 7 is given another opportunity to take over the system after the BIOS has completed the hardware initialization and set up some of the system interrupts. The following events occur before the ROM boot code is called again.

- Download the BIOS RAM code and jump into RAM to execute.
- Disable all hardware interrupts (1LK5, PPU).
- Initialize interrupt vectors 0 - 1Fh and 40h - 5Fh to an IRET instruction.
- Initialize interrupt vectors for sleep, low battery, dead battery, system services and the hardware interrupt (Int OFFh).
- Initialize the LCD contrast if the reset occurred due to the reset button being pushed or the contrast key being held down for several seconds.

- Initialize the PPU power load constants.
- The current country pointer is initialized from information provided in the Configuration EPROM.

The same process of copying the executable code into display RAM occurs but this time AX = 1 to indicate that some system initialization has occurred and the system RAM has been modified.

### 11.2.6 Boot Code From the Config EPROM After Initialization

If there is boot code in the configuration EPROM, it is given control again after the memory management initializaton. The boot code is copied into display RAM using the method described earlier. This time AX = 1 to indicate that some of the system has been initialized and the system RAM has been modified. At the time of the call, the DX register contains the size of system memory (not including the Edisc) in paragraphs. When (and if) this call returns, the size of system memory is modified to the value returned in DX to provide a mechanism of grabbing some RAM that is outside both the Edisc and the MS-DOS system.

### 11.2.7 Boot Using the CONFIG.SYS on the ROM in ROM Slot 7

If we have made it this far, then the operating system to be booted is MS-DOS 2.11. MS-DOS will search and use a CONFIG.SYS file, if found, on ROM Slot 7.

### 11.2.8 Boot Using the CONFIG.SYS on the Default Drive

If a CONFIG.SYS file was not found in ROM Slot 7, then the default drive is checked for a CONFIG.SYS file. The default drive is usually the Edisc, A:. There are two ways to change the default drive to the ROM disc, B:. If the boot occurs by pressing the **Shift** **CTRL** **Extend char** **Break** key combination then the default drive is B:. When booting with a corrupt Edisc, the user has the option of reformatting the Edisc and booting from A: or continuing with the corrupt Edisc and booting from B:.

## 11.2.9 PAM Executes AUTOEXEC.BAT From ROM Slot 7

If the shell was not changed, the default shell PAM will be booted. PAM looks on the ROM in ROM Slot 7 for the file AUTOEXEC.BAT. If this file is found, PAM executes COMMAND.COM to run the batch file at boot.

## 11.2.10 PAM Executes AUTOEXEC.BAT From Drive A:

If there is not an AUTOEXEC.BAT file on the ROM in ROM Slot 7, drive A: is checked for the batch file. If found, PAM executes COMMAND.COM to run the batch file at boot.

---

## 11.3 The CONFIG.SYS File

Every time the system is booted, MS-DOS searches for a configuration file named CONFIG.SYS. The first directory searched is in the special ROM position, ROM Slot 7. If this directory is not found, the default directory (either A: or B:) is then searched. The configuration file is simply an ASCII file that has certain commands for MS-DOS boot. The following commands can be included in the CONFIG.SYS file to alter the default state of the system after booting:

### Defaults:

BREAK:	OFF
BUFFERS:	2
FILES:	10
SHELL:	B:\BIN\PAM.COM

### Break = <On or Off>

The Break command controls when MS-DOS looks for ^C to break out of an executing program. When it is set to Off, the check is made only during certain MS-DOS calls (console input, output among others). If Break is set to On, every time MS-DOS is called, the check will be made.

### Buffers = <number>

This is the number of MS-DOS buffers that are allocated at boot. A buffer is used by MS-DOS as a temporary transfer area when reading from or writing to a disc. The size of memory used by MS-DOS is increased by 1040 bytes for each additional buffer.

**Device = <filename>**

The Device command is used to install the device driver <filename> into the system at boot. The driver must be in the MS-DOS driver format specified in the Programmer's Toolkit.

**Files = <number>**

This is the number of open files that system calls (Int 21h) 2Fh through 5Fh can access. The size of memory used by MS-DOS is increased by approximately 40 bytes for each file opened above 10.

**Shell = <filename>**

The shell command specifies that execution begins with the shell (top-level command processor) in <filename>. To boot using COMMAND.COM as the shell, enter the following line in a CONFIG.SYS file.

```
SHELL=B:\BIN\COMMAND.COM B:\BIN /P
```

This command sets the shell to COMMAND.COM. The B:\BIN argument tells COMMAND.COM where to look for itself when it needs to re-read from disc. The /P parameter tells COMMAND.COM that it is the first program running on the system so that it can process the MS-DOS EXIT command.

Whenever the shell is changed from the default, PAM, the system will not be configured as specified in the PAM System Configuration and Datacom Configuration menus.

**Example:** A CONFIG.SYS file might look like

```
BREAK=ON  
FILES=14  
DEVICE=B:\BIN\AMIGO.SYS  
SHELL=B:\BIN\COMMAND.COM B:\BIN /P
```

# 12

# Modem Interface

---

## 12.1 Overview

The optional modem implements commands and responses that enable it to perform the most common modem functions. The commands are a superset of the Hayes Smartmodem command set. Many software packages that use the Hayes Smartmodem are able to operate with the optional modem and use its capabilities to full advantage. The following paragraphs give an overview of the operation of the modem. Detailed information about commands and responses are presented later.

### 12.1.1 Command Mode and Data Mode

The modem provides two modes of operation: *Command Mode* and *Data Mode*. In Command Mode, the modem is ready to receive and execute commands from the mainframe or to auto-answer an incoming phone call. In Data Mode, the modem can pass data only from the mainframe to the phone line and from the phone line to the mainframe. The modem will not treat any of the data as commands--it ignores the data as it passes the data on through.

The modem automatically moves into Data Mode when it has established a valid data connection with another modem. This occurs when the modem answers an incoming call and successfully handshakes with the calling modem, or when it places a call and successfully handshakes with the answering modem. (Note that if a command is followed by a semicolon (;), the modem will always remain in command mode.)

The modem can also be placed into Data Mode by the execution of the ATO command (go on-line as the originating modem) or the ATA command (go on-line as the answering modem) if the ensuing handshake is successful.

The modem will return to Command Mode any time carrier is lost. (Loss of carrier can be caused by severe noise on the line, the remote modem turning off carrier, or a break in the connection.) It will also return to Command Mode if an "Any Key Abort" (AKA) command is executed, or if the modem is reset.

An AKA command occurs if any byte of data is sent to the modem while it is trying to dial a number or handshake with another modem, but before it has entered into Data mode. When an AKA is executed, the command in progress is terminated immediately and the modem returns to Command Mode. For example, if the command to dial a host computer has been issued but the data connection has not yet been established and the mainframe sends another byte to the modem, the modem will terminate the call, hang up the phone, and return to Command Mode.

The one other way to return to Command Mode from Data mode is to assert the RCM (Return to Command) line to the modem. This is used when you want to change some setting in the modem *without* breaking the data connection. (All other ways of returning to Command Mode break the data connection.) After the RCM line is asserted, the modem issues the "OK" response, returns to Command Mode, but *maintains* the data connection. You may then return to Data Mode by issuing the ATO or ATA command, as appropriate. The state of RCM has no effect when the modem is in Command Mode.

Note that the RCM function replaces the "Stop Action" sequence (pause +++ pause) of the Hayes-compatible modems. (The S2 and S12 registers used by Hayes-compatible modems for the "Stop Action" sequence exist in the modem, but they have no effect on modem operation.) Although the modem itself doesn't implement the "Stop Action" sequence, the AUX driver *does* detect this sequence and asserts the RCM line--so the "Stop Action" sequence *can* be used by the user.

### 12.1.2 Commands

All commands to the modem must be preceded by capital-A capital-T (AT) and must be followed by a carriage return (CR). The only exception to this rule is the "repeat last command line" command, which is invoked by a capital A slash (A/) and executes immediately upon receipt of the slash character.

Multiple commands may be given to the modem in one command line. A command line consists of one command prefix (AT), followed by up to 40 additional characters (which may be either upper or lower case), and ended with a carriage return (CR). If more than 43 characters are received (the AT plus 40 more plus the carriage return), the modem ignores *all* of the characters and issues an error message.

If an AT is given, followed immediately by a (CR) (no command given), the modem returns the "OK" response. This can be used to test if the modem is functioning in Command Mode.

The backspace character has the effect of erasing the most recent character received by the modem. However, the backspace will not erase the AT command prefix once it has been received.

### 12.1.3 Baud Rate Selection

The baud rate selection for the modem is handled by the modem itself. When the AT command prefix is sent, the modem samples the incoming data stream and automatically determines the baud rate and the parity of the AT command prefix data. The modem then assumes that the remainder of that command line will be sent at the same baud rate. This automatic sampling of the baud rate occurs for *every* command line prefix (AT). Thus, it is possible to issue one command line at 1200 baud and then issue the next at 300 baud without doing anything to the modem in between. (Of course, the mainframe baud rate must be changed.)

The baud rates that the modem will recognize while in Command mode are:

- 110 bps.
- 150 bps.
- 300 bps.
- 1200 bps.

The modem will not recognize any other baud rates while in Command mode. However, once a data connection has been established for Bell 103J mode, it will receive and transmit at 0 through 300 bps in a transparent manner.

If the mainframe is talking to the modem at one baud rate and the phone line data connection is made at a different baud rate, there is a speed compatibility problem. When this occurs, there are three different ways the modem will react, based on the response mode selected.

If the minimum response set has been selected (XO command), the modem will issue the "NO CARRIER" error, hang up the phone, terminate the handshake, and return to command mode. The modem will make no attempt to harmonize the transmission speeds.

In the extended response mode (X1 command), the modem will issue the "CONNECT" response when a 300 baud connection is made or it will issue the "CONNECT 1200"

response when a 1200 baud connection is made. It then enters the Data Mode. In this case, the user at least knows what type of connection has been established and can take intelligent action based on that knowledge.

In the full response mode (X2 command), the modem will notify the mainframe that a speed mismatch is in existence by issuing the "SET 1200" response (when the connection is at 1200 bps and the mainframe has been talking to the modem at 300 bps or less) or else the "SET 300" response for the opposite case. The modem then remains in the Command Mode while also maintaining the data connection which has been established. The user (or system) may then change the baud rate appropriately, and then issue an ATA (or ATO) command to enter the Data Mode at the new baud rate. Alternately, any other command could be issued to the modem, including the command to break the connection.

#### 12.1.4 Transmission Settings

In addition to the baud rate, these additional transmission settings are recognized by the modem:

- Parity: Odd, even, or none (must be "none" at 1200 bps and 8 data bits).
- Data Bits: 7 or 8 bits.
- Start Bit: Must be 1 bit.
- Stop Bits: 1 or 2 bits.

Note that even if the baud rates match for the mainframe and phone, it's still possible for the number of data bits, start bits, stop bits, or parity to be out of sync. Any mismatch in these areas can sometimes result in garbled data.

#### 12.1.5 Auto-Answering

The modem will auto-answer an incoming phone call if register S0 is not equal to "0". After the number of rings specified by register S0 have been received, the modem will answer the phone and attempt to establish a valid data connection using the auto-speed selection specified for Bell 212A modems.

If the modem has just been powered up and has *not* received an AT since power up, it will still auto-answer and establish the data connection at 1200 bps. However, it will

not issue any response to the mainframe, including the "RING" responses normally issued prior to answering the phone. It will simply start passing data using default parameters (1200 bps, 7 data bits, even parity, 1 stop bit). There will be no response if the connection is interrupted. No response occurs until an AT is received in Command mode.

If the modem has received an AT since power up, then it will issue the appropriate responses as described in detail in other sections of this document.

**Note**

If the modem thinks that an incoming call is being received ( $S1 \neq 0$ ), it will not allow a dial (ATD) or an originate (ATO) command to be executed. This is significant when accessing some of the dial-back security systems. To get around this, set the modem register S1 to zero before dialing ( $ATS1=0Dnnnn$ , where  $nnnn$  is the dialing sequence).

### 12.1.6 Default State of the Modem

Following a power-on cycle, a hardware reset (via MRESET\*), or the execution of an ATZ (software reset) command, the default state of the modem is as follows:

- **Command Mode.** The modem will be "on-hook" and in Command Mode. This means that the modem is ready to receive and execute commands from the mainframe or to answer an incoming phone call--whichever occurs first.
- **Full Response Set.** The modem's full response set is available for communication with the mainframe. This response set is a superset of the Hayes capabilities. This state is the same as if an ATX2 command had been issued to the modem.
- **Verbose Response Mode.** The modem will communicate to the mainframe in its verbose mode. This means that all responses will be English words preceded and followed by carriage return, line feed (CR LF). This state is the same as if an ATV1 command had been issued to the modem.
- **Echo Commands On.** The modem will echo to the mainframe any bytes received from the mainframe while in Command Mode. This is the same as if an ATE1 command had been sent to the modem.

- **Tone Dialing.** The modem defaults to tone dialing. All dial commands will be executed using tone dialing unless the "P" (pulse) dialing parameter is used in the dial command. (Refer to "Modem Commands" for more details.)
- **S-Register Defaults.** All of the S-registers have default values. Refer to "S-Register Description" for the details of each register and its default value. Note that the S18 register is not affected by a software reset (ATZ command); it is only re-initialized by a hardware reset.
- **Baud Rate Defaults.** The modem has the ability to sample the incoming data from the mainframe and automatically determine the baud rate and parity of the mainframe system. Consequently, the mainframe need not worry about the baud rate settings of the modem under most circumstances. There are two exceptions.

First, if the modem has just been powered up (or reset), no commands have been sent from the mainframe to the modem, and the modem auto-answers an incoming call, then the modem defaults to 1200 bps, 7 data bits, even parity, and 1 stop bit. The mainframe must be set to the same transmission characteristics to avoid garbled communication.

Second, if the modem answers an incoming call and the calling modem is set at a different baud rate than the caller, a speed mismatch will occur. There are several possible results of a speed mismatch. Refer to "Baud Rate Selection" above.

*All of these defaults can be overridden* by issuing the appropriate commands to the modem, as discussed below. (For example, ATX1V0E0 sets the modem to the extended response set, terse response mode, with command echoing disabled.)

---

## 12.2 Modem Commands

The modem implements a set of "smart-modem" commands. Table 12-1 presents information about each of the commands. The commands can be executed only in Command Mode.

Each command line must begin with "AT" and must end with a carriage return (CR). Multiple commands may be issued in the same command line.

If a command that requires a numeric parameter is issued without one (for example, ATE(CR)), the modem interprets it as selecting the "0" parameter.

If the parameter for a command is outside the range specified, the "ERROR" response will be returned, no action will be taken on that command, and the rest of the command line will be ignored.

If a command line contains invalid characters, the modem will simply ignore those characters. (For example, ATWYK(CR) will result in an "OK" response but no other action.)

12

Whenever the modem attempts to go off-hook (A, D, H, and O commands), it will always check that a valid telephone line is connected. If there isn't a valid telephone line, the "LINE?" error response will be sent to the mainframe in the X2 response mode; the "NO CARRIER" response will be sent in either X0 or X1 response mode.

**Table 12-1. Modem Commands**

Command	Description
<b>Answer</b> A	This command is most useful when auto-answer has been disabled (S0=0), the local phone is ringing, and you want the modem to answer. Upon receipt of this command, the modem goes off-hook and puts the answer tone on the telephone line to initiate handshake with the remote modem. When a successful data connection has been established, the modem sends the "CONNECT" or "CONNECT 1200" response to the system and enters Data Mode.  If no data connection is established within 17 seconds, the modem hangs up the line, issues the "NO CARRIER" response, and remains in Command Mode.
<b>Dial</b> D	This command dials an outgoing call. The command must be followed by other characters that indicate the number and other options. Refer to "Dialing" section which follows this table.

**Table 12-1. Modem Commands (Continued)****Command Echo**

- |    |                 |
|----|-----------------|
| E0 | No echo.        |
| E1 | Echo (default). |

This command controls whether or not commands sent to the modem are echoed back to the mainframe. E0 (or E) disables this echo function while E1 enables it.

**Data Echo**

- |    |  |
|----|--|
| F0 | This command is not implemented in the modem--transmit data is never echoed back to the mainframe. Entering an F0 or F1 command generates an "OK" response, but the effect is a NOP. |
| F1 |  |

**Hang Up**

- |    |   |
|----|---|
| H0 | Hang up the phone (go on-hook) (default). |
| H1 | Pick up the phone (go off-hook).          |
| H2 | No effect.                                |

H0 (or H) causes the modem to go on-hook (hang up the phone). H1 causes the modem to go off-hook but remain in the Command Mode. H2 has no function but does return an "OK" response.

**Table 12-1. Modem Commands (Continued)**

**Interrogate**

- I0      Modem identity.  
I1      Modem status.

I or I0 causes the modem to send a modem ID string to the mainframe. This string varies depending on the response set chosen. In X0 or X1, the string is compatible with the Hayes Smartmodem:

(delimiter) 1 2 3 (delimiter)

In X2, the string is six bytes long:

(delimiter) H P P C (space) r (delimiter)

where r is the software revision level (A).

I1 causes the modem to return a status string of the form

(delimiter) s s s s (delimiter) l (delimiter) p (delimiter)

where ssss is the baud rate (0110, 0150, 0300, or 1200), l is the character length (7 or 8 data bits), and p is the parity ("O" odd, "E" even, and "N" none).

The *delimiter* is CR in terse mode and CR LF in verbose mode.

**Monitor**

- M0      This command isn't implemented in the modem because the modem doesn't have a speaker. The command is treated as a NOP and simply returns the "OK" response.  
M1  
M2

**Originate**

- O      The O command attempts to place the modem into Data Mode as the originating (dialing) modem. The modem first tests for a good telephone line and then goes off-hook. In the case where the modem has been forced into Command mode by use of RCM during a data connection, O returns the modem to the Data Mode.

If the modem is on-hook, the O command causes the modem to go off-hook and attempt to handshake with a remote modem. If the data connection is not made within the time spceified by register S7, the modem will go back on-hook and send the appropriate response message to the mainframe.

**Table 12-1. Modem Commands (Continued)****Quiet Responses**

- Q0**      Enables responses (default).  
**Q1**      Disables responses.

Q or Q0 enables the modem to send responses. Q1 prevents the modem from sending any responses. Responses include information returned in reply to a command, such as I or Sr?.

**Reverse/Answer**

- R**      Entering an R in a dialing sequence or in the same command line as an O command causes the modem to handshake as the answer modem instead of as the originate modem. This reversal lasts only for the duration of that specific data connection.

**Display S-Register**

- Sr ?**      This command causes the modem to return the current contents of the S-register specified by the *r* parameter. The value is sent as a three-digit ASCII-coded decimal number (000 through 255), most significant digit first, followed by an "OK" response. The *r* parameter may have a value from 0 through 18. If a value higher than 18 is given, an "ERROR" response is sent to the mainframe.

**Set S-Register**

- Sr = x**      This command sets the S-register specified by the parameter *r* (0 through 18) to the value *x*. All S-registers have default values that are set at power on. Loss of power or a hardware reset will return all S-registers to their default values. The Z command will reset all S-registers except the S18 register. This command causes the "OK" response to be sent to the mainframe upon its completion.

Entering an *r* value greater than 18 or an *x* value greater than allowed for the specific register (refer to "S-Register Description" below) will cause the "ERROR" response to be sent to the mainframe and will leave the S-register unchanged. The *x* value entered must be the ASCII-coded digits of a decimal number.

**Table 12-1. Modem Commands (Continued)**

**Verbosity**

- V0      Terse mode.  
V1      Verbose mode (default).

V or V0 causes the modem to send terse responses to the mainframe (single character) preceded and followed by a delimiter, which is defined as carriage return (CR). V1 allows the modem to send verbose responses to the mainframe (English words) preceded and followed by a delimiter, which is defined as carriage return, line feed (CR LF).

**Vocabulary**

- X0      Minimum response set.  
X1      Extended response set.  
X2      Full response set (default).

X or X0 limits the modem responses to the minimum response set. This provides Hayes Smartmodem 300 compatibility. X1 limits the modem responses to the extended response set. This provides Hayes Smartmodem 1200 compatibility. X2 allows the modem to use the full response set, including tone detection, call progressing, and baud rate compatibility responses.

Responses are described under "Modem Responses" below.

**Note:** In X2 mode, the modem always tries to sense a dial tone before attempting to dial. If a connection already exists, the modem will issue the "NO TONE" response and hang up. This behavior may cause difficulty if the modem is used with security systems which require that an access code be "dialed" after an initial phone call has been made. Setting the modem to either X0 or X1 mode will solve this problem.

**Zap**

- Z      This command resets S-registers 0 through 17 to their default values, reinitializes all other defaults, ensures that the modem is on-hook, and places the modem in Command Mode.

**Table 12-1. Modem Commands (Continued)**

**(Backspace)**

Not a command, the backspace character deletes the previous character in the command line received by the modem. It will not, however, delete the AT command line prefix once the AT has been received.

## **12.3 Dialing**

The first D command in a command line causes the modem to test for a valid phone line, go off-hook, wait for the time specified by register S6, then proceed to dial. If tone detection is enabled (X2 mode), the modem also waits for the dial tone before dialing. If the S6 time is exceeded before dial tone has been detected, the call is aborted and the "NO TONE" response is issued.

If the modem thinks an incoming call is being received (S1 ≠ 0), a D command will cause an "ERROR" response to be issued. If you want to dial under these circumstances (after a ring is received but before a data connection is made), first set S1 to zero and then dial (ATS1=0D $nnnn$ , where  $nnnn$  is the dial sequence).

If a data connection already exists (such as if the modem has answered an incoming call and established carrier), a D command will cause an "ERROR" response to be issued but will *not* interrupt the data connection.

Dialing sequences always begin with a D and end with either a semi-colon (" ; ") or a carriage return (CR).

The ASCII digits "0" through "9", the pound sign "#", and the asterisk "\*" can all be dialed. (The "#" and "\*" are ignored during pulse dialing.)

The "D", "R", "T", "P", " ", and " ; " characters can also be used in a dialing sequence. In X2 mode, "D" can be used within a dialing sequence to require additional dial tone sensing. "T" selects tone dialing. "P" selects pulse dialing ("#" and "\*" are ignored). "R" is the reverse command and is described in the previous table. A " , " inserts a pause of the length specified in the S8 register. Almost any other punctuation character is ignored.

Note that once a dialing mode has been set (either tone or pulse), it becomes the default mode for all subsequent dial commands. The new default remains in effect until the

modem is reset or another "T" or "P" parameter is issued as part of a dial sequence. The modem defaults to tone dialing.

Note also that in order to set the dialing default, the "T" or "P" parameter must be issued *outside* of the dial command string. For example, ATPD7572000 sets the default dialing mode to be pulse dialing and pulse dials the number. On the other hand, ATDP7572000 leaves the default dial mode unchanged but pulse dials this one number. As a final example, assuming that you are in X2 mode, ATTDP9DP116D7572000 causes the default dial mode to be set to tone dialing, pulse dials 9, waits for another dial tone, pulse dials 116, waits for another dial tone and tone dials 7572000.

A ";" causes the modem to remain in Command Mode after dialing, instead of automatically trying to handshake with a remote modem and then going into Data Mode. The O or A commands could then be used to complete the data connection. If the full response set (X2) has been selected, the modem will report the telephone line status (tone sensing) after every dialing sequence ending in a ";". The ";" can also be used to have the modem auto-dial voice calls when another telephone is connected to the same line.

---

## 12.4 Modem Responses

Table 12-2 below summarizes all of the responses the modem is capable of issuing. Note that all of the X0 responses are available when in X1 mode. Similarly, all of the X0 and X1 responses are available when in X2 mode. In other words, each higher mode simply adds more possible responses to the modem vocabulary. Also note that the call monitoring responses (BUSY, VOICE, RR) are sent only when the modem aborts a call because it didn't establish a data connection *or* the modem is told to remain in command mode through the use of the semi-colon (;).

Each response is preceded and followed by a delimiter. In terse mode, the delimiter is carriage return (CR). In verbose mode, the delimiter is carriage return, line feed (CR LF).

Table 12-2. Modem Responses

Terse Response	Verbose Response	Condition When Issued
<b>Mode X0</b>		
0	OK	Normal response to the successful execution of an internal command line.
1	CONNECT	Data connection established and Data Mode entered. In X0 (minimum response) mode, this response means that <i>any</i> valid data connection has been established. In X1 or X2 modes, this response means that <i>only</i> a Bell 103 type (300 bps or slower) data connection has been established.
2	RING	Incoming call ringing detected.
3	NO CARRIER	Call terminated, data connection lost, and back in Command Mode. In X0 or X1 modes, this response also means that there isn't a valid phone line.
4	ERROR	Error in executing a command.
<b>Mode X1</b>		
5	CONNECT 1200	Data connection established at 1200 bps and Data Mode entered.
<b>Mode X2 (Default)</b>		
6	SET 1200	Mainframe speed set to slow, change to 1200 bps.
7	SET 300	Mainframe speed set to fast, change to 300 bps.

Table 12-2. Modem Responses (Continued)

8	TONE	Unrecognized tone on the telephone line.
9	NO TONE	Expected tone not received.
:	BUSY	Busy tone detected.
;	VOICE	Voice detected on the telephone line.
<	RR	Remote ring: the called number is ringing.
?	LINE?	Telephone line not working or not connected to the modem.

## 12.5 S-Register Description

The modem stores 19 registers of information that govern the detailed behavior of the modem. These S-registers (S0 through S18) can be read with the `Sr?` command and changed with the `Sr=x` command. Registers 0 through 17 are reset during power-on, a hardware reset, or execution of a Z command. Register 18 is reset only at power-on or at a hardware reset. Table 12-3 summarizes the operation of the S-registers. Note that "range" and "default" numbers are in decimal and need to be sent to and received from the modem as ASCII-coded decimal numbers.

Table 12-3. Modem S-Registers

Register	Range (Default)	Description
S0	0-255 (2)	<b>Rings to Auto-Answer</b>  Sets the number of rings before the modem auto-answers an incoming call. A value of 0 disables auto-answer.

**Table 12-3. Modem S-Registers (Continued)**

S1	0-255 (0)	<b>Ring Count</b>
		This register contains the count of how many local rings have occurred. It is automatically reset 8 seconds after the last ring is detected. When the value in S1 equals the value in S0, the modem answers the call.
S2	0-255 (43)	<b>Stop Action Character Code</b>
		This register specifies the character that will be recognized as the "Stop Action" character. The default character code of 43 corresponds to the "+" character.
		This "stop action" function is not implemented in the modem software. Consequently, this register is effectively a NOP. (Refer to "Overview" above.)
S3	0-127 (13)	<b>Carriage Return Character</b>
		This register specifies the character that is used as the carriage return character. The default character code of 13 is equivalent to ^M.
S4	0-127 (10)	<b>Line Feed Character</b>
		This register specifies the character that is used as the line feed character. The default value of 10 is equivalent to ^J.
S5	0-127 (8)	<b>Backspace Character</b>
		This register specifies the character that is used as the backspace character. The character value must be a decimal number from 0 to 32, or 127. Backspace is disabled if the value is greater than 127 (128-255). The default value of 8 is equivalent to ^H.

Table 12-3. Modem S-Registers (Continued)

S6	2-255 (2)	<b>Wait Time for Dial Tone</b>
		The modem will wait to sense the dial tone before dialing. If the dial tone is not detected within the number of seconds specified by this register, the dialing sequence is aborted.
S7	1-255 (30)	<b>Wait Time</b>
		The modem will allow the number of seconds specified by this register to establish a valid data connection. When the time expires (or an "Any Key Abort" occurs), the modem goes back on-hook and issues the appropriate "NO CARRIER", "BUSY", "VOICE", or "RR" response.
S8	0-255 (2)	<b>Pause Time for Comma</b>
		When a comma (",") is encountered in a dial command sequence, the modem pauses for the number of seconds specified by this register. Multiple commas result in multiple pauses.
S9	0-255 (0)	<b>Reserved</b>
S10	3-255 (3)	<b>Carrier Off Hang-Up Delay</b>
		When carrier is lost, the modem will wait for the time specified by this register before going on-hook. (Each unit represents 0.1 second.) While waiting, the modem will not send any data, and data received from the mainframe is ignored, preventing any inadvertent passage of data into the command line buffer. At the end of this time, the modem goes on-hook, returns to Command Mode, and issues the "NO CARRIER" response to the mainframe.
S11	0-255 (0)	<b>Reserved</b>

**Table 12-3. Modem S-Registers (Continued)**

S12	0-255 (50)	<b>Stop Action Guard Time</b>
		This register specifies the time intervals used for the recognition of a valid "stop action" sequence. (Each unit represents 0.2 second.) Shorter guard times increase the chance of an inadvertent "Stop Action" occurring.
		This "stop action" function is not implemented in the modem software. Consequently, this register, while it can be changed, is effectively a NOP. (Refer to the "Overview" section for more details.)
S13	0-255 (0)	<b>Available to the Mainframe</b>
S14	0-255 (0)	These registers are available to the user or to the mainframe. These are 8 bit read/write registers.
S15	0-255 (0)	<b>Lamp Control Enable</b>
		This register enables the Lamp Control capabilities of the modem. However, the hardware to support this feature is not provided, and this register is a "don't care".

**Table 12-3. Modem S-Registers (Continued)**

12

S16	0-2 (0)	<b>Local Analog Loopback Control</b>
		<p>Local Analog Loopback (LAL) is used as a self-test of the modem modulation and demodulation circuitry. In LAL, the modem sends the carrier to itself and echoes data back to the mainframe. During LAL, all of the circuitry except the telephone interface, the final output amplifier stage, and the first input amplifier stage is exercised.</p> <p>This register defaults to a value of 0, allowing normal modem usage. A value of 1 enables LAL. A value of 2 is a NOP.</p> <p>During LAL, the modem does not use the phone line and does not generate any responses to the mainframe.</p> <p>When LAL is initiated at 1200 bps, there are a few "spurious" characters output by the modem. This is a result of the type of encoding used by the Bell 212A standard. Any test software needs to filter out these initial bad characters.</p> <p>Note that while in LAL, sending ATS16=0 has no effect. It is treated as data by the modem.</p> <p>The RCM line may be used to end LAL, reset register S16, and return to Command Mode.</p>
S17	0-255 (0)	<b>Reserved</b>

**Table 12-3. Modem S-Registers (Continued)**

S18	0-2 (0)	EIA Control Signals
		<p>Originally designed to dictate the control of the DSR, CTS, DCD, and DTR lines of an RS-232 interface, this register affects only the MCARRIER output of the modem. Values of 0 and 1 enable the MCARRIER output to follow the actual carrier. A value of 2 forces MCARRIER to always be true (as for a "dumb" terminal).</p> <p>This register is <i>not</i> affected by the Z command.</p>

## 12.6 Hayes Compatibility

The modem is compatible with most communications software written for the Hayes Smartmodem 1200, but there are some significant differences. These differences are described below.

**Auto-Answer.** Unlike the Smartmodem, which defaults to auto-answer on the first ring or not at all (switch controlled), the modem defaults to auto-answer on the second ring. This allows extra time to answer manually if desired. You can set the auto-answer to occur on any ring from the first to the 255th, or not at all. To match Smartmodem behavior, set S0 to "0" or "1", depending on the switch setting desired.

**Response Sets.** The modem has the minimum and extended response sets of the Smartmodem (X0 and X1). In addition, it has the full response set (X2), which provides many more features. To fully emulate the Smartmodem, select either the X0 or X1 response set.

**Command Set.** The modem commands are consistent with the Smartmodem with the following exceptions:

- **Cn:** Amateur radio application isn't supported.
- **Fn:** Data echoing isn't supported.
- **H2:** Special off-hook isn't provided.
- **I0:** Modem ID is sent in full response mode.
- **I1:** Status of transmission settings is provided.
- **X2:** Full response set is provided.
- **EIA:** Hardware control of the EIA interface signals CTS, DTR, DSR, DCD isn't provided.

12

**Configuration Switches.** Unlike the Smartmodem, the optional modem provides no configuration switches. In most cases, the equivalent capability is provided by S-registers and commands, as shown below. (Remember that the modem returns to the default values when turned off, hardware reset, or issued a Z command.)

#### Smartmodem Switch

#### Modem Equivalent

Switch 1: DTR Control	DTR not implemented in hardware.
Switch 2: Response Type Down Up	V Command. V0: Terse response mode. V1: Verbose response mode.
Switch 3: Response Inhibit Down Up	Q Command. Q0: Responses sent. Q1: Responses inhibited. (After power-on, no responses are sent until a command has been received.)
Switch 4: Command Echo Down Up	E Command. E0: Commands not echoed. E1: Commands echoed.

	<b>Switch 5: Auto-Answer</b>	<b>S-Register 0.</b>
	Down	S0=0: Auto-Answer disabled.
	Up	S1=1: Auto-Answer first ring. Default: Auto-Answer second ring.
	<b>Switch 6: RS-232 Control Signals</b>	<b>S-Register 18</b>
	Down (CTS, DSR, DCD true)	DCD only signal provided.
	Up (CTS true; DSR, DCD follow carrier)	S18=0: Mcarrier(DCD) follows carrier S18=2: Mcarrier(DCD) always true
	<b>Switch 7: A-Lead Control Enable</b>	Always enabled.
	<b>Switch 8: Command Recognition</b>	Commands always recognized.

## 12.7 Special Considerations for Programmatic Control

### 12.7.1 Modem Power-On Problem

When the BIOS turns on the modem (using an I/O control call or sending a character to the AUX driver via Int 14h), the BIOS does not wait for the response from the modem before it returns. This means that an application program could potentially try to send a command to the modem at the same time the modem is sending its response to being turned on. In this case the command from the application program would not work (refer to next topic). The fix is to wait for at least 21 milliseconds after the BIOS returns (1 character + 21 ms after response complete), or to wait for a linefeed character to be received plus an additional 20 milliseconds.

### 12.7.2 Ignores Characters While Responding

When the modem is in Command Mode and is sending a response to the BIOS, it will ignore any incoming characters from BIOS. After the modem completes an action and issues a response, it is necessary to wait for a minimum of 20 ms before sending it the next command or data. This wait is caused by the modem's auto-baud rate sensing logic.

### **12.7.3 Can't Dial Out While Receiving Ring**

The modem will not allow you to dial when it is receiving a ring from an incoming call ( $S1\neq 0$ ). If you try to dial when  $S1\neq 0$  (the modem thinks there is a call coming in), the modem will return the "ERROR" response. You can trick the modem by setting  $S1=0$  and then dialing ( $ATS1=0Dnnnn$ , where  $nnnn$  is the dialing sequence). This becomes an issue with some of the "Defender" type security systems that call you back and yet expect a certain tone before establishing the connection.

12

### **12.7.4 Spurious Extra Characters Generated**

When the modem is set up for 8 data bits, 1 stop bit, and no parity, the UART will occasionally produce an additional character (0FFh) when receiving back-to-back characters from the host.

A fix for this problem is to ensure that characters sent to the modem are separated by at least an 8.33-millisecond delay.

### **12.7.5 Spurious Interrupts at Power-Up**

This problem is a concern only if you bypass the AUX driver when doing data communications (that is, if you write your own driver).

When first powering up the modem, there will be a spurious interrupt generated on the MCARRIER line and, if there is a phone line connected, the MRING\* line. These spurious interrupts need to be masked out. The MCARRIER interrupt occurs approximately 25 msec after the MRESET\* line is taken high. The MRING\* interrupt occurs immediately after power is applied to the modem.

---

## **12.8 Directly Connecting Two Modems**

The Portable (HP 110) and the Portable PLUS modems sense the DC line current in order to determine if a "valid" phone line is connected. Therefore it is not possible to connect two of these modems back-to-back using a simple phone cable. You can create the needed DC bias voltage by connecting a 4.0-volt DC supply (with AC bypass coupling) in series with either the tip or ring wire.

This also applies when hooking up a "long haul" phone line simulator box.



# 13

# Keyboards and Keycodes

---

This chapter describes the available keyboard language options and lists the corresponding character code mappings.

Each unit is equipped with one keyboard language option. A corresponding internal configuration EEPROM maps each key location to one character code or a sequence of character codes. This *localized* mapping enables the computer to provide operation that's compatible with many local languages.

The character codes that are generated for each key depend upon the mode of the CONsole driver: HP mode or Alternate mode. (Refer to "CONsole Driver" "Keyboard Modes" in chapter 5.) The tables in this chapter are organized in pairs, showing the responses in HP mode and Alternate mode.

The first four tables below list the character codes that are generated for keys that are shared by all keyboard versions. The first two list codes for non-character keys (English labels are listed; other keyboards use the same key locations). The next two tables list codes for "common" character keys.

For each language, the following information is presented:

- A figure showing the keyboard layout.
- A table listing character codes for HP mode--for keys that aren't "common".
- A table listing character codes for Alternate mode--for keys that aren't "common".

The last two tables in the chapter list the "muted" character codes. Muted keys are indicated in the main character code tables as "--". When one of these keys is pressed, no character code is generated, but the *next* key generates a character code according to the "muted key" table, rather than the main table.

The following information applies to the character code tables in this chapter:

- All character codes are shown in hexadecimal form.
- A blank entry indicates that no character code is generated and no response occurs.
- Many keys generate more than one character code. The sequences of codes for these keys are listed in the order they're generated.

**Table 13-1. HP Mode Character Codes**

English Keycap	Normal	Shift	CTRL	Shift- CTRL	Ext	Shift- Ext	Ext- CTRL	Shift- Ext- CTRL
<b>f1</b>	1B 70 (ESC p)	1B 70 (ESC p)	1B 70 (ESC p)	1B 70 (ESC p)	(cursor)	(beep)	(beep)	(beep)
<b>f2</b>	1B 71 (ESC q)	1B 71 (ESC q)	1B 71 (ESC q)	1B 71 (ESC q)	(dsp fcn)	(beep)	(beep)	(beep)
<b>f3</b>	1B 72 (ESC r)	1B 72 (ESC r)	1B 72 (ESC r)	1B 72 (ESC r)	1B 4B (ESC K)	1B 4B (ESC K)	1B 4B (ESC K)	1B 4B (ESC K)
<b>f4</b>	1B 73 (ESC s)	1B 73 (ESC s)	1B 73 (ESC s)	1B 73 (ESC s)	1B 4A (ESC J)	1B 4A (ESC J)	1B 4A (ESC J)	1B 4A (ESC J)
<b>f5</b>	1B 74 (ESC t)	1B 74 (ESC t)	1B 74 (ESC t)	1B 74 (ESC t)	1B 4C (ESC L)	1B 4C (ESC L)	1B 4C (ESC L)	1B 4C (ESC L)
<b>f6</b>	1B 75 (ESC u)	1B 75 (ESC u)	1B 75 (ESC u)	1B 75 (ESC u)	1B 4D (ESC M)	1B 4D (ESC M)	1B 4D (ESC M)	1B 4D (ESC M)
<b>f7</b>	1B 76 (ESC v)	1B 76 (ESC v)	1B 76 (ESC v)	1B 76 (ESC v)	1B 51 (ESC Q)	1B 51 (ESC Q)	1B 51 (ESC Q)	1B 51 (ESC Q)
<b>f8</b>	1B 77 (ESC w)	1B 77 (ESC w)	1B 77 (ESC w)	1B 77 (ESC w)	1B 50 (ESC P)	1B 50 (ESC P)	1B 50 (ESC P)	1B 50 (ESC P)
<b>▲</b>	1B 41 (ESC A)	1B 53 (ESC S)	1B 41 (ESC A)	1B 41 (ESC A)	1B 56 (ESC V)	1B 56 (ESC V)	1B 56 (ESC V)	1B 56 (ESC V)
<b>▼</b>	1B 42 (ESC B)	1B 54 (ESC T)	1B 42 (ESC B)	1B 42 (ESC B)	1B 55 (ESC U)	1B 55 (ESC U)	1B 55 (ESC U)	1B 55 (ESC U)
<b>►</b>	1B 43 (ESC C)	1B 43 (ESC C)	1B 43 (ESC C)	1B 43 (ESC C)	1B 46 (ESC F)	1B 46 (ESC F)	1B 46 (ESC F)	1B 46 (ESC F)
<b>◀</b>	1B 44 (ESC D)	1B 44 (ESC D)	1B 44 (ESC D)	1B 44 (ESC D)	1B 68 (ESC h)	1B 68 (ESC h)	1B 68 (ESC h)	1B 68 (ESC h)
<b>Select</b>	*	*	*	*	(numpad)	(numpad)	(numpad)	(numpad)
<b>Stop</b>	13/11 (^S/^Q)	(break)	(C-break)	(reset)	13/11 (^S/^Q)	13/11 (^S/^Q)	13/11 (^S/^Q)	(reset) INT 19h
<b>Enter</b>	1B 64 (ESC d)	(print)	10 (^P)	(print)	1B 64 (ESC d)	(print)	10 (^P)	(print) INT 53h
<b>Menu</b>	(menu) INT 56h							

\* Generates three character codes: 1B 26 50 (ESC & P).

**Table 13-1. HP Mode Character Codes (Continued)**

English Keycap	Normal	Shift	CTRL	Shift- CTRL	Ext	Shift- Ext	Ext- CTRL	Shift- Ext- CTRL
<b>System</b>	(syst) INT 57h							
<b>ESC</b>	1B (ESC)	7F (DEL)	1B (ESC)	7F (DEL)	1B (ESC)	7F (DEL)	1B (ESC)	7F (DEL)
<b>► </b>	09 (^I)	1B 69 (ESC i)						
<b>Return</b>	0D (^M)							
<b>Backsp</b>	08 (^H)							

Table 13-2. Alternate Mode Character Codes

English Keycap	Normal	Shift	CTRL	Shift- CTRL	Ext	Shift- Ext	Ext- CTRL	Shift- Ext- CTRL
<b>f1</b>	00 3B	00 54	00 5E	00 5E	00 68	00 68	00 68	00 68
<b>f2</b>	00 3C	00 55	00 5F	00 5F	00 69	00 69	00 69	00 69
<b>f3</b>	00 3D	00 56	00 60	00 60	00 75	00 6A	00 6A	00 6A
<b>f4</b>	00 3E	00 57	00 61	00 61	00 76	00 6B	00 6B	00 6B
<b>f5</b>	00 3F	00 58	00 62	00 62		00 6C	00 6C	00 6C
<b>f6</b>	00 40	00 59	00 63	00 63		00 6D	00 6D	00 6D
<b>f7</b>	00 41	00 5A	00 64	00 64	00 52	00 6E	00 6E	00 6E
<b>f8</b>	00 42	00 5B	00 65	00 65	00 53	00 6F	00 6F	00 6F
<b>▲</b>	00 48	(roll up)	00 49	00 49	00 49	00 49	00 84	00 84
<b>▼</b>	00 50	(roll dn)	00 51	00 51	00 51	00 51	00 76	00 76
<b>▶</b>	00 4D	00 4F	00 74	00 74	00 4F	00 4F	00 75	00 75
<b>◀</b>	00 4B	00 47	00 73	00 73	00 47	00 47	00 77	00 77
<b>Select</b>			(cursor)		(numpad)	(numpad)	(numpad)	(numpad)
<b>Stop</b>	13/11	13/11	(C-break)	(reset)				(reset)
<b>Enter</b>			(print)	10				
<b>Menu</b>	00 43	00 5C	00 66	00 66	00 70	00 70	00 70	00 70
<b>System</b>	00 44	00 5D	00 67	00 67	00 71	00 71	00 71	00 71
<b>ESC</b>	1B	7F	1B	1B	1B	1B	1B	1B
<b>►</b>	09	00 0F	00 0F	00 0F	09	09	09	09
<b>Return</b>	0D	0D	0D	0D	0D	0D	0D	0D
<b>Backsp</b>	08	08	10	10	08	08	08	08

Table 13-3. Common HP Mode Character Codes

Keycap Char	Normal	Shift	CTRL	Shift- CTRL	Ext	Shift- Ext	Ext- CTRL	Shift- Ext- CTRL
B	c 62	c 42	02	02	FC	FC		
C	c 63	c 43	03	03	c B5	c B4		
D	c 64	c 44	04	04	c E4	c E3		
E	c 65	c 45	05	05	c D7	c D3		
F	c 66	c 46	06	06	BE	BE		
G	c 67	c 47	07	07	BA	BA		
H	c 68	c 48	08	08	BC	BC		
I	c 69	c 49	09	09	m -- ~	AC		
J	c 6A	c 4A	0A	0A	24	24		
K	c 6B	c 4B	0B	0B	BF	BF		
L	c 6C	c 4C	0C	0C	BB	BB		
N	c 6E	c 4E	0E	0E	F9	F9		
O	c 6F	c 4F	0F	0F	c D6	c D2		
P	c 70	c 50	10	10	c F1	c F0		
R	c 72	c 52	12	12	m -- '	A8		
S	c 73	c 53	13	13	DE	DE		
T	c 74	c 54	14	14	m -- `	A9		
U	c 75	c 55	15	15	m -- ``	AB		
V	c 76	c 56	16	16	BD	BD		
X	c 78	c 58	18	18	c EC	c EB		
space	20	20	20	20	20	20	20	20

c Affected by **Caps** key.m Mute character (as shown). Character code generated *next* keystroke.

Table 13-4. Common Alternate Mode Character Codes

Keycap Char	Normal	Shift	CTRL	Shift- CTRL	Ext	Shift- Ext	Ext- CTRL	Shift- Ext- CTRL
B	c 62	c 42	02	02	00 30	00 30	00 30	00 30
C	c 63	c 43	03	03	00 2E	00 2E	00 2E	00 2E
D	c 64	c 44	04	04	00 20	00 20	00 20	00 20
E	c 65	c 45	05	05	00 12	00 12	00 12	00 12
F	c 66	c 46	06	06	00 21	00 21	00 21	00 21
G	c 67	c 47	07	07	00 22	00 22	00 22	00 22
H	c 68	c 48	08	08	00 23	00 23	00 23	00 23
I	c 69	c 49	09	09	00 17	00 17	00 17	00 17
J	c 6A	c 4A	0A	0A	00 24	00 24	00 24	00 24
K	c 6B	c 4B	0B	0B	00 25	00 25	00 25	00 25
L	c 6C	c 4C	0C	0C	00 26	00 26	00 26	00 26
N	c 6E	c 4E	0E	0E	00 31	00 31	00 31	00 31
O	c 6F	c 4F	0F	0F	00 18	00 18	00 18	00 18
P	c 70	c 50	10	10	00 19	00 19	00 19	00 19
R	c 72	c 52	12	12	00 13	00 13	00 13	00 13
S	c 73	c 53	13	13	00 1F	00 1F	00 1F	00 1F
T	c 74	c 54	14	14	00 14	00 14	00 14	00 14
U	c 75	c 55	15	15	00 16	00 16	00 16	00 16
V	c 76	c 56	16	16	00 2F	00 2F	00 2F	00 2F
X	c 78	c 58	18	18	00 2D	00 2D	00 2D	00 2D
space	20	20	20	20	20	20	20	20

c      Affected by **(Caps)** key.

**Figure 13–1. English (U.S.) Keyboard**

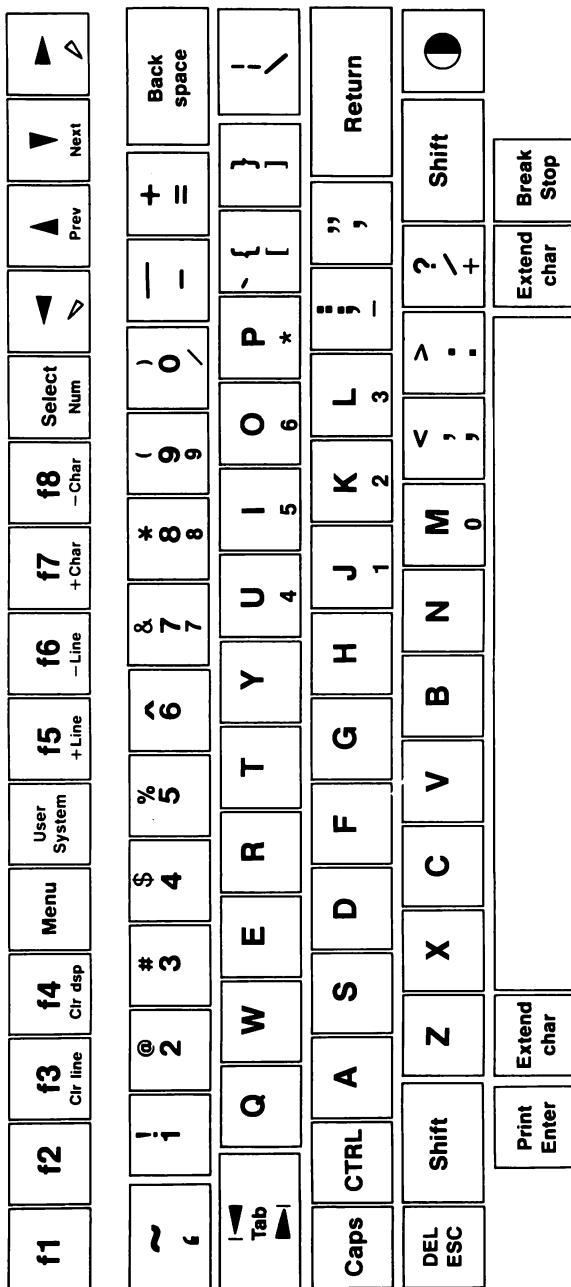


Table 13-5. English (U.S.) HP Mode Character Codes

Keycap Char	Normal	Shift	CTRL	Shift- CTRL	Ext	Shift- Ext	Ext- CTRL	Shift- Ext- CTRL
' ~	60	7E			FB	FD		
1 !	31	21			B8	B8		
2 @	32	40	00	00	40	40	00	00
3 #	33	23			23	23		
4 \$	34	24			F7	F7		
5 %	35	25			F8	F8		
6 ^	36	5E	1E	1E	5E	5E	1E	1E
7 &	37	26			5C	5C	1C	1C
8 *	38	2A			5B	7B	1B	1B
9 (	39	28			5D	7D	1D	1D
0 )	30	29			B9	B9		
- _	2D	5F	1F	1F	F6	B0		
= +	3D	2B			FE	FE		
Q c	71	c 51	11	11				
W c	77	c 57	17	17	7E	7E		
Y c	79	c 59	19	19	m -- ^	AA		
[ {	5B	7B	1B	1B	B3	B3		
] }	5D	7D	1D	1D	7C	7C		
\	5C	7C	1C	1C				
A c	61	c 41	01	01	c D4	c D0		
; :	3B	3A			AF	AF		
;" "	27	22			60	27		
Z c	7A	c 5A	1A	1A				
M c	6D	c 4D	0D	0D	FA	FA		
, <	2C	3C			3C	3C		
. >	2E	3E			3E	3E		
/ ?	2F	3F			5F	5F	1F	1F

c Affected by Caps key.

m Mute character (as shown). Character code generated next keystroke.

**Table 13-6. English (U.S.) Alternate Mode Character Codes**

Keycap Char	Normal	Shift	CTRL	Shift- CTRL	Ext	Shift- Ext	Ext- CTRL	Shift- Ext- CTRL
' ~	60	7E						
1 !	31	21			00 78	00 78	00 78	00 78
2 @	32	40	00 03	00 03	00 79	00 79	00 79	00 79
3 #	33	23			00 7A	00 7A	00 7A	00 7A
4 \$	34	24			00 7B	00 7B	00 7B	00 7B
5 %	35	25			00 7C	00 7C	00 7C	00 7C
6 ^	36	5E	1E	1E	00 7D	00 7D	00 7D	00 7D
7 &	37	26			00 7E	00 7E	00 7E	00 7E
8 *	38	2A			00 7F	00 7F	00 7F	00 7F
9 (	39	28			00 80	00 80	00 80	00 80
0 )	30	29			00 81	00 81	00 81	00 81
- _	2D	5F	1F	1F	00 82	00 82	00 82	00 82
= +	3D	2B			00 83	00 83	00 83	00 83
Q c	71	c 51	11	11	00 10	00 10	00 10	00 10
W c	77	c 57	17	17	00 11	00 11	00 11	00 11
Y c	79	c 59	19	19	00 15	00 15	00 15	00 15
[ {	5B	7B	1B	1B				
] }	5D	7D	1D	1D				
\	5C	7C	1C	1C				
A c	61	c 41	01	01	00 1E	00 1E	00 1E	00 1E
; :	3B	3A						
, "	27	22						
Z c	7A	c 5A	1A	1A	00 2C	00 2C	00 2C	00 2C
M c	6D	c 4D	0D	0D	00 32	00 32	00 32	00 32
, <	2C	3C						
. >	2E	3E						
/ ?	2F	3F						

c Affected by the **Caps** key.

Figure 13-2. English (U.K.) Keyboard

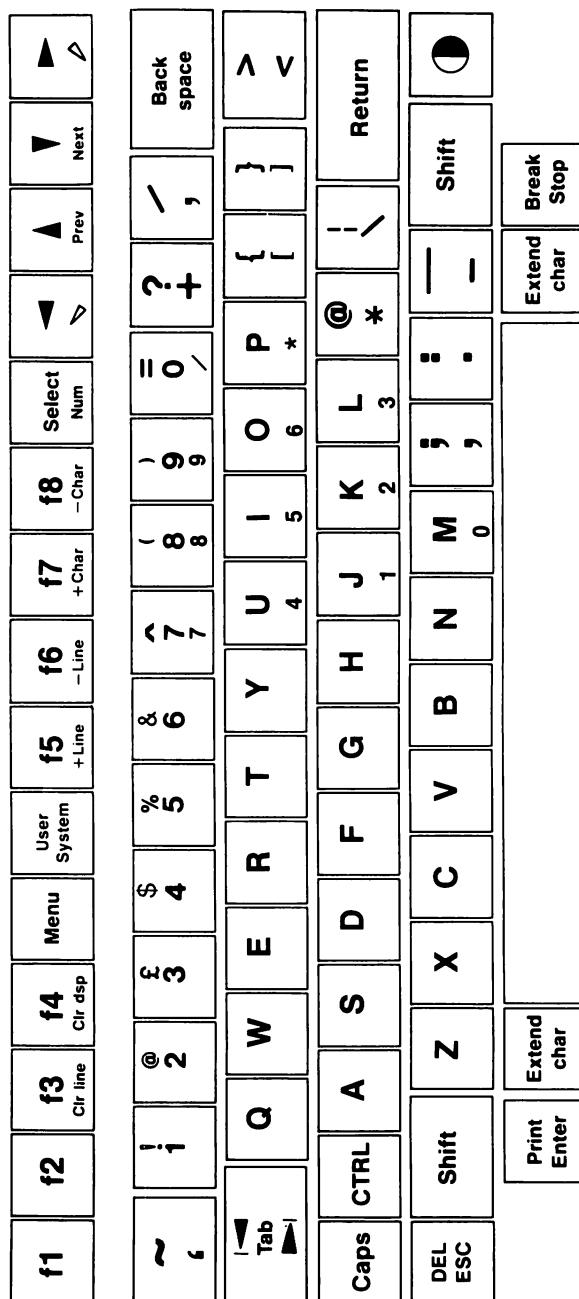


Table 13-7. English (U.K.) HP Mode Character Codes

Keycap Char	Normal	Shift	CTRL	Shift- CTRL	Ext	Shift- Ext	Ext- CTRL	Shift- Ext- CTRL
' ~	60	7E			FB	FD		
1 !	31	21			B8	B8		
2 "	32	22			40	40	00	00
3 £	33	BB			23	23		
4 \$	34	24			F7	F7		
5 %	35	25			F8	F8		
6 &	36	26			5E	5E	1E	1E
7 ^	37	5E	1E	1E	5C	5C	1C	1C
8 (	38	28			5B	7B	1B	1B
9 )	39	29			5D	7D	1D	1D
0 =	30	3D			B9	B9		
+ ?	2B	3F			F6	B0		
,	/	27	2F		FE	FE		
Q c	71	c 51	11	11				
W c	77	c 57	17	17	7E	7E		
Y c	79	c 59	19	19	m -- ^	AA		
[ {	5B	7B	1B	1B	B3	B3		
] }	5D	7D	1D	1D	7C	7C		
< >	3C	3E						
A c	61	c 41	01	01	c D4	c D0		
* @	2A	40	00	00	AF	AF		
\	5C	7C	1C	1C	60	27		
Z c	7A	c 5A	1A	1A				
M c	6D	c 4D	0D	0D	FA	FA		
, ;	2C	3B			3C	3C		
. :	2E	3A			3E	3E		
- _	2D	5F	1F	1F	5F	5F	1F	1F

c Affected by Caps key.

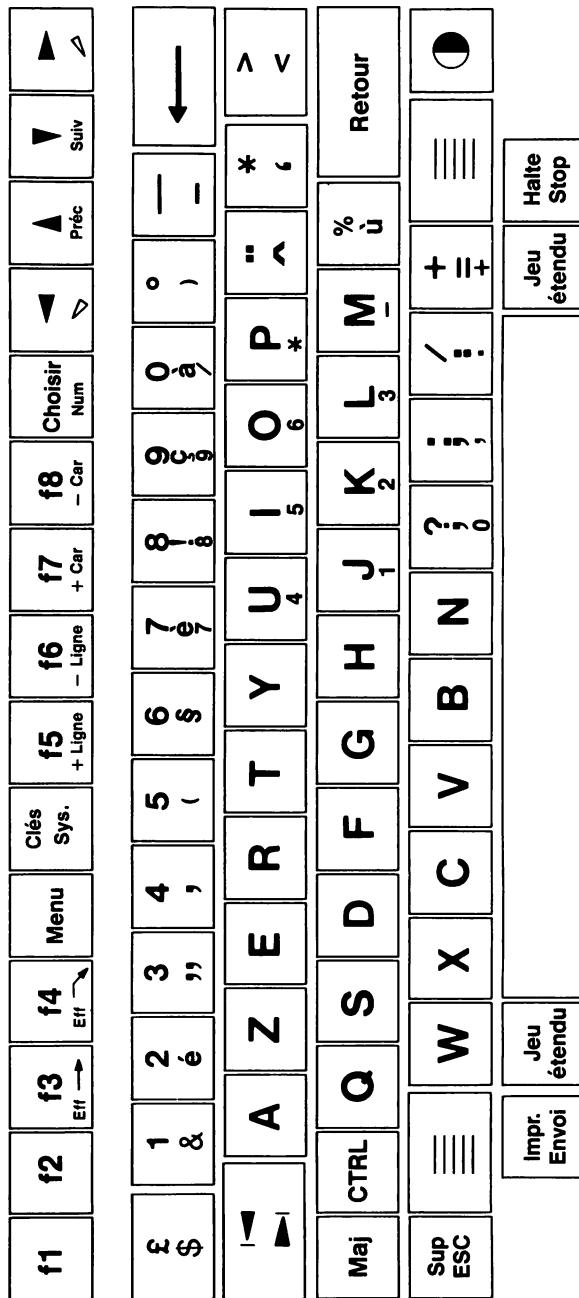
m Mute character (as shown). Character code generated next keystroke.

Table 13-8. English (U.K.) Alternate Mode Character Codes

Keycap Char	Normal	Shift	CTRL	Shift-		Shift-		Ext- CTRL	Ext- CTRL	Shift-
				CTRL	Ext	Ext	Ext			Ext
' ~	60	7E								
1 !	31	21			00 78	00 78	00 78	00 78	00 78	
2 "	32	22			00 79	00 79	00 79	00 79	00 79	
3 £	33	9C			00 7A	00 7A	23	23		
4 \$	34	24			00 7B	00 7B	00 7B	00 7B	00 7B	
5 %	35	25			00 7C	00 7C	00 7C	00 7C	00 7C	
6 &	36	26			00 7D	00 7D	00 7D	00 7D	00 7D	
7 ^	37	5E	1E	1E	00 7E	00 7E	00 7E	00 7E	00 7E	
8 (	38	28			00 7F	00 7F	00 7F	00 7F	00 7F	
9 )	39	29			00 80	00 80	00 80	00 80	00 80	
0 =	30	3D			00 81	00 81	00 81	00 81	00 81	
+	?	2B	3F		00 82	00 82	00 82	00 82	00 82	
,	/	27	2F		00 83	00 83	00 83	00 83	00 83	
Q c	71	c 51	11	11	00 10	00 10	00 10	00 10	00 10	
W c	77	c 57	17	17	00 11	00 11	00 11	00 11	00 11	
Y c	79	c 59	19	19	00 15	00 15	00 15	00 15	00 15	
[ {	5B	7B	1B	1B						
] }	5D	7D	1D	1D						
< >	3C	3E								
A c	61	c 41	01	01	00 1E	00 1E	00 1E	00 1E	00 1E	
* @	2A	40	00 03	00 03						
\	5C	7C	1C	1C						
Z c	7A	c 5A	1A	1A	00 2C	00 2C	00 2C	00 2C	00 2C	
M c	6D	c 4D	0D	0D	00 32	00 32	00 32	00 32	00 32	
,	;	2C	3B							
.	:	2E	3A							
- _	2D	5F	1F	1F						

c Affected by the **(Caps)** key.

**Figure 13-3. French Keyboard**



**Figure 13-4. Belgian Keyboard**

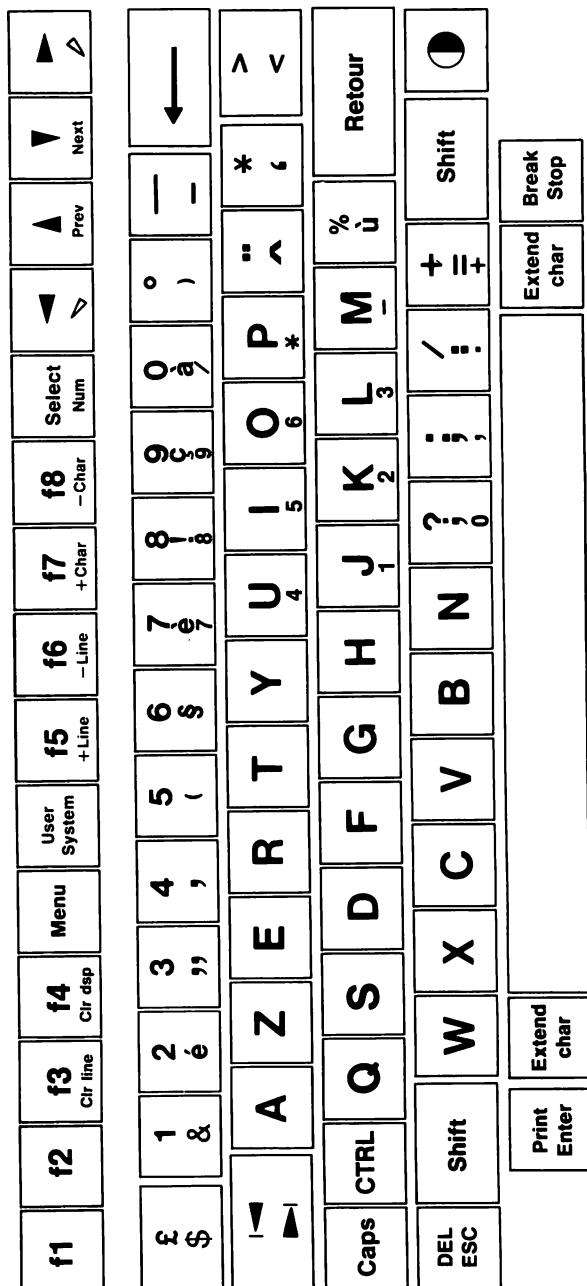


Table 13-9. French/Belgian HP Mode Character Codes

Keycap				Shift-		Shift-	Ext-	Shift-
Char	Normal	Shift	CTRL	CTRL	Ext	Ext	CTRL	Ext- CTRL
\$	£	c 24	c BB		c FB	c FD		
&	1	c 26	c 31		B8	B8		
é	2	c C5	c 32		40	40	00	00
"	3	c 22	c 33		23	23		
,	4	c 27	c 34		F7	F7		
(	5	c 28	c 35		F8	F8		
§	6	c BD	c 36		5E	5E	1E	1E
è	7	c C9	c 37		5C	5C	1C	1C
!	8	c 21	c 38		c 5B	c 7B	1B	1B
ç	9	c B5	c 39		c 5D	c 7D	1D	1D
à	0	c C8	c 30		B9	B9		
)	º	c 29	c B3		c F6	c B0		
-	-	c 2D	c 5F	1F	1F	FE	FE	
A	-	c 61	c 41	01	01			
Z		c 7A	c 5A	1A	1A	7E	7E	
Y		c 79	c 59	19	19	m -- ^	AA	
^	..	mc--	^	mc--	..	B3	B3	
`	*	c 60	c 2A			7C	7C	
<	>	c 3C	c 3E					
Q		c 71	c 51	11	11	c D4	c D0	
M		c 6D	c 4D	0D	0D	AF	AF	
ù	%	c CB	c 25			c 60	c 27	
W		c 77	c 57	17	17			
,	?	c 2C	c 3F			FA	FA	
;	.	c 3B	c 2E			3C	3C	
:	/	c 3A	c 2F			3E	3E	
=	+	c 3D	c 2B			5F	5F	1F

c Affected by **(Caps)** key.

m Mute character (as shown). Character code generated *next* keystroke.

Table 13-10. French/Belgian Alternate Mode Character Codes

Keycap Char	Normal	Shift	CTRL	Shift- CTRL	Ext	Shift- Ext	Ext- CTRL	Shift- Ext- CTRL
\$	c 24	c 9C						
&	c 26	c 31			00 78	00 78	00 78	00 78
é	c 82	c 32	00 03	00 03	00 79	00 79	40	40
"	c 22	c 33			00 7A	00 7A	23	23
,	c 27	c 34			00 7B	00 7B	00 7B	00 7B
(	c 28	c 35			00 7C	00 7C	00 7C	00 7C
\$	c 15	c 36	1E	1E	00 7D	00 7D	5E	5E
è	c 8A	c 37	1C	1C	00 7E	00 7E	5C	5C
!	c 21	c 38	1B	1B	00 7F	00 7F	5B	7B
ç	c 87	c 39	1D	1D	00 80	00 80	5D	7D
à	c 85	c 30			00 81	00 81	00 81	00 81
)	c 29	c F8			00 82	00 82	00 82	00 82
-	c 2D	c 5F	1F	1F	00 83	00 83	00 83	00 83
A	c 61	c 41	01	01	00 1E	00 1E	00 1E	00 1E
Z	c 7A	c 5A	1A	1A	00 2C	00 2C	7E	7E
Y	c 79	c 59	19	19	00 15	00 15	00 15	00 15
^ ..	mc--	^	mc-- ..					
` *	c 60	c 2A					7C	7C
< >	c 3C	c 3E						
Q	c 71	c 51	11	11	00 10	00 10	00 10	00 10
M	c 6D	c 4D	0D	0D	00 32	00 32	00 32	00 32
ù %	c 97	c 25						
W	c 77	c 57	17	17	00 11	00 11	00 11	00 11
, ?	c 2C	c 3F						
;	c 3B	c 2E						
:	/	c 3A	c 2F					
= +	c 3D	c 2B						

c Affected by **Caps** key.

m Mute character (as shown). Character code generated *next* keystroke.

Figure 13–5. German Keyboard

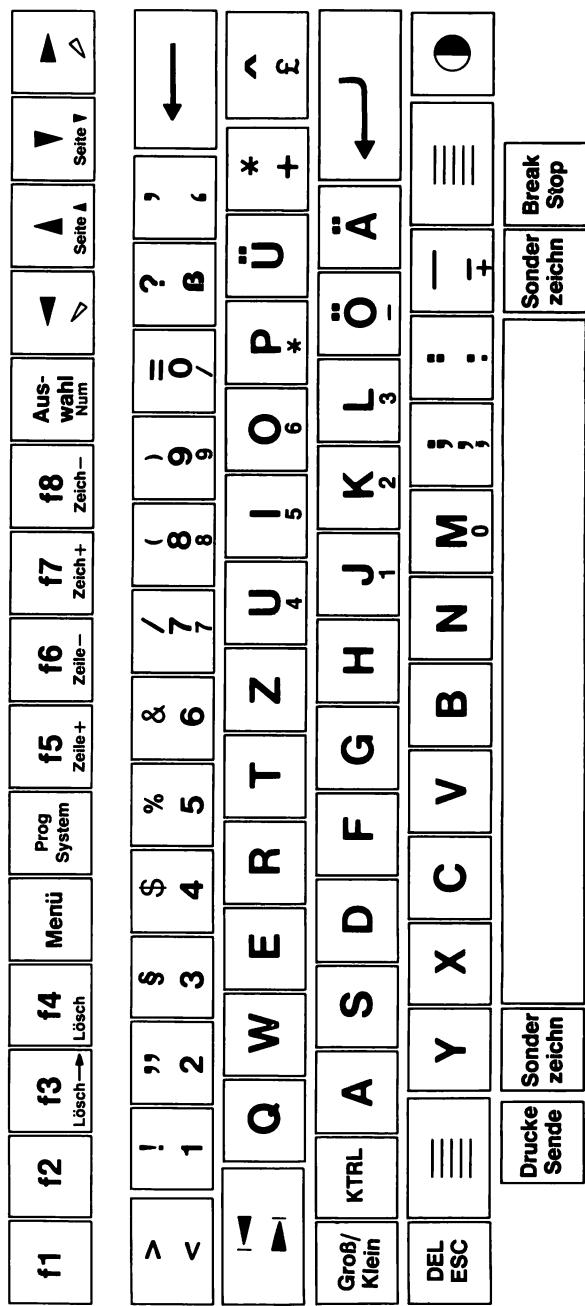


Table 13-11. German HP Mode Character Codes

Keycap Char	Normal	Shift	CTRL	Shift- CTRL	Ext	Shift- Ext	Ext- CTRL	Shift- Ext- CTRL
<	3C	3E			FB	FD		
1 !	31	21			B8	B8		
2 "	32	22			40	40	00	00
3 \$	33	BD			23	23		
4 \$	34	24			F7	F7		
5 %	35	25			F8	F8		
6 &	36	26			5E	5E	1E	1E
7 /	37	2F			5C	5C	1C	1C
8 (	38	28			5B	7B	1B	1B
9 )	39	29			5D	7D	1D	1D
0 =	30	3D			B9	B9		
ß ?	DE	3F			F6	B0		
,	27	60			FE	FE		
Q c	71	c 51	11	11				
W c	77	c 57	17	17	7E	7E		
Z c	7A	c 5A	1A	1A	m -- ^	AA		
Ü c	CF	c DB			B3	B3		
+	2B	2A			7C	7C		
£ ^	BB	5E	1E	1E				
A c	61	c 41	01	01	c D4	c D0		
Ö c	CE	c DA			AF	AF		
Ä c	CC	c D8			60	27		
Y c	79	c 59	19	19				
M c	6D	c 4D	0D	0D	FA	FA		
,	2C	3B			3C	3C		
.	2E	3A			3E	3E		
- _	2D	5F	1F	1F	5F	5F	1F	1F

c Affected by **(Caps)** key.

m Mute character (as shown). Character code generated *next* keystroke.

Table 13-12. German Alternate Mode Character Codes

Keycap Char	Normal	Shift	CTRL	Shift- CTRL	Ext	Shift- Ext	Ext- CTRL	Shift- Ext- CTRL
<	3C	3E						
1 !	31	21			00 78	00 78	00 78	00 78
2 "	32	22	00 03	00 03	00 79	00 79	40	40
3 §	33	15			00 7A	00 7A	23	23
4 \$	34	24			00 7B	00 7B	00 7B	00 7B
5 %	35	25			00 7C	00 7C	00 7C	00 7C
6 &	36	26			00 7D	00 7D	00 7D	00 7D
7 /	37	2F	1C	1C	00 7E	00 7E	5C	5C
8 (	38	28	1B	1B	00 7F	00 7F	5B	7B
9 )	39	29	1D	1D	00 80	00 80	5D	7D
0 =	30	3D			00 81	00 81	00 81	00 81
ß ?	E1	3F			00 82	00 82	00 82	00 82
,	27	60			00 83	00 83	00 83	00 83
Q c	71	c 51	11	11	00 10	00 10	00 10	00 10
W c	77	c 57	17	17	00 11	00 11	7E	7E
Z c	7A	c 5A	1A	1A	00 2C	00 2C	00 2C	00 2C
Ü c	81	c 9A						
+	*	2B	2A				7C	7C
£ ^	9C	5E	1E	1E				
A c	61	c 41	01	01	00 1E	00 1E	00 1E	00 1E
Ö c	94	c 99						
Ä c	84	c 8E						
Y c	79	c 59	19	19	00 15	00 15	00 15	00 15
M c	6D	c 4D	0D	0D	00 32	00 32	00 32	00 32
,	;	2C	3B					
.	:	2E	3A					
- _	2D	5F	1F	1F				

c Affected by the **Caps** key.

Figure 13-6. Italian Keyboard

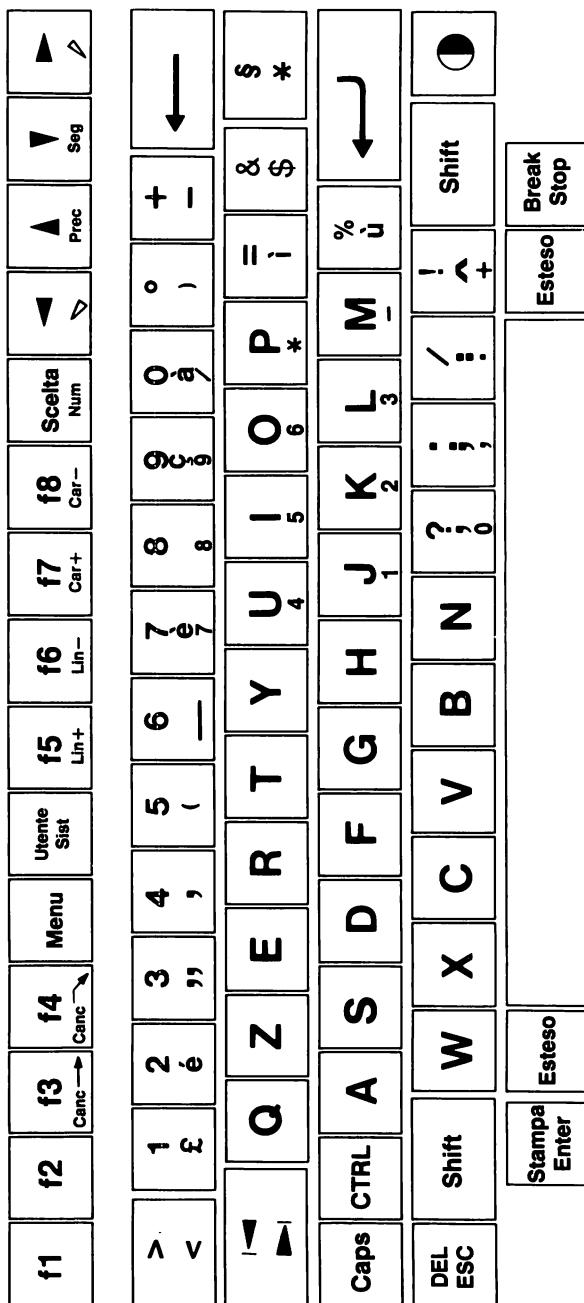


Table 13-13. Italian HP Mode Character Codes

Keycap	Char	Normal	Shift	CTRL	Shift- CTRL	Ext	Shift- Ext	Ext- CTRL	Shift- Ext- CTRL
<	>	c 3C	c 3E			c FB	c FD		
£	1	c BB	c 31			B8	B8		
é	2	c C5	c 32			40	40	00	00
"	3	c 22	c 33			23	23		
,	4	c 27	c 34			F7	F7		
(	5	c 28	c 35			F8	F8		
)	6	c 5F	c 36	1F	1F	5E	5E	1E	1E
é	7	c C9	c 37			5C	5C	1C	1C
^	8	c 5E	c 38	1E	1E	c 5B	c 7B	1B	1B
ç	9	c B5	c 39			c 5D	c 7D	1D	1D
à	0	c C8	c 30			B9	B9		
)	°	c 29	c B3			c F6	c B0		
-	+	c 2D	c 2B			FE	FE		
Q		c 71	c 51	11	11				
Z		c 7A	c 5A	1A	1A	7E	7E		
Y		c 79	c 59	19	19	m -- ^	AA		
í	=	c D9	c 3D			B3	B3		
\$	&	c 24	c 26			7C	7C		
*	§	c 2A	c BD						
A		c 61	c 41	01	01	c D4	c D0		
M		c 6D	c 4D	0D	0D	AF	AF		
ù	%	c CB	c 25			c 60	c 27		
W		c 77	c 57	17	17				
,	?	c 2C	c 3F			FA	FA		
;	.	c 3B	c 2E			3C	3C		
:	/	c 3A	c 2F			3E	3E		
ò	!	c CA	c 21			5F	5F	1F	1F

c Affected by **Caps** key.

m Mute character (as shown). Character code generated *next* keystroke.

Table 13-14. Italian Alternate Mode Character Codes

Keycap Char	Normal	Shift	CTRL	Shift- CTRL	Ext	Shift- Ext	Ext- CTRL	Shift- Ext- CTRL
<	c 3C	c 3E						
£	c 9C	c 31			00 78	00 78	00 78	00 78
é	c 82	c 32	00 03	00 03	00 79	00 79	40	40
"	c 22	c 33			00 7A	00 7A	23	23
,	c 27	c 34			00 7B	00 7B	00 7B	00 7B
(	c 28	c 35			00 7C	00 7C	00 7C	00 7C
)	c 5F	c 36	1F	1F	00 7D	00 7D	00 7D	00 7D
é	c 8A	c 37	1C	1C	00 7E	00 7E	5C	5C
^	c 5E	c 38	1E	1B	00 7F	00 7F	5B	7B
ç	c 87	c 39	1D	1D	00 80	00 80	5D	7D
à	c 85	c 30			00 81	00 81	00 81	00 81
)	c 29	c F8			00 82	00 82	00 82	00 82
-	c 2D	c 2B			00 83	00 83	00 83	00 83
Q	c 71	c 51	11	11	00 10	00 10	00 10	00 10
Z	c 7A	c 5A	1A	1A	00 2C	00 2C	7E	7E
Y	c 79	c 59	19	19	00 15	00 15	00 15	00 15
í	=	c 8D	c 3D					
\$	&	c 24	c 26				7C	7C
*	§	c 2A	c 15					
A	c 61	c 41	01	01	00 1E	00 1E	00 1E	00 1E
M	c 6D	c 4D	0D	0D	00 32	00 32	00 32	00 32
ù	%	c 97	c 25				60	60
W	c 77	c 57	17	17	00 11	00 11	00 11	00 11
,	?	c 2C	c 3F					
;	.	c 3B	c 2E					
:	/	c 3A	c 2F					
ò	!	c 95	c 21					

c Affected by **Caps** key.

**Figure 13-7. Dutch Keyboard**

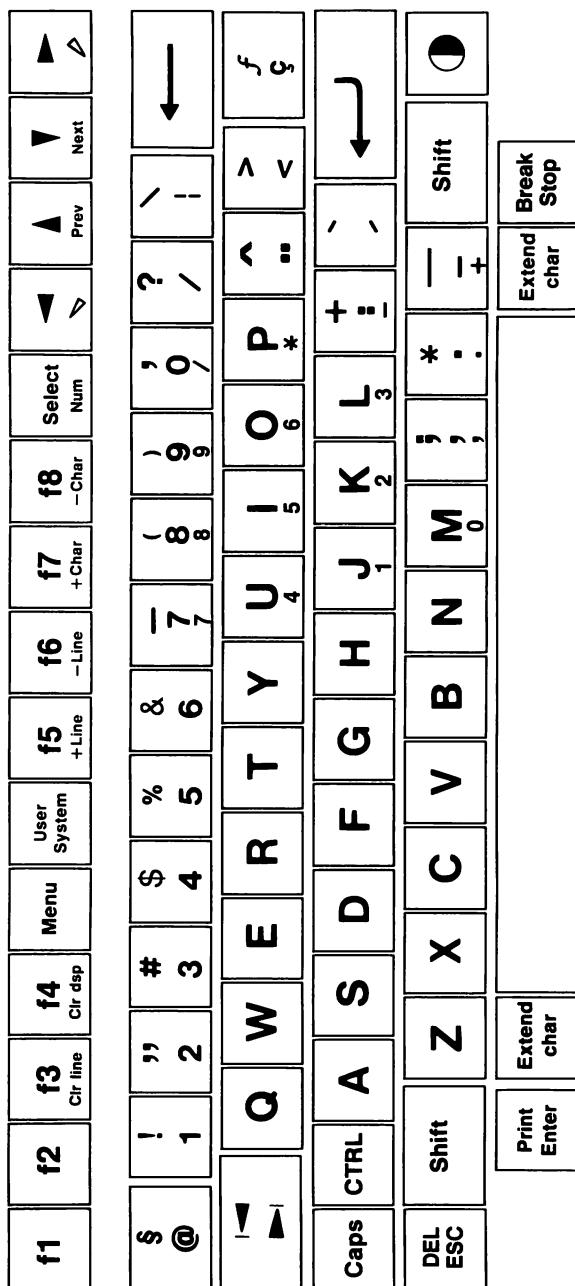


Table 13-15. Dutch HP Mode Character Codes

Keycap	Char	Normal	Shift	CTRL	Shift- CTRL	Ext	Shift- Ext	Ext- CTRL	Shift- Ext- CTRL
@	§	40	BD	00	00	FB	FD		
1	!	31	21			B8	B8		
2	"	32	22			40	40	00	00
3	#	33	23			23	23		
4	\$	34	24			F7	F7		
5	%	35	25			F8	F8		
6	&	36	26			5E	5E	1E	1E
7	-	37	5F	1F	1F	5C	5C	1C	1C
8	(	38	28			5B	7B	1B	1B
9	)	39	29			5D	7D	1D	1D
0	,	30	27			B9	B9		
/	?	2F	3F			F6	B0		
	\	7C	5C	1C	1C	FE	FE		
Q	c	71	c	51	11	11			
W	c	77	c	57	17	17	7E	7E	
Y	c	79	c	59	19	19	m -- ^	AA	
..	^	m -- ..	m -- ^				B3	B3	
<	>	3C	3E				7C	7C	
ç	f	B5	BE						
A	c	61	c	41	01	01	c D4	c D0	
:	+	3A	2B				AF	AF	
,	`	m -- ,	m -- `				60	27	
Z	c	7A	c	5A	1A	1A			
M	c	6D	c	4D	0D	0D	FA	FA	
,	;	2C	3B				3C	3C	
.	*	2E	2A				3E	3E	
-	=	2D	3D				5F	5F	1F
									1F

c Affected by Caps key.

m Mute character (as shown). Character code generated next keystroke.

Table 13-16. Dutch Alternate Mode Character Codes

Keycap Char	Normal	Shift	CTRL	Shift- CTRL	Ext	Shift- Ext	Ext- CTRL	Shift- Ext- CTRL
@	§	40	15	00 03	00 03			
1	!	31	21		00 78	00 78	00 78	00 78
2	"	32	22		00 79	00 79	00 79	00 79
3	#	33	23		00 7A	00 7A	00 7A	00 7A
4	\$	34	24		00 7B	00 7B	00 7B	00 7B
5	%	35	25		00 7C	00 7C	00 7C	00 7C
6	&	36	26	1E	1E	00 7D	00 7D	5E 5E
7	-	37	5F	1F	1F	00 7E	00 7E	00 7E
8	(	38	28	1B	1B	00 7F	00 7F	5B 7B
9	)	39	29	1D	1D	00 80	00 80	5D 7D
0	,	30	27		00 81	00 81	00 81	00 81
/	?	2F	3F		00 82	00 82	00 82	00 82
	\	7C	5C	1C	1C	00 83	00 83	00 83
Q	c	71	c 51	11	11	00 10	00 10	00 10
W	c	77	c 57	17	17	00 11	00 11	7E 7E
Y	c	79	c 59	19	19	00 15	00 15	00 15
..	^	m --	m -- ^					
<	>	3C	3E					
ç	f	87	9F					
A	c	61	c 41	01	01	00 1E	00 1E	00 1E
:	+	3A	2B					
;	,	m --	m -- `				60	60
Z	c	7A	c 5A	1A	1A	00 2C	00 2C	00 2C
M	c	6D	c 4D	0D	0D	00 32	00 32	00 32
,	;	2C	3B					
.	*	2E	2A					
-	=	2D	3D					

c Affected by the Caps key.

m Mute character (as shown). Character code generated *next* keystroke.

**Figure 13-8. Swiss (German) Keyboard**

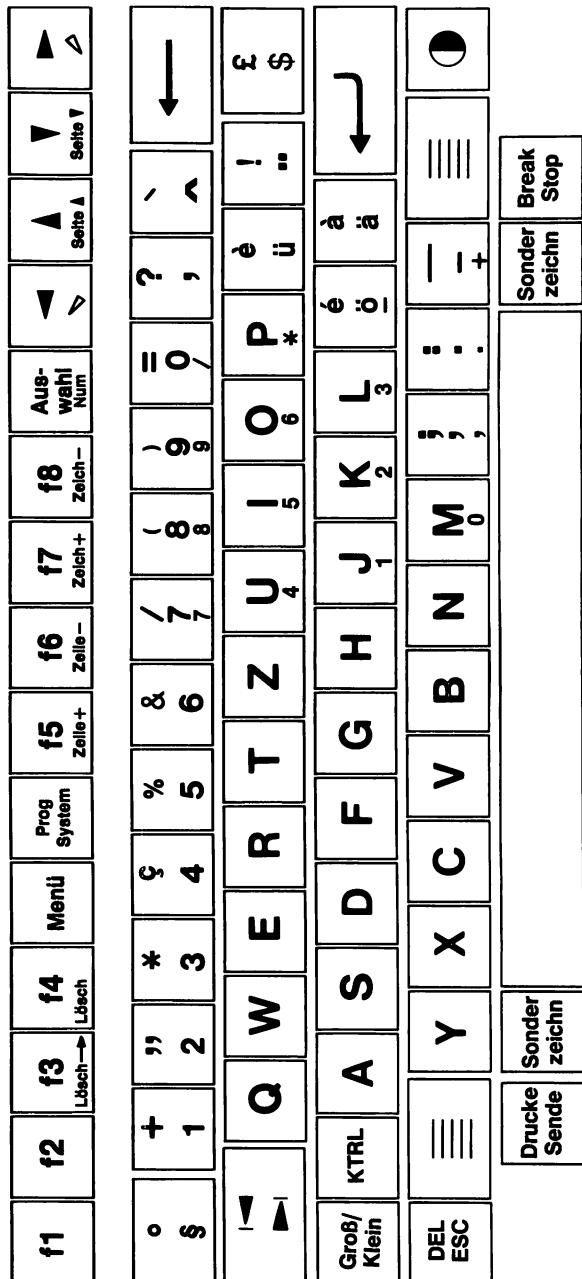


Table 13-17. Swiss (German) HP Mode Character Codes

Keycap Char	Normal	Shift	CTRL	Shift- CTRL	Ext	Shift- Ext	Ext- CTRL	Shift- Ext- CTRL
\$ ¦	BD	B3			FB	FD		
1 +	31	2B			B8	B8		
2 "	32	22			40	40	00	00
3 *	33	2A			23	23		
4 ¢	34	B5			F7	F7		
5 %	35	25			F8	F8		
6 &	36	26			5E	5E	1E	1E
7 /	37	2F			5C	5C	1C	1C
8 (	38	28			5B	7B	1B	1B
9 )	39	29			5D	7D	1D	1D
0 =	30	3D			B9	B9		
' ?	27	3F			F6	B0		
^ `	m -- ^	m -- `			FE	FE		
Q c	71	c 51	11	11				
W c	77	c 57	17	17	7E	7E		
Z c	7A	c 5A	1A	1A	m -- ^	AA		
ü è	CF	C9			B3	B3		
“ !	m -- “	21			7C	7C		
\$ £	24	BB						
A c	61	c 41	01	01	c D4	c D0		
ö é	CE	C5			AF	AF		
ä à	CC	C8			60	27		
Y c	79	c 59	19	19				
M c	6D	c 4D	0D	0D	FA	FA		
, ;	2C	3B			3C	3C		
. :	2E	3A			3E	3E		
- _	2D	5F	1F	1F	5F	5F	1F	1F

c Affected by Caps key.

m Mute character (as shown). Character code generated next keystroke.

Table 13-18. Swiss (German) Alternate Mode Character Codes

Keycap Char	Normal	Shift	CTRL	Shift-		Shift-		Ext- CTRL	Ext- CTRL
				CTRL	Ext	Ext	CTRL		
§	°	15	F8						
1	+	31	2B		00 03	00 03	00 78	00 78	00 78
2	"	32	22	00 03	00 03	00 79	00 79	40	40
3	*	33	2A			00 7A	00 7A	23	23
4	¢	34	87			00 7B	00 7B	00 7B	00 7B
5	%	35	25			00 7C	00 7C	00 7C	00 7C
6	&	36	26	1E	1E	00 7D	00 7D	5E	5E
7	/	37	2F	1C	1C	00 7E	00 7E	5C	5C
8	(	38	28	1B	1B	00 7F	00 7F	5B	7B
9	)	39	29	1D	1D	00 80	00 80	5D	7D
0	=	30	3D			00 81	00 81	00 81	00 81
,	?	27	3F			00 82	00 82	00 82	00 82
^	~	m -- ^	m -- ~			00 83	00 83	00 83	00 83
Q	c	71	c 51	11	11	00 10	00 10	00 10	00 10
W	c	77	c 57	17	17	00 11	00 11	7E	7E
Z	c	7A	c 5A	1A	1A	00 2C	00 2C	00 2C	00 2C
ü	è	81	8A						
”	!	m -- ”	21					7C	7C
\$	£	24	9C						
A	c	61	c 41	01	01	00 1E	00 1E	00 1E	00 1E
ö	é	94	82						
ä	à	84	85					60	60
Y	c	79	c 59	19	19	00 15	00 15	00 15	00 15
M	c	6D	c 4D	0D	0D	00 32	00 32	00 32	00 32
,	;	2C	3B					3C	3C
.	:	2E	3A					3E	3E
-	_	2D	5F	1F	1F				

c Affected by the **Caps** key.

m Mute character (as shown). Character code generated *next* keystroke.

Figure 13-9. Swiss (French) Keyboard

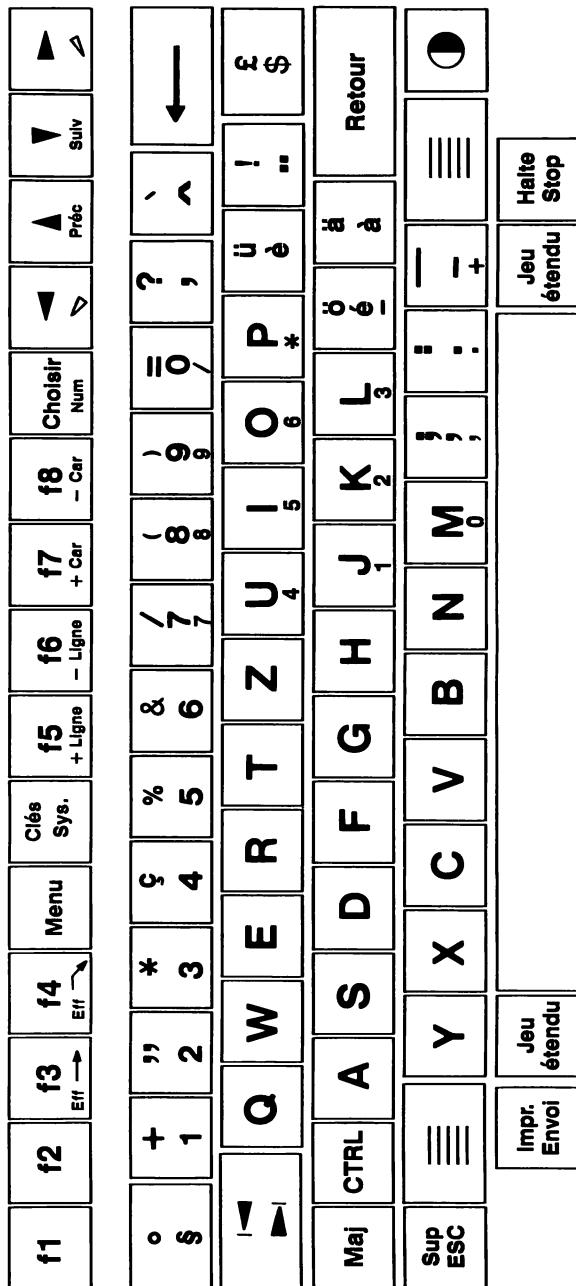


Table 13-19. Swiss (French) HP Mode Character Codes

Keycap Char	Normal	Shift	CTRL	Shift- CTRL	Ext	Shift- Ext	Ext- CTRL	Shift- Ext- CTRL
§ º	BD	B3			FB	FD		
1 +	31	2B			B8	B8		
2 "	32	22			40	40	00	00
3 *	33	2A			23	23		
4 ¢	34	B5			F7	F7		
5 %	35	25			F8	F8		
6 &	36	26			5E	5E	1E	1E
7 /	37	2F			5C	5C	1C	1C
8 (	38	28			5B	7B	1B	1B
9 )	39	29			5D	7D	1D	1D
0 =	30	3D			B9	B9		
' ?	27	3F			F6	B0		
^ `	m -- ^	m -- `			FE	FE		
Q c	71	c 51	11	11				
W c	77	c 57	17	17	7E	7E		
Z c	7A	c 5A	1A	1A	m -- ^	AA		
è ü	C9	CF			B3	B3		
.. !	m -- ..	21			7C	7C		
\$ £	24	BB						
A c	61	c 41	01	01	c D4	c D0		
é ö	C5	CE			AF	AF		
à ä	C8	CC			60	27		
Y c	79	c 59	19	19				
M c	6D	c 4D	0D	0D	FA	FA		
, ;	2C	3B			3C	3C		
. :	2E	3A			3E	3E		
- _	2D	5F	1F	1F	5F	5F	1F	1F

c Affected by Caps key.

m Mute character (as shown). Character code generated next keystroke.

Table 13-20. Swiss (French) Alternate Mode Character Codes

Keycap Char	Normal	Shift	CTRL	Shift- CTRL	Ext	Shift- Ext	Ext- CTRL	Ext- CTRL
\$ °	15	F8						
1 +	31	2B			00 78	00 78	00 78	00 78
2 "	32	22	00 03	00 03	00 79	00 79	40	40
3 *	33	2A			00 7A	00 7A	23	23
4 ¢	34	87			00 7B	00 7B	00 7B	00 7B
5 %	35	25			00 7C	00 7C	00 7C	00 7C
6 &	36	26	1E	1E	00 7D	00 7D	5E	5E
7 /	37	2F	1C	1C	00 7E	00 7E	5C	5C
8 (	38	28	1B	1B	00 7F	00 7F	5B	7B
9 )	39	29	1D	1D	00 80	00 80	5D	7D
0 =	30	3D			00 81	00 81	00 81	00 81
' ?	27	3F			00 82	00 82	00 82	00 82
^ `	m -- ^	m -- `			00 83	00 83	00 83	00 83
Q c	71	c 51	11	11	00 10	00 10	00 10	00 10
W c	77	c 57	17	17	00 11	00 11	7E	7E
Z c	7A	c 5A	1A	1A	00 2C	00 2C	00 2C	00 2C
è ü	8A	81						
" !	m -- "	21					7C	7C
\$ &	24	9C						
A c	61	c 41	01	01	00 1E	00 1E	00 1E	00 1E
é ö	82	94						
à ä	85	84					60	60
Y c	79	c 59	19	19	00 15	00 15	00 15	00 15
M c	6D	c 4D	0D	0D	00 32	00 32	00 32	00 32
,	2C	3B					3C	3C
.	2E	3A					3E	3E
- _	2D	5F	1F	1F				

c Affected by the Caps key.

m Mute character (as shown). Character code generated next keystroke.

**Figure 13–10. Danish Keyboard**

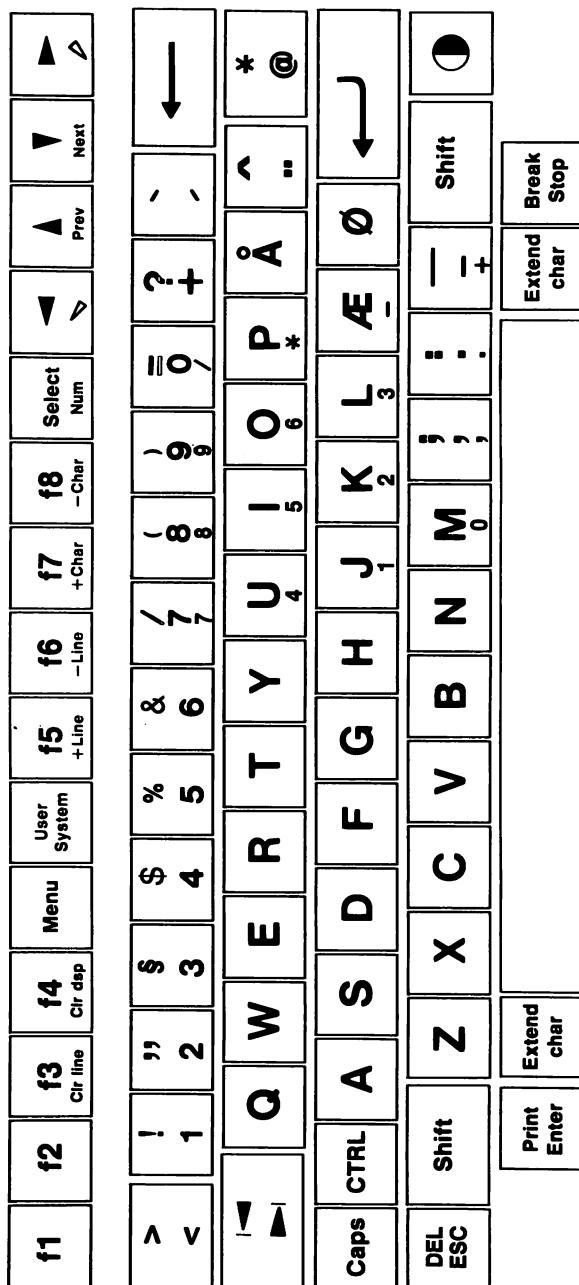


Table 13-21. Danish HP Mode Character Codes

Keycap Char	Normal	Shift	CTRL	Shift-		Shift-		Ext- CTRL	Shift- Ext- CTRL
				CTRL	Ext	Ext	CTRL		
<	>	3C	3E			FB	FD		
1	!	31	21			B8	B8		
2	"	32	22			40	40	00	00
3	\$	33	BD			23	23		
4	\$	34	24			F7	F7		
5	%	35	25			F8	F8		
6	&	36	26			5E	5E	1E	1E
7	/	37	2F			5C	5C	1C	1C
8	(	38	28			5B	7B	1B	1B
9	)	39	29			5D	7D	1D	1D
0	=	30	3D			B9	B9		
+	?	2B	3F			F6	B0		
,	,	m -- ,	m -- ,			FE	FE		
Q	c	71	c 51	11	11				
W	c	77	c 57	17	17	7E	7E		
Y	c	79	c 59	19	19	m -- ^	AA		
A	c	D4	c D0				B3	B3	
"	^	m -- "	5E	1E	1E	7C	7C		
@	*	40	2A	00	00				
A	c	61	c 41	01	01	c D4	c D0		
Æ	c	D7	c D3			AF	AF		
Ø	c	D6	c D2			60	27		
Z	c	7A	c 5A	1A	1A				
M	c	6D	c 4D	0D	0D	FA	FA		
,	;	2C	3B			3C	3C		
.	:	2E	3A			3E	3E		
-	-	2D	5F	1F	1F	5F	5F	1F	1F

c Affected by Caps key.m Mute character (as shown). Character code generated *next* keystroke.

Table 13-22. Danish Alternate Mode Character Codes

Keycap Char	Normal	Shift	CTRL	Shift- CTRL	Ext	Shift- Ext	Ext- CTRL	Shift- Ext- CTRL
< >	3C	3E						
1 !	31	21			00 78	00 78	00 78	00 78
2 "	32	22			00 79	00 79	00 79	00 79
3 §	33	15			00 7A	00 7A	23	23
4 \$	34	24			00 7B	00 7B	00 7B	00 7B
5 %	35	25			00 7C	00 7C	00 7C	00 7C
6 &	36	26			00 7D	00 7D	00 7D	00 7D
7 /	37	2F	1C	1C	00 7E	00 7E	5C	5C
8 (	38	28	1B	1B	00 7F	00 7F	5B	7B
9 )	39	29	1D	1D	00 80	00 80	5D	7D
0 =	30	3D			00 81	00 81	00 81	00 81
+ ?	2B	3F			00 82	00 82	00 82	00 82
' `	m -- ,	m -- `			00 83	00 83	00 83	00 83
Q c	71	c 51	11	11	00 10	00 10	00 10	00 10
W c	77	c .57	17	17	00 11	00 11	7E	7E
Y c	79	c 59	19	19	00 15	00 15	00 15	00 15
A c	86	c 8F						
" ^	m -- "	5E	1E	1E			7C	7C
@ *	40	2A	00 03	00 03				
A c	61	c 41	01	01	00 1E	00 1E	00 1E	00 1E
Æ c	91	c 92						
Ø++ c	6F	c 4F					60	27
Z c	7A	c 5A	1A	1A	00 2C	00 2C	00 2C	00 2C
M c	6D	c 4D	0D	0D	00 32	00 32	00 32	00 32
, ;	2C	3B						
. :	2E	3A						
- _	2D	5F	1F	1F				

c Affected by the **Caps** key.

m Mute character (as shown). Character code generated *next* keystroke.

++ Coded as normal "O" because "Ø" doesn't exist in Alternate mode character set.

Figure 13–11. Norwegian Keyboard

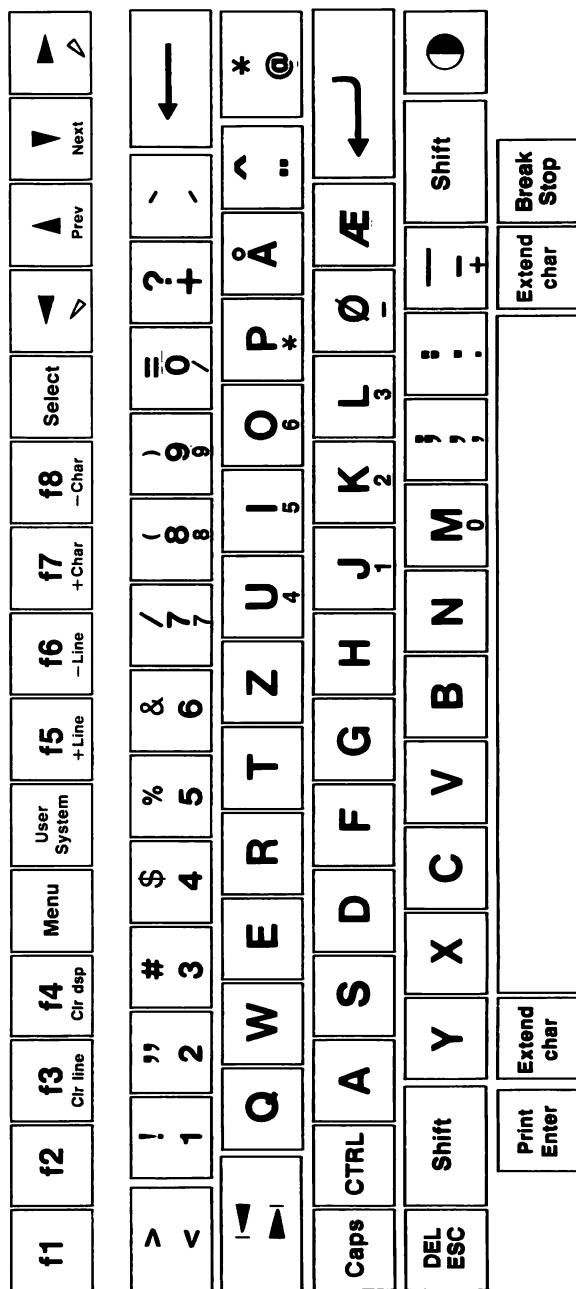


Table 13-23. Norwegian HP Mode Character Codes

Keycap Char	Normal	Shift	CTRL	Shift- CTRL	Ext	Shift- Ext	Ext- CTRL	Shift- Ext- CTRL
< >	3C	3E			FB	FD		
1 !	31	21			B8	B8		
2 "	32	22			40	40	00	00
3 #	33	23			23	23		
4 \$	34	24			F7	F7		
5 %	35	25			F8	F8		
6 &	36	26			5E	5E	1E	1E
7 /	37	2F			5C	5C	1C	1C
8 (	38	28			5B	7B	1B	1B
9 )	39	29			5D	7D	1D	1D
0 =	30	3D			B9	B9		
+	2B	3F			F6	B0		
,	m -- ,	m -- ,			FE	FE		
Q c	71	c 51	11	11				
W c	77	c 57	17	17	7E	7E		
Y c	79	c 59	19	19	m -- ^	AA		
A c	D4	c D0			B3	B3		
" ^	m -- "	5E	1E	1E	7C	7C		
@ *	40	2A	00	00				
A c	61	c 41	01	01	c D4	c D0		
Ø c	D6	c D2			AF	AF		
Æ c	D7	c D3			60	27		
Z c	7A	c 5A	1A	1A				
M c	6D	c 4D	0D	0D	FA	FA		
,	2C	3B			3C	3C		
.	2E	3A			3E	3E		
- _	2D	5F	1F	1F	5F	5F	1F	1F

c Affected by Caps key.

m Mute character (as shown). Character code generated next keystroke.

Table 13-24. Norwegian Alternate Mode Character Codes

Keycap Char	Normal	Shift	CTRL	Shift- CTRL	Ext	Shift- Ext	Ext- CTRL	Shift- Ext- CTRL
< >	3C	3E						
1 !	31	21		00	78	00	78	00
2 "	32	22		00	79	00	79	00
3 #	33	23		00	7A	00	7A	00
4 \$	34	24		00	7B	00	7B	00
5 %	35	25		00	7C	00	7C	00
6 &	36	26		00	7D	00	7D	00
7 /	37	2F	1C	1C	00	7E	00	5C
8 (	38	28	1B	1B	00	7F	00	5B
9 )	39	29	1D	1D	00	80	00	7D
0 =	30	3D			00	81	00	81
+ ?	2B	3F			00	82	00	82
,	m -- ,	m -- ,			00	83	00	83
Q c	71	c 51	11	11	00	10	00	10
W c	77	c 57	17	17	00	11	00	11
Y c	79	c 59	19	19	00	15	00	15
A c	86	c 8F						
" ^	m -- "	5E	1E	1E			7C	7C
@ *	40	2A	00 03	00 03				
A c	61	c 41	01	01	00	1E	00	1E
ø++	c 6F	c 4F						
Æ c	91	c 92					60	27
Z c	7A	c 5A	1A	1A	00	2C	00	2C
M c	6D	c 4D	0D	0D	00	32	00	32
,	2C	3B						
.	2E	3A						
- _	2D	5F	1F	1F				

c Affected by the **Caps** key.

m Mute character (as shown). Character code generated *next* keystroke.

++ Coded as normal "O" because "ø" doesn't exist in Alternate mode character set.

Figure 13–12. Swedish Keyboard

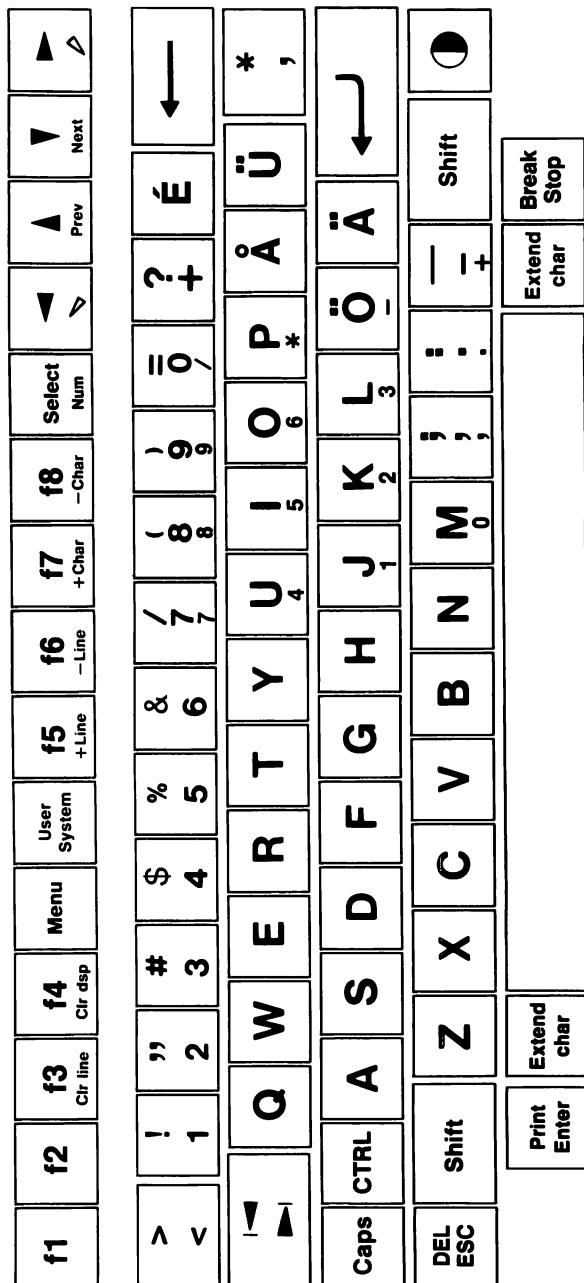


Table 13-25. Swedish HP Mode Character Codes

Keycap Char	Normal	Shift	CTRL	Shift- CTRL	Ext	Shift- Ext	Ext- CTRL	Shift- Ext- CTRL
< >	3C	3E			FB	FD		
1 !	31	21			B8	B8		
2 "	32	22			40	40	00	00
3 #	33	23			23	23		
4 \$	34	24			F7	F7		
5 %	35	25			F8	F8		
6 &	36	26			5E	5E	1E	1E
7 /	37	2F			5C	5C	1C	1C
8 (	38	28			5B	7B	1B	1B
9 )	39	29			5D	7D	1D	1D
0 =	30	3D			B9	B9		
+	2B	3F			F6	B0		
É	c C5	c DC			FE	FE		
Q	c 71	c 51	11	11				
W	c 77	c 57	17	17	7E	7E		
Y	c 79	c 59	19	19	m -- ^	AA		
Å	c D4	c D0			B3	B3		
Ü	c CF	c DB			7C	7C		
,	*	27	2A					
A	c 61	c 41	01	01	c D4	c D0		
Ö	c CE	c DA			AF	AF		
Ä	c CC	c D8			60	27		
Z	c 7A	c 5A	1A	1A				
M	c 6D	c 4D	0D	0D	FA	FA		
,	;	2C	3B		3C	3C		
.	:	2E	3A		3E	3E		
-	-	2D	5F	1F	5F	5F	1F	1F

c Affected by **Caps** key.  
 m Mute character (as shown). Character code generated *next* keystroke.

Table 13-26. Swedish Alternate Mode Character Codes

Keycap Char	Normal	Shift	CTRL	Shift- CTRL	Ext	Shift- Ext	Ext- CTRL	Shift- Ext- CTRL
< >	3C	3E						
1 !	31	21			00 78	00 78	00 78	00 78
2 "	32	22	00 03	00 03	00 79	00 79	40	40
3 #	33	23			00 7A	00 7A	00 7A	00 7A
4 \$	34	24			00 7B	00 7B	00 7B	00 7B
5 %	35	25			00 7C	00 7C	00 7C	00 7C
6 &	36	26	1E	1E	00 7D	00 7D	5E	5E
7 /	37	2F	1C	1C	00 7E	00 7E	5C	5C
8 (	38	28	1B	1B	00 7F	00 7F	5B	7B
9 )	39	29	1D	1D	00 80	00 80	5D	7D
0 =	30	3D			00 81	00 81	00 81	00 81
+	?	2B	3F		00 82	00 82	00 82	00 82
É	c 82	c 90			00 83	00 83	00 83	00 83
Q	c 71	c 51	11	11	00 10	00 10	00 10	00 10
W	c 77	c 57	17	17	00 11	00 11	7E	7E
Y	c 79	c 59	19	19	00 15	00 15	00 15	00 15
Å	c 86	c 8F						
Ü	c 81	c 9A					7C	7C
,	*	27	2A					
A	c 61	c 41	01	01	00 1E	00 1E	00 1E	00 1E
Ö	c 94	c 99						
Ä	c 84	c 8E					60	60
Z	c 7A	c 5A	1A	1A	00 2C	00 2C	00 2C	00 2C
M	c 6D	c 4D	0D	0D	00 32	00 32	00 32	00 32
,	;	2C	3B					
.	:	2E	3A					
-	-	2D	5F	1F	1F			

c Affected by the **Caps** key.

When a mute key (as defined for the local keyboard) is pressed, that mute character becomes "pending". A character code isn't generated until the following key is pressed. The generated character code depends upon the pending mute character and the key that is pressed next. (Refer to the following tables.) The following rules apply to mute characters:

- The mute mapping is in effect for only the key that immediately follows the mute character.
- Only one mute can be pending at a time. If another mute character is pressed, *that* character becomes the pending mute character.
- If the key following a mute character is held down to invoke a repeated key function, then the outgoing muted character code is repeated.

13

Table 13-27. HP Mode Muted Character Codes

Next Keycap Character	,	'	^	"	-
space (20h)	27	60	5E	20	7E
a (61h)	C4	C8	C0	CC	E2
e (65h)	C5	C9	C1	CD	65
i (69h)	D5	D9	D1	DD	69
n (6Eh)	6E	6E	6E	6E	B7
o (6Fh)	C6	CA	C2	CE	EA
u (75h)	C7	CB	C3	CF	75
y (79h)	79	79	79	EF	79
A (41h)	E0	A1	A2	D8	E1
E (45h)	DC	A3	A4	A5	45
I (49h)	E5	E6	A6	A7	49
N (4Eh)	4E	4E	4E	4E	B6
O (4Fh)	E7	E8	DF	DA	E9
U (55h)	ED	AD	AE	DB	55
Y (59h)	59	59	59	EE	59
other	*	*	*	*	*

\* Character code is same as that for "other" character.

**Table 13-28. Alternate Mode Muted Character Codes**

Next Keycap Character	,	'	^	"	-
space (20h)	27	60	5E	20	7E
a (61h)	A0	85	83	84	61
e (65h)	82	8A	88	89	65
i (69h)	A1	8D	8C	8B	69
n (6Eh)	6E	6E	6E	6E	A4
o (6Fh)	A2	95	93	94	6F
u (75h)	A3	97	96	81	75
A (41h)	41	41	41	8E	41
E (45h)	90	45	45	45	45
N (4Eh)	4E	4E	4E	4E	A5
O (4Fh)	4F	4F	4F	99	4F
U (55h)	55	55	55	9A	55
other	*	*	*	*	*

\* Character code is same as that for "other" character.



# Comparisons With Other Computers

## A.1 Comparison With the HP 110

The HP 110 (The Portable) is largely compatible with the Portable PLUS. However, some of the advanced capabilities of the Portable PLUS cause some differences, which are described below.

The HP 110 provided a general-purpose, escape-sequence driven software interface. The Portable PLUS has a more flexible and expanded operating system that provides this same interface plus a complete BIOS interface. To a great extent, software is "upward compatible" between these products--HP 110 software is largely compatible with the Portable PLUS.

Differences in hardware components introduced some incompatibilities. The 25-line display is an obvious change from the 16-line display of the HP 110. However, most "well-behaved" applications written for the HP 110 will run without modification on the Portable PLUS using a 16-line format. Applications that are less "well-behaved" may require changes.

The following list describes the main areas of incompatibility:

- **Reading Serial Numbers and Device ID.** The Portable PLUS reads the serial number using the Read Device ID escape sequence, unlike the HP 110. The HP 110 product number (110A) is returned for all three of the values specified in the escape sequence. (Refer to the "HP Graphics Escape Sequences" table in chapter 6.) The device ID returned for the Portable PLUS is 45711.
- **Accessing the Display.** The Portable PLUS provides an expanded software interface to the display in both alpha and graphics modes. (Refer to "Video I/O Interrupt" in chapter 5 and "CONsole Driver" in chapter 6.) In addition, it uses a different LCD controller than the HP 110. (Refer to "Display Operation" in chapter 7.)

- Accessing the Modem. The modem is an optional addition to the Portable PLUS, whereas it was standard in the HP 110. This means that applications will have to check for the presence of a modem. (Refer to the "AUX I/O Control Commands" table in chapter 6.)
- Detecting a Modem Carrier. Unlike the HP 110, the Portable PLUS generates a software interrupt when a modem carrier signal is detected. (Refer to "Int 42h" in chapter 5.)
- Reading from the Aux Device. The Portable PLUS AUX device does not block characters. For example, when a driver call is made to the AUX device to read 10 characters, it will return immediately with the 10 or less characters that it has already buffered. In the same situation the HP 110 will wait in the driver until it has the full 10 characters to return.
- Using Device Names. The HP 110 shared the COM1 device name between the serial interface and the built-in modem. The Portable PLUS uses the COM1 name for only the serial port, and uses the new COM3 name for the optional modem. (Refer to "Introduction" in chapter 6.)

---

## A.2 Comparison With the HP 150

There are some fundamental differences between the HP 150 and the Portable PLUS that may make it difficult to move programs between the two.

- Display Size. The Portable PLUS has a 25 x 80 character display, while the HP 150 has a 27 x 80 character display. Similarly, the Portable PLUS has a 480 x 200 pixel graphics display, while the HP 150 has a 512 x 390 pixel graphics display.
- Display Buffers. The HP 150 has two separate display buffers, one for alpha and one for graphics. The Portable PLUS has one display buffer which can be either alpha or graphics.
- Touch Screen. The Portable PLUS does not have the touch screen provided on the HP 150.
- Accessing the Display. Methods of communicating to the display also differ. The Portable PLUS is a simple, character mode terminal which accepts the escape sequences listed in "CONsole Driver" in chapter 6, including simple graphics escape sequences. There are alternate ways to communicate with the console such as using as

the Video I/O Interrupt, the System Service Interrupt and other BIOS interrupts. The HP 150 implements the HP 2623 terminal escape sequences which support block mode and full graphics capabilities. It has its own alternate method of communicating with the console, AGIOS, which is not implemented on the Portable PLUS. The various BIOS interrupts on the Portable PLUS do not exist on the HP 150.

- Keyboard. The keyboard layout is obviously different on the two computers. The HP 150 has some keys that the Portable PLUS does not, but most of the functions can be generated by using the modifier keys. For example, the Insert Char key is a separate key on the HP 150 but it is accessed on the Portable PLUS by pressing the Extend char +Char key combination.
- Reading from the Aux Device. The Portable PLUS AUX device does not block characters. For example, when a driver call is made to the AUX device to read 10 characters, it will return immediately with the 10 or fewer characters that it has already buffered. In the same situation the HP 150 will wait in the driver until it has the full 10 characters to return.

---

## A.3 Comparison With the IBM PC

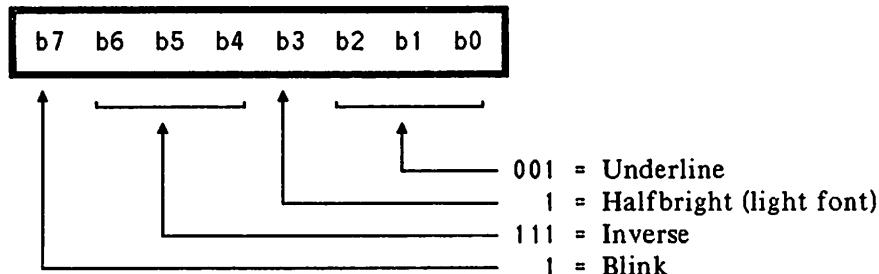
The Portable PLUS is designed to have a subset of the IBM PC BIOS Interrupts. Due to differences in hardware, it is not completely BIOS interrupt compatible with the IBM PC. *Programs that run on the IBM PC by talking directly with the hardware will not run on the Portable PLUS without modification.* Many of the programs that run on the IBM PC by talking to the BIOS interrupts will run on the Portable PLUS with little or no modifications. The differences in the Portable PLUS BIOS interrupts and the IBM PC BIOS interrupts are listed below.

### A.3.1 Video Interrupt (Int 10h)

- The Portable PLUS LCD controller allows for two display modes, 80 x 25 Alpha and 480 x 200 Graphics. The IBM PC can display three Alpha modes (40 x 25 BW, 40 x 25 color and 80 x 25 BW) and three Graphics modes (320 x 200 color, 320 x 200 BW and 640 x 200 BW) depending on the hardware configuration. All references to graphics bits beyond 480 are ignored on the Portable PLUS.
- The Portable PLUS provides two cursors, underline and box. The IBM PC allows for a configurable cursor from a single underline, a double underline, a triple underline--all the way to a full box.

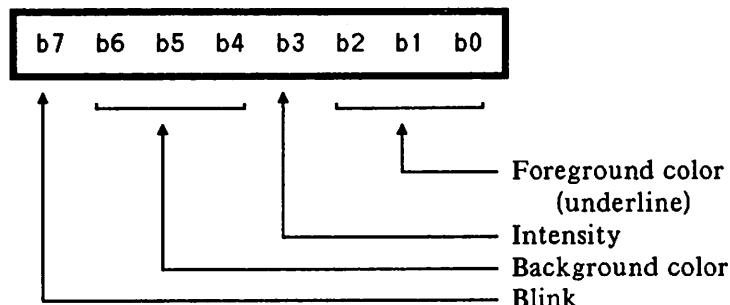
- The Portable PLUS has two pages of alpha display memory. The IBM PC has eight pages of alpha display memory.
- The Portable PLUS does not have a light pen; it will always indicate not triggered.
- The attribute bytes are mapped differently on the Portable PLUS and the IBM PC as shown below.

#### Portable PLUS Attribute Byte



A

#### IBM PC Attribute Byte



- The function to write characters in graphics mode works differently on the Portable PLUS than on the IBM PC. On the Portable PLUS, when a character is written to the display in graphics mode, it is ORed with the current character. When a character is written in graphics mode on the IBM PC, the current character is erased and the new character is written.

### A.3.2 Equipment Check Interrupt (Int 11h)

The result of the equipment check interrupt is dependent on the settings in the PAM System Configuration Menu rather than an actual poll of the hardware. There are a few items to keep in mind when interpreting the status.

- If the selected Printer Interface is HP-IL or HP-IB, then the number of printers attached is reported to be one if there are one or more printers on the loop. If no printer is found, then the number of printers is zero. If the selected Printer Interface is Serial then the number of printers reported is always one.
- The number of disc drives reported is two greater than the number of External Disc Drives selected in the PAM System Configuration Menu in order to include drives A: and B: .
- The initial video mode reported is 80 x 25 using Color Card, although there are no color capabilities.

### A.3.3 Diskette/Disc Interrupt (Int 13h)

The diskette interrupt is not implemented on the Portable PLUS. If called, it will simply return.

### A.3.4 Communications Interrupt (Int 14h)

- The Communications Interrupt can be directed on the Portable PLUS towards the serial port (DX = 0), the modem (DX = 1) or the current AUX device as specified in the PAM System Configuration Menu (DX > 1). The Communications Interrupt on the IBM PC can be used to select one of two RS-232 cards (DX = 0 or DX = 1).
- The status bits returned by the Portable PLUS are very similar to the status bits returned by the IBM PC, but there is a major difference in what the status means. The Portable PLUS buffers the input of characters. The status returned is the current status for the port, not necessarily the status of the port when the character was read. The IBM PC does not buffer its input; therefore, the status returned with each character is both the current status of the port and the status of the port when the character was read.

### A.3.5 Cassette Interrupt (Int 15h)

The cassette interrupt is not implemented on the Portable PLUS. If called, it will simply return.

### A.3.6 Keyboard Interrupt (Int 16h)

- The keyboard scan codes on the Portable PLUS do not match the keyboard scan codes on the IBM PC. The Portable PLUS always returns a scan code of 0 for the Keyboard Interrupt.
- The keyboard status bits differ slightly on the Portable PLUS and the IBM PC. The IBM PC returns the status of the left and right shift keys separately. The Portable PLUS can not distinguish between the left and right shift keys, so the status of the two always match. The IBM PC returns one additional status bit, the Scroll Lock State, which is not implemented on the Portable PLUS.

A

### A.3.7 Printer Interrupt (Int 17h)

- The IBM PC provides an area in RAM (40h:08h) called *Printer\_Base*, which contains the base address of the printer cards available. The Printer Interrupt can then be directed to one of three printer cards on the IBM PC. The same *Printer\_Base* area has been reserved by the Portable PLUS although it is not actually used. The Printer Interrupt is always directed to the Printer Interface specified in the PAM System Configuration Menu.
- The IBM PC also provides a data area that may be modified to alter the printer time-out. The Portable PLUS reserves this data area but does not use the information. Printer timeouts may not be altered in the Portable PLUS.
- The printer status byte returned by the Portable PLUS will always indicate "0" for Out-of-Paper (bit 5) and I/O Error (bit 3).

### A.3.8 Re-Boot Interrupt (Int 19h)

The Portable PLUS forces a hardware reset when the Re-boot interrupt is called. The IBM PC loads in track 0, sector 1 to the boot location and jumps to it.

### **A.3.9 Time-of-Day Interrupt (Int 1Ah)**

- The Portable PLUS maintains a separate time-of-day count and system clock. Writing to the Time of Day Interrupt on the Portable PLUS does not reset the system clock as it does on the IBM PC.
- Counts occur on the Portable PLUS at 18 times per second. Counts occur on the IBM PC at 18.2 times per second.

### **A.3.10 Keyboard Break Interrupt (Int 1Bh)**

The default action for the Portable PLUS is to flush the key queue and then put a ^C (03h) in it. The default action for the IBM PC is no action.



**B**

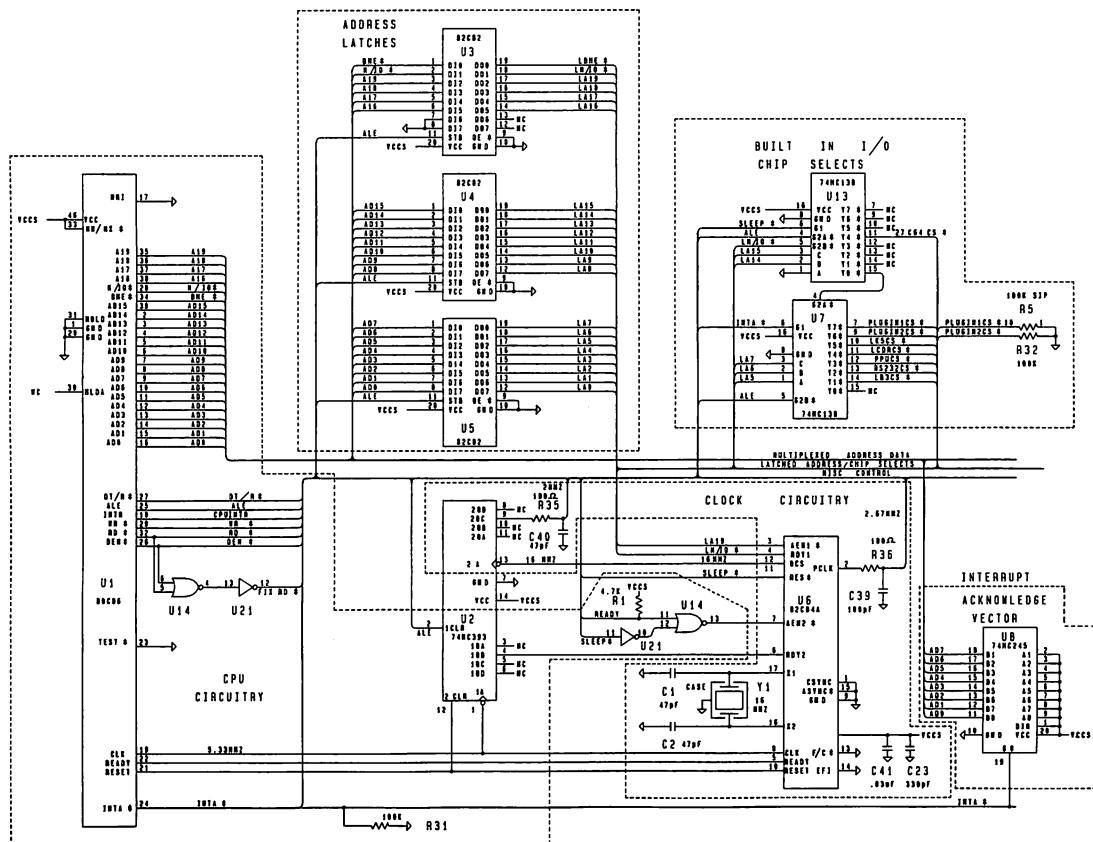
## Schematic Diagrams

---

This chapter presents schematic diagrams for the Portable PLUS and certain plug-in extensions. The following diagrams are included:

- Motherboard printed-circuit assembly (four figures).
- Memory board printed-circuit assembly.
- Modem printed-circuit assembly.
- Software (ROM) drawer printed-circuit assembly.
- Memory (RAM) drawer printed-circuit assembly (two figures).
- Memory drawer piggy-back printed-circuit assembly.

**Figure B-1. Motherboard PCA - Sheet 1**



MOTHER BOARD TO MEMORY BOARD  
CONNECTORS

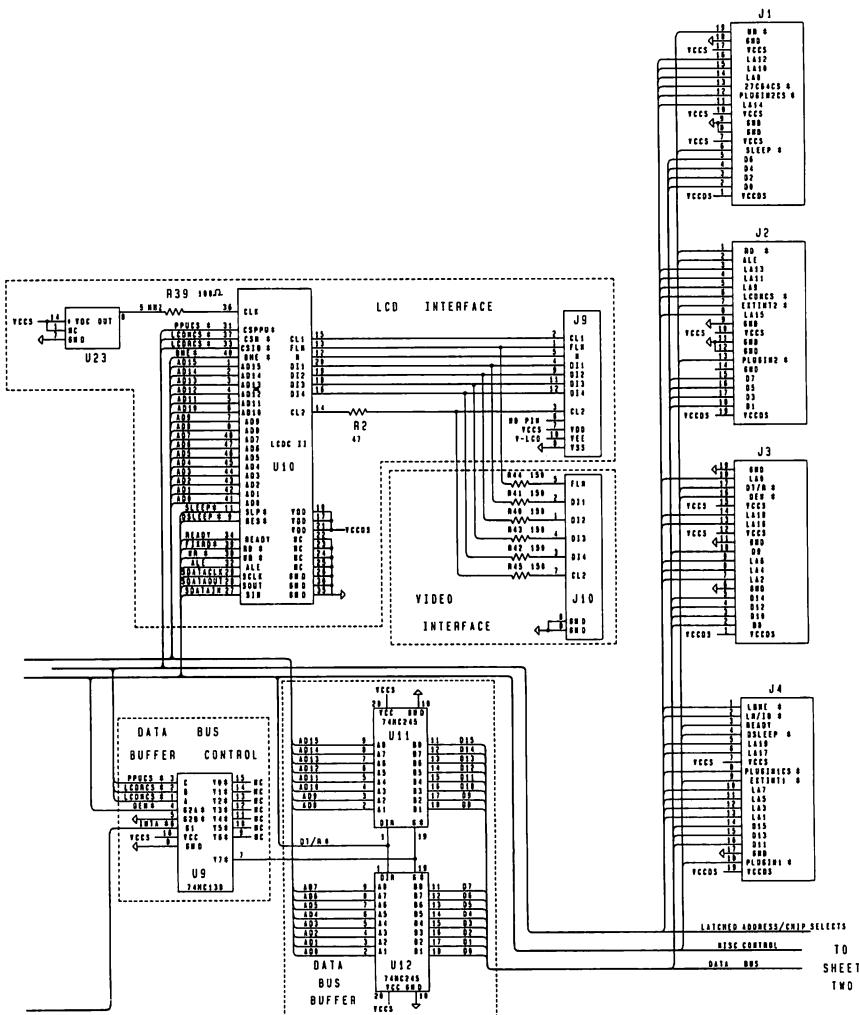
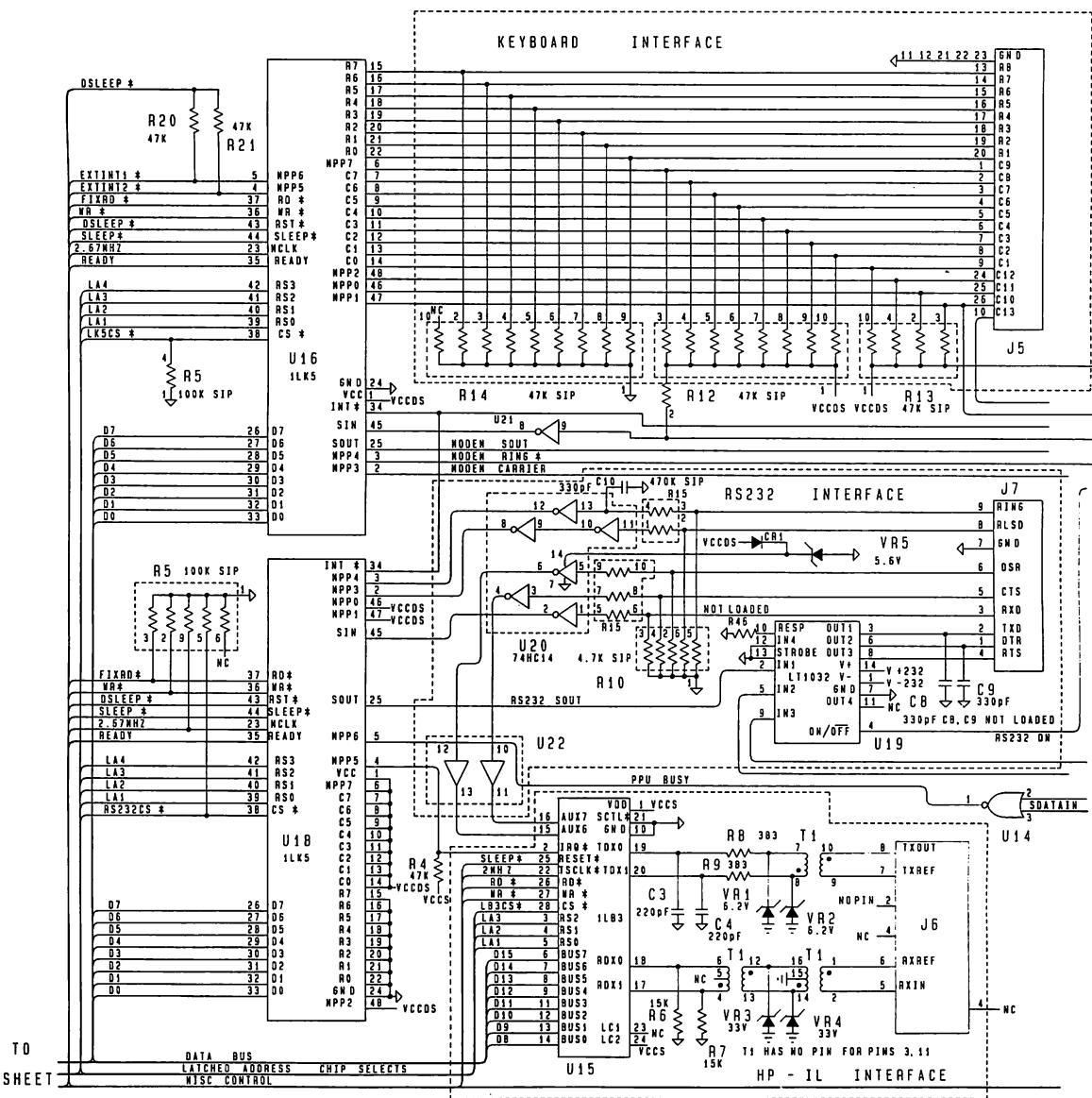
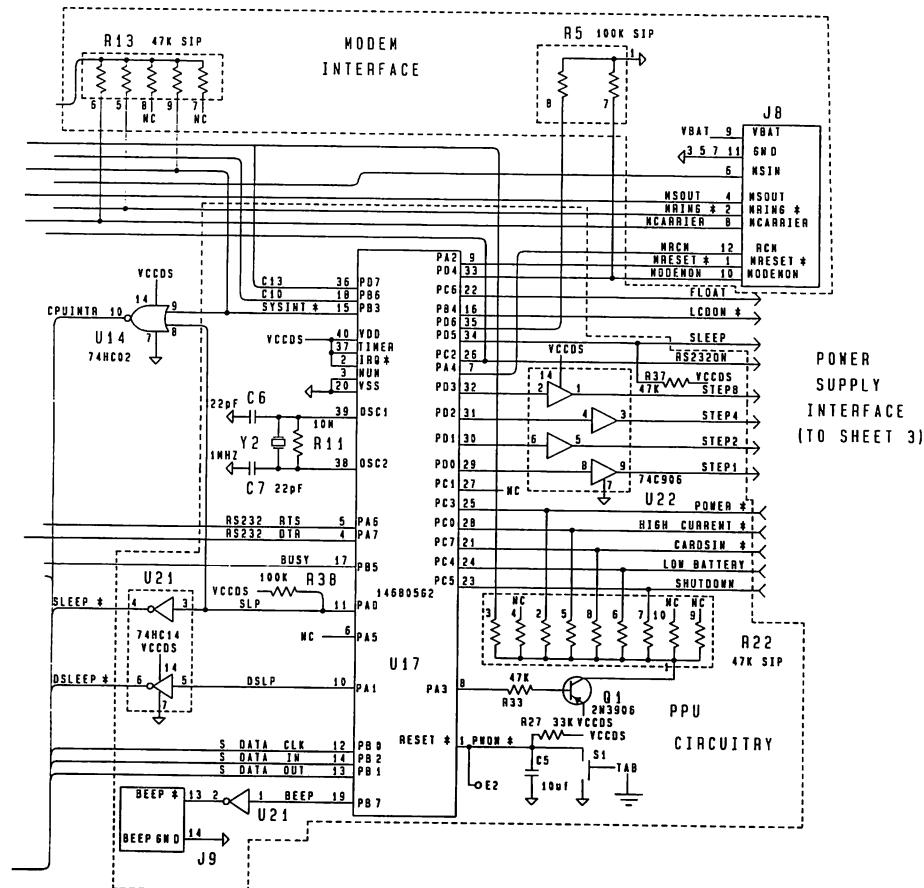
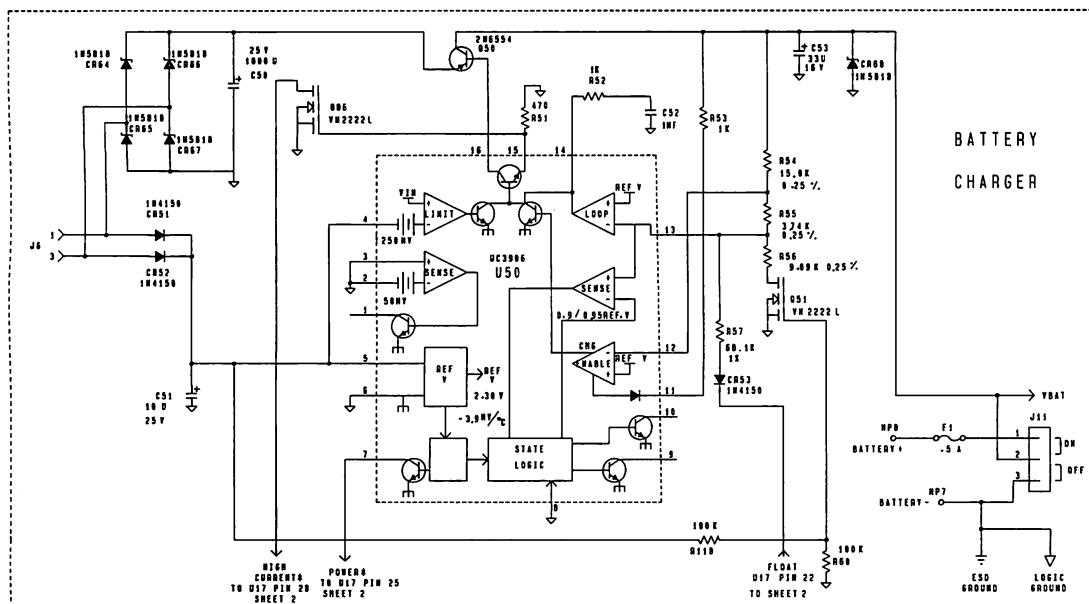


Figure B-2. Motherboard PCA - Sheet 2

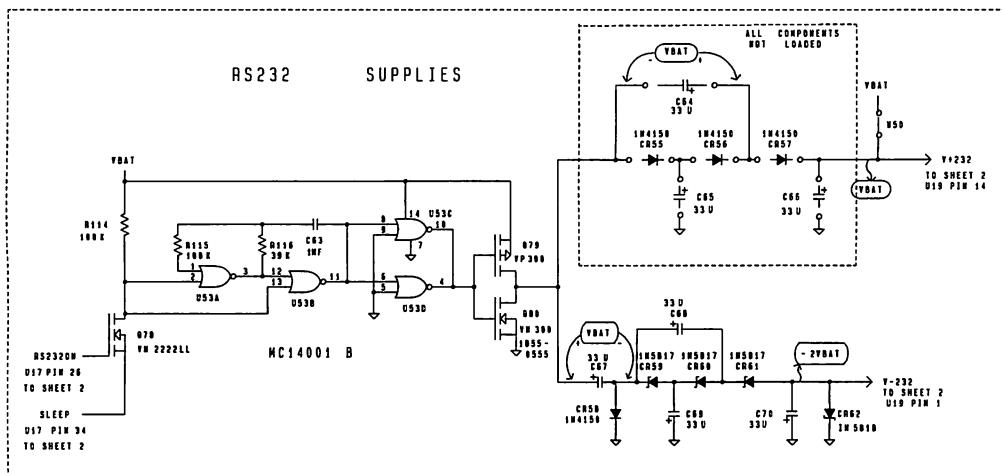




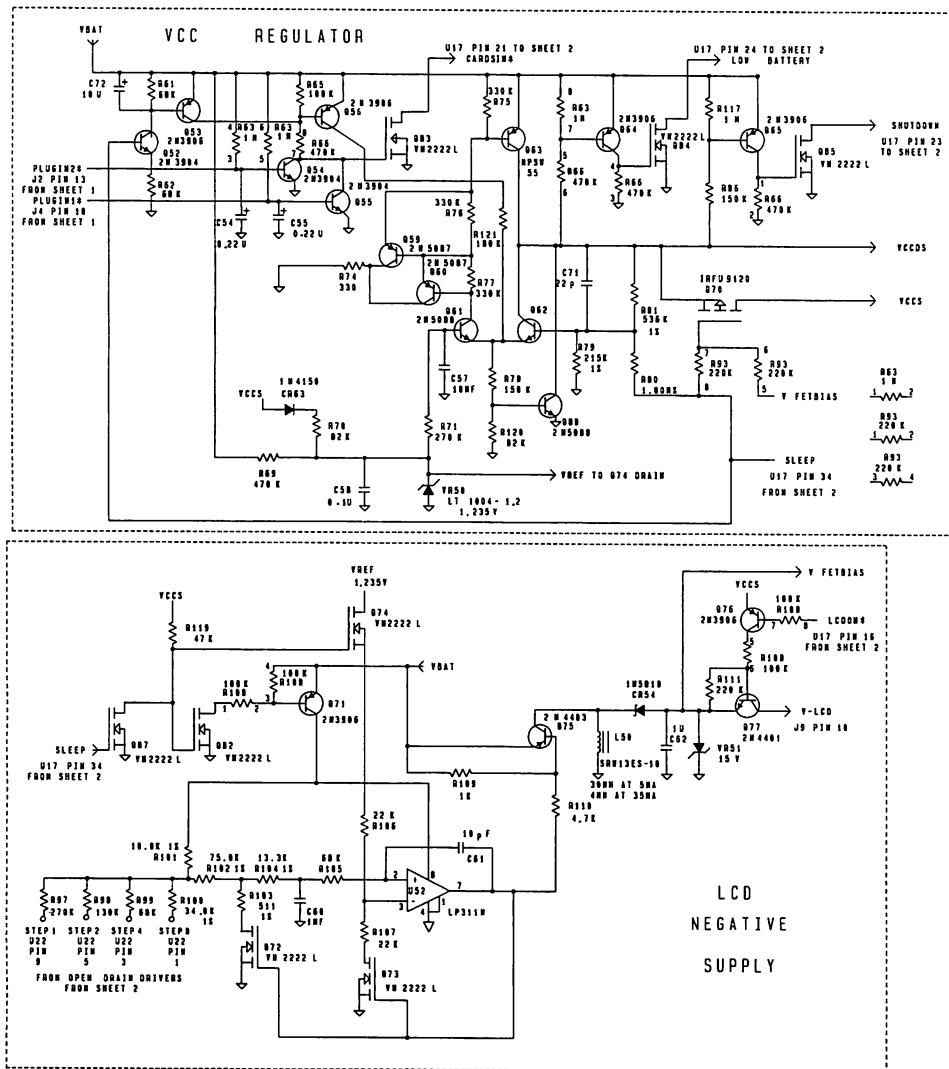
**Figure B-3. Motherboard PCA – Sheet 3**



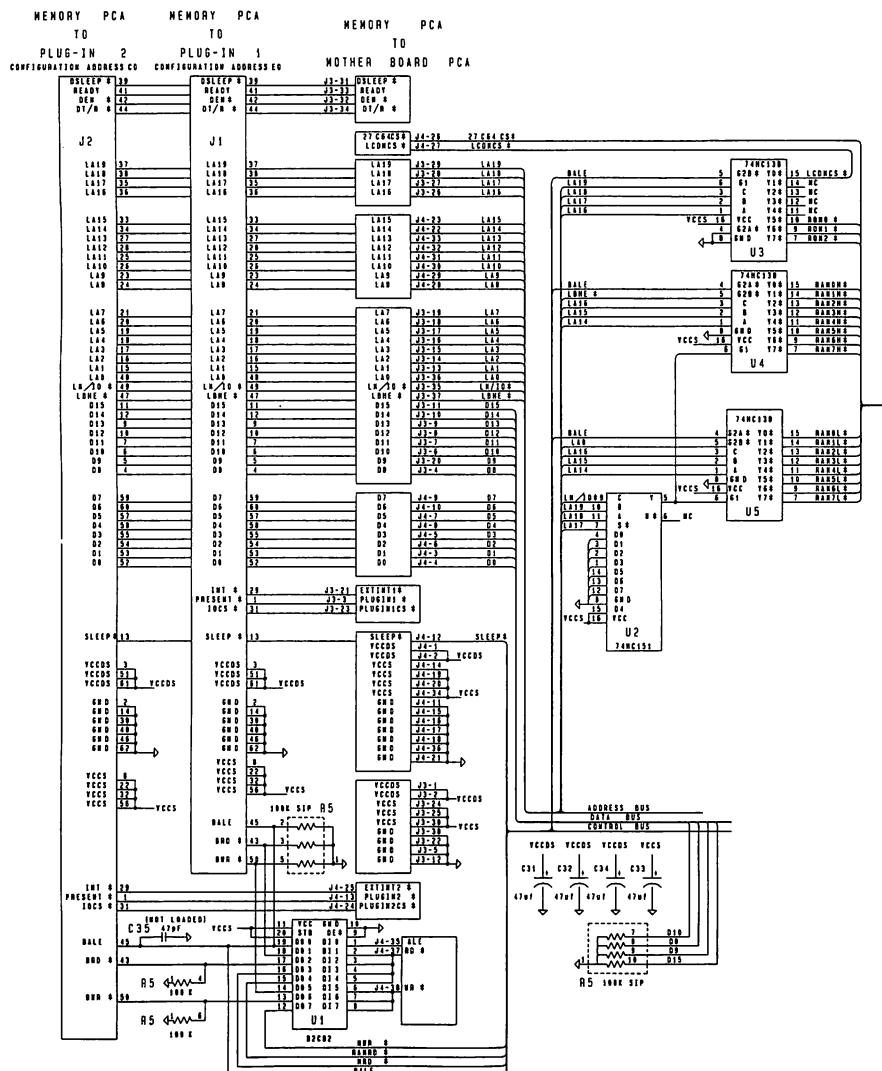
B

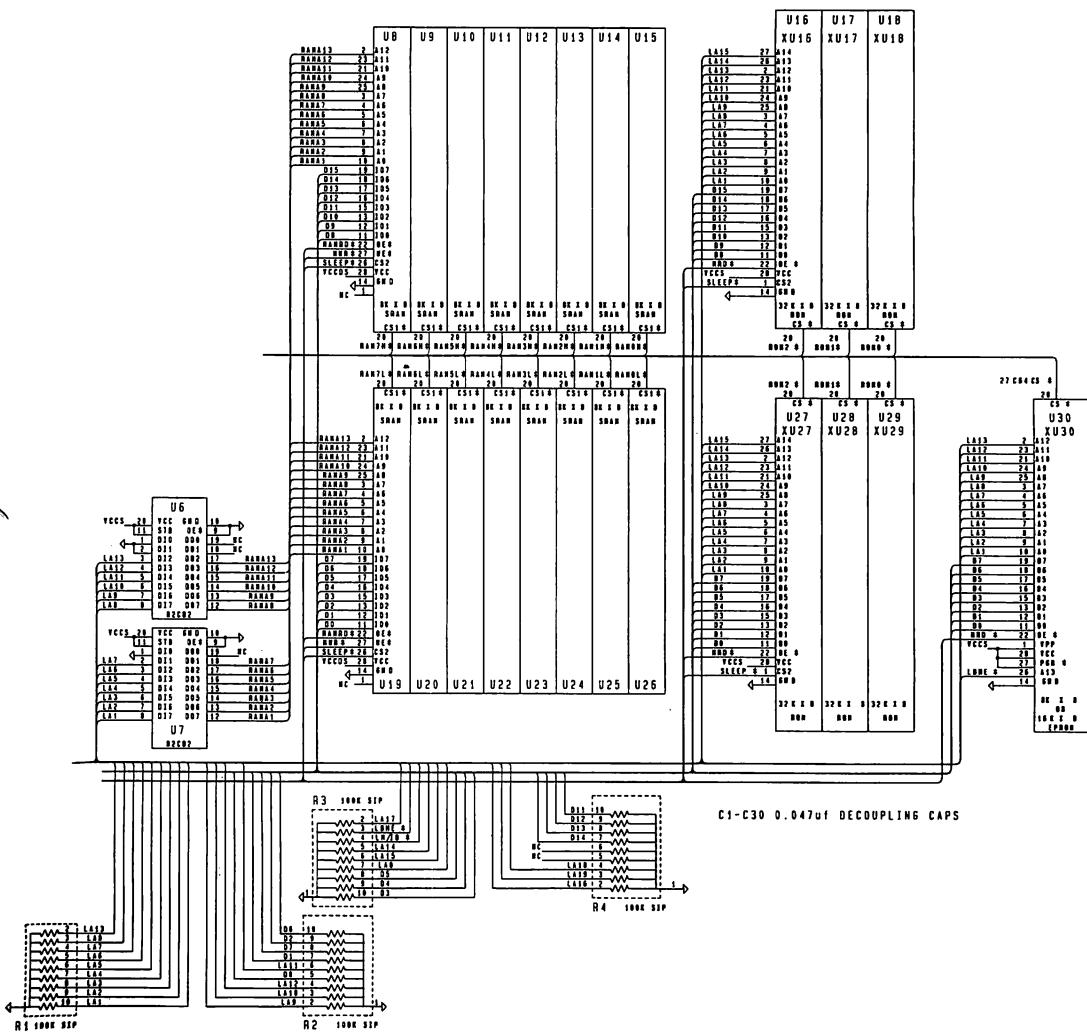


**Figure B-4. Motherboard PCA - Sheet 4**

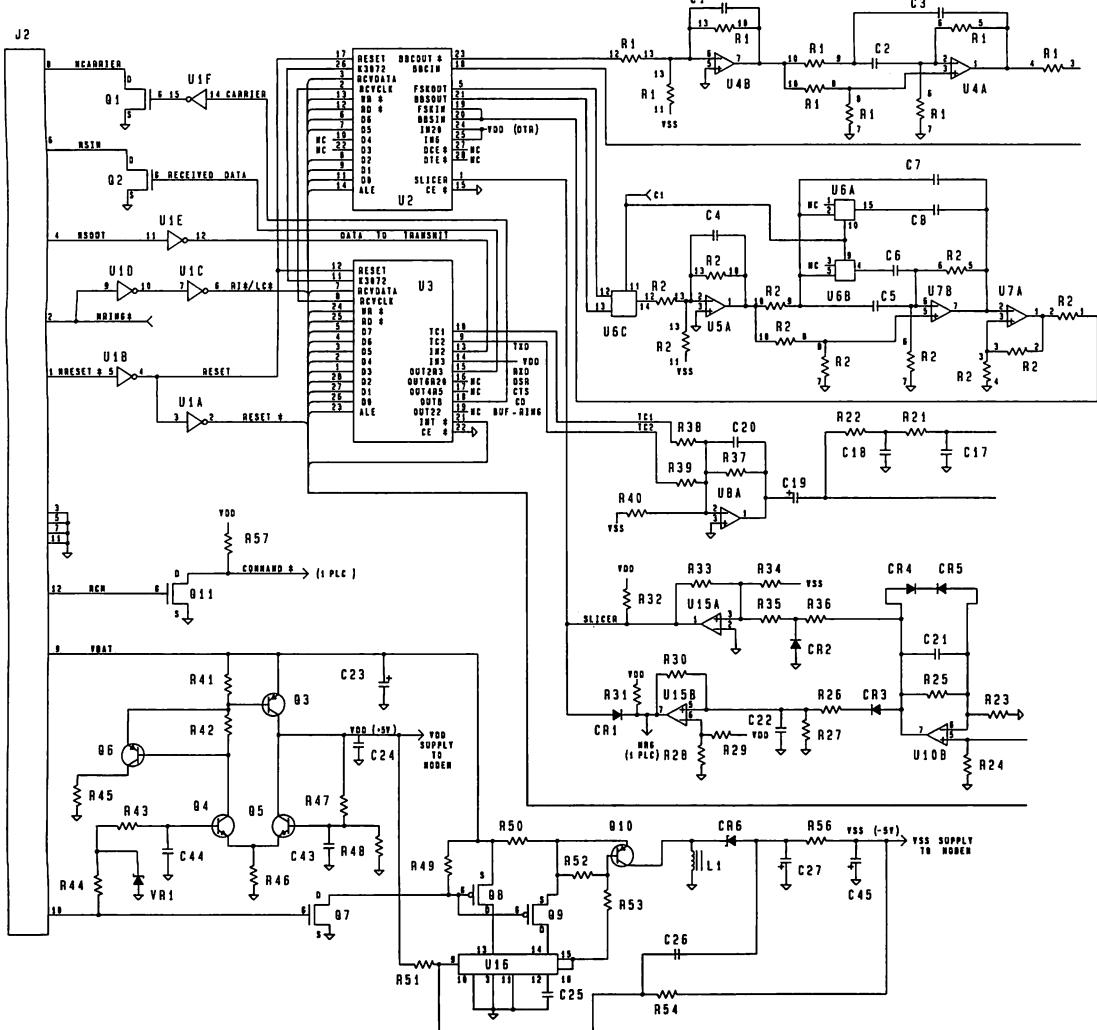


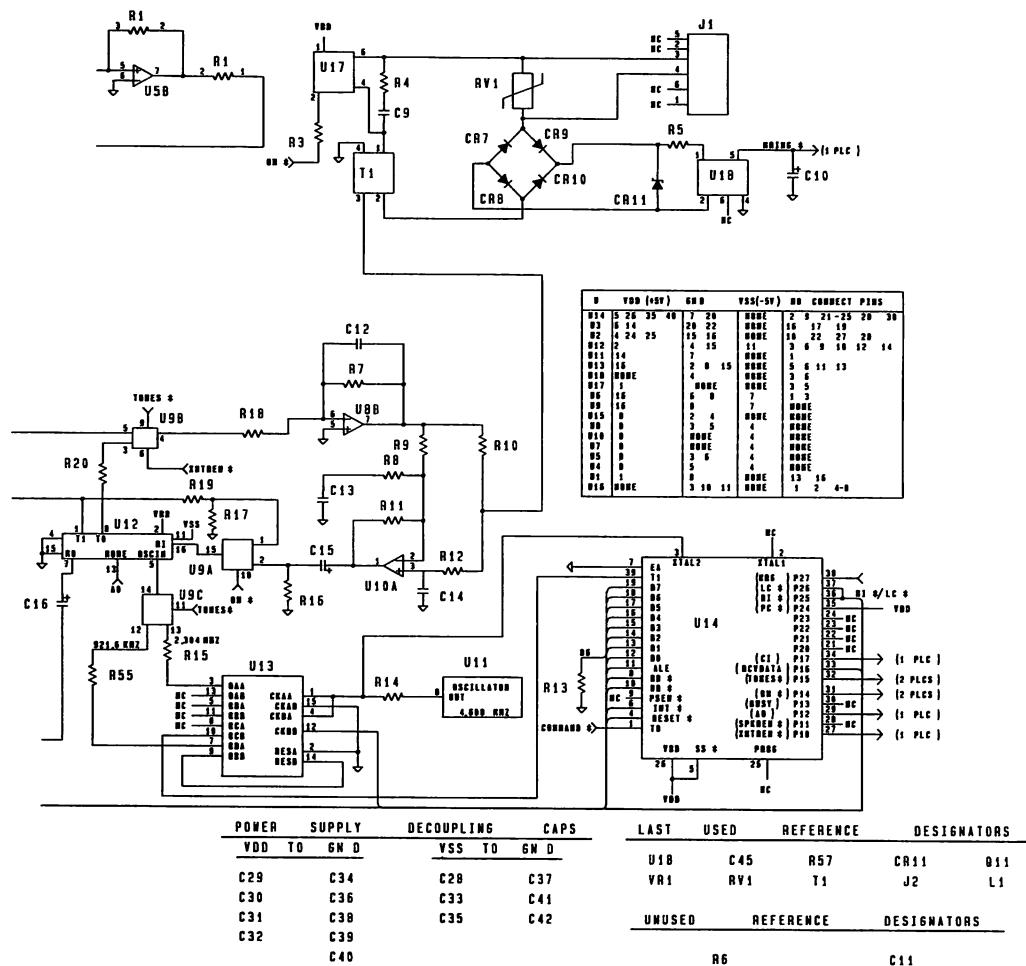
**Figure B-5. Memory Board PCA**



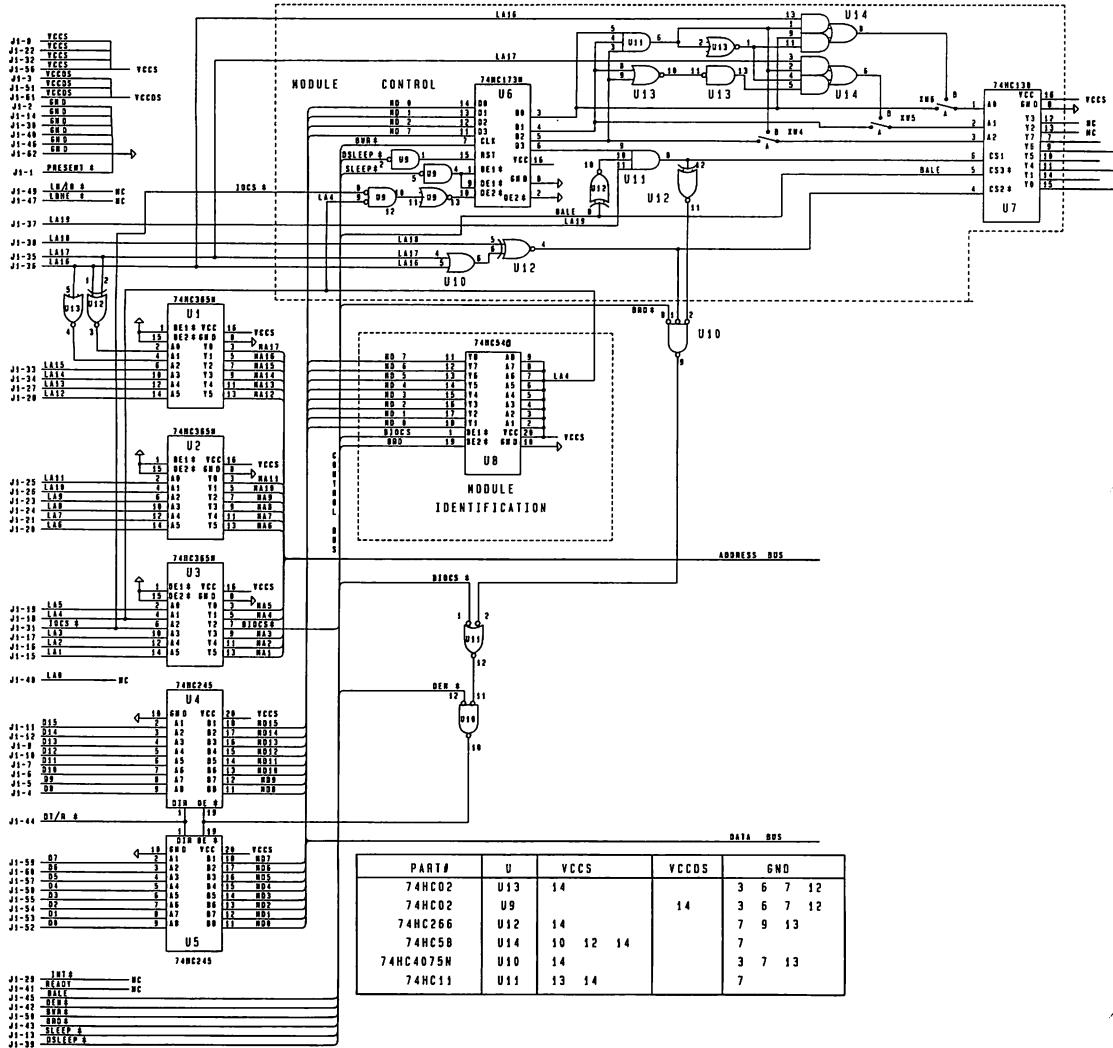


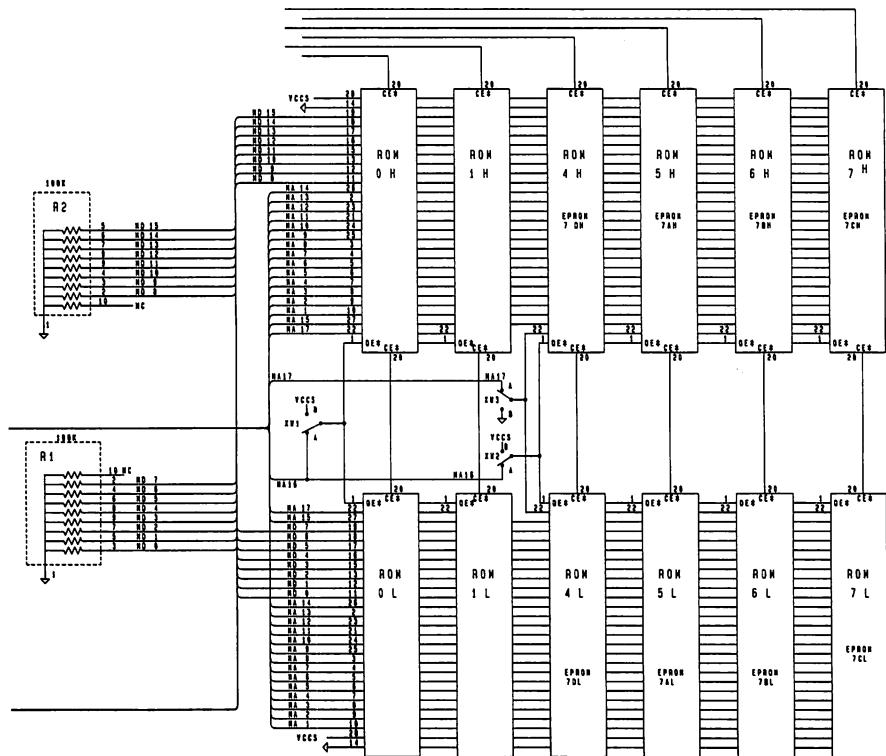
**Figure B-6. Modem PCA**



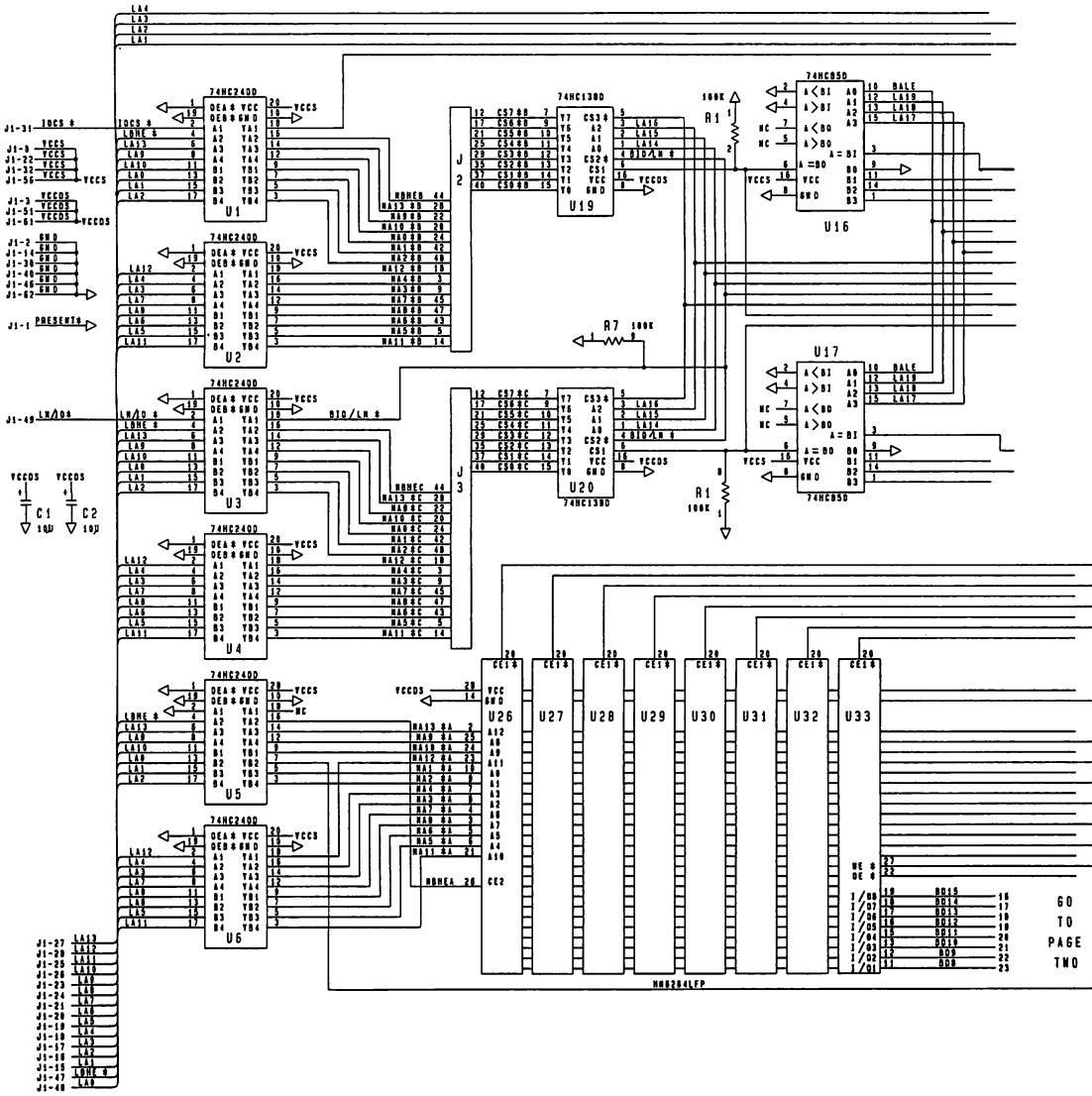


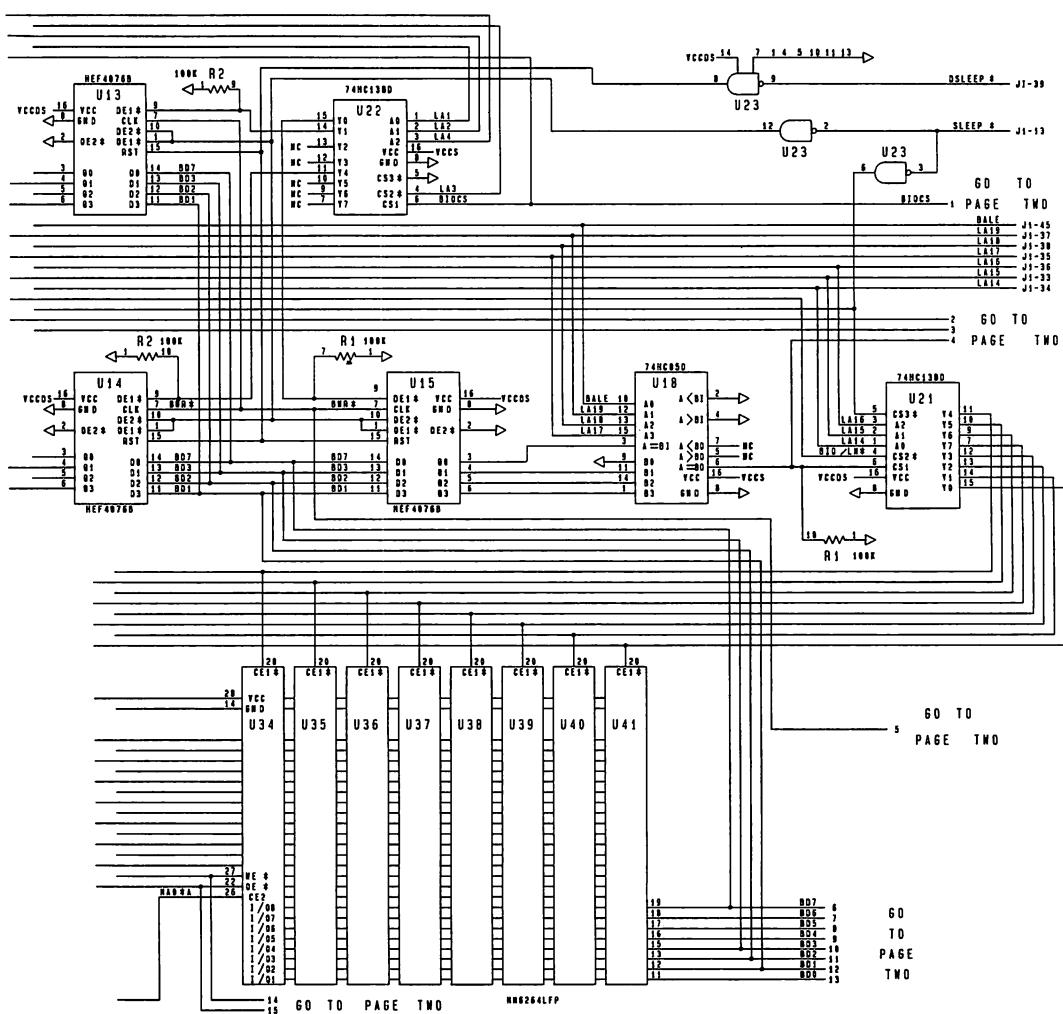
**Figure B-7. Software Drawer PCA**



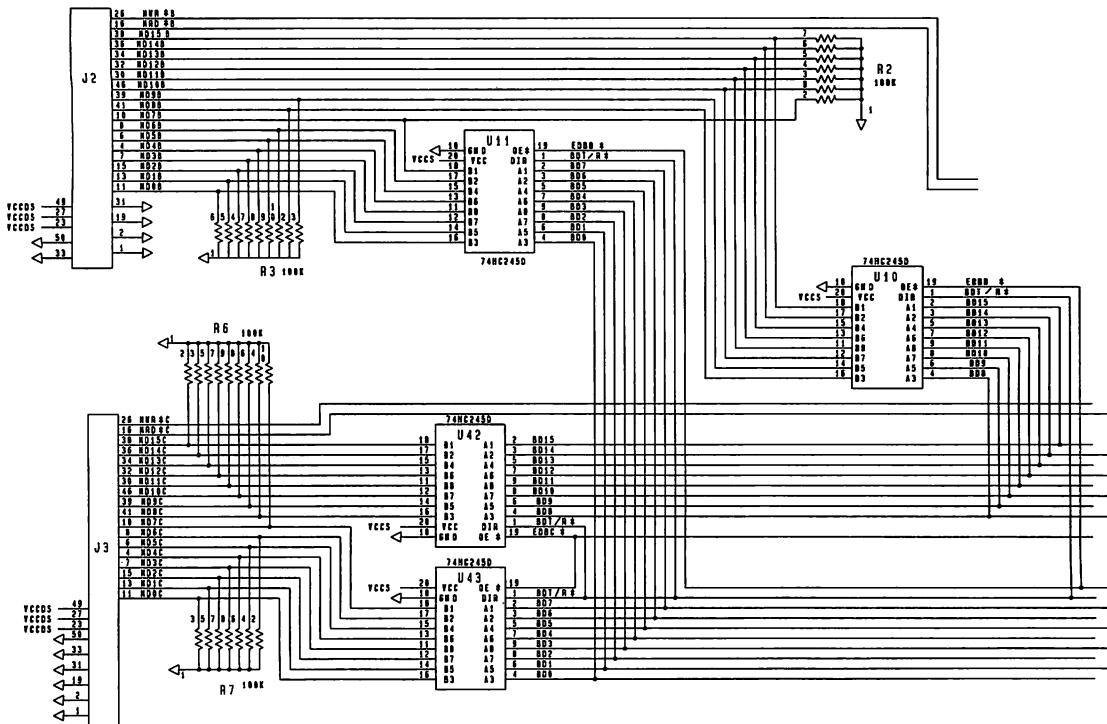


**Figure B-8. Memory Drawer PCA - Sheet 1**



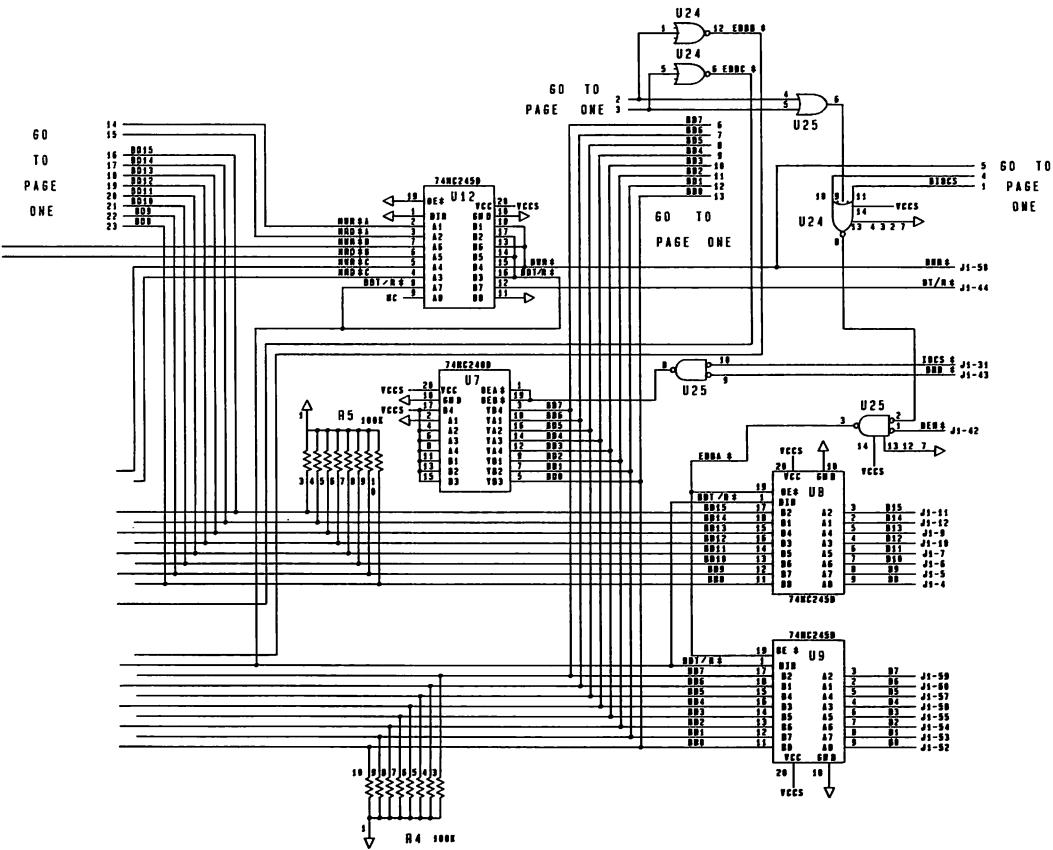


**Figure B-9. Memory Drawer PCA - Sheet 2**

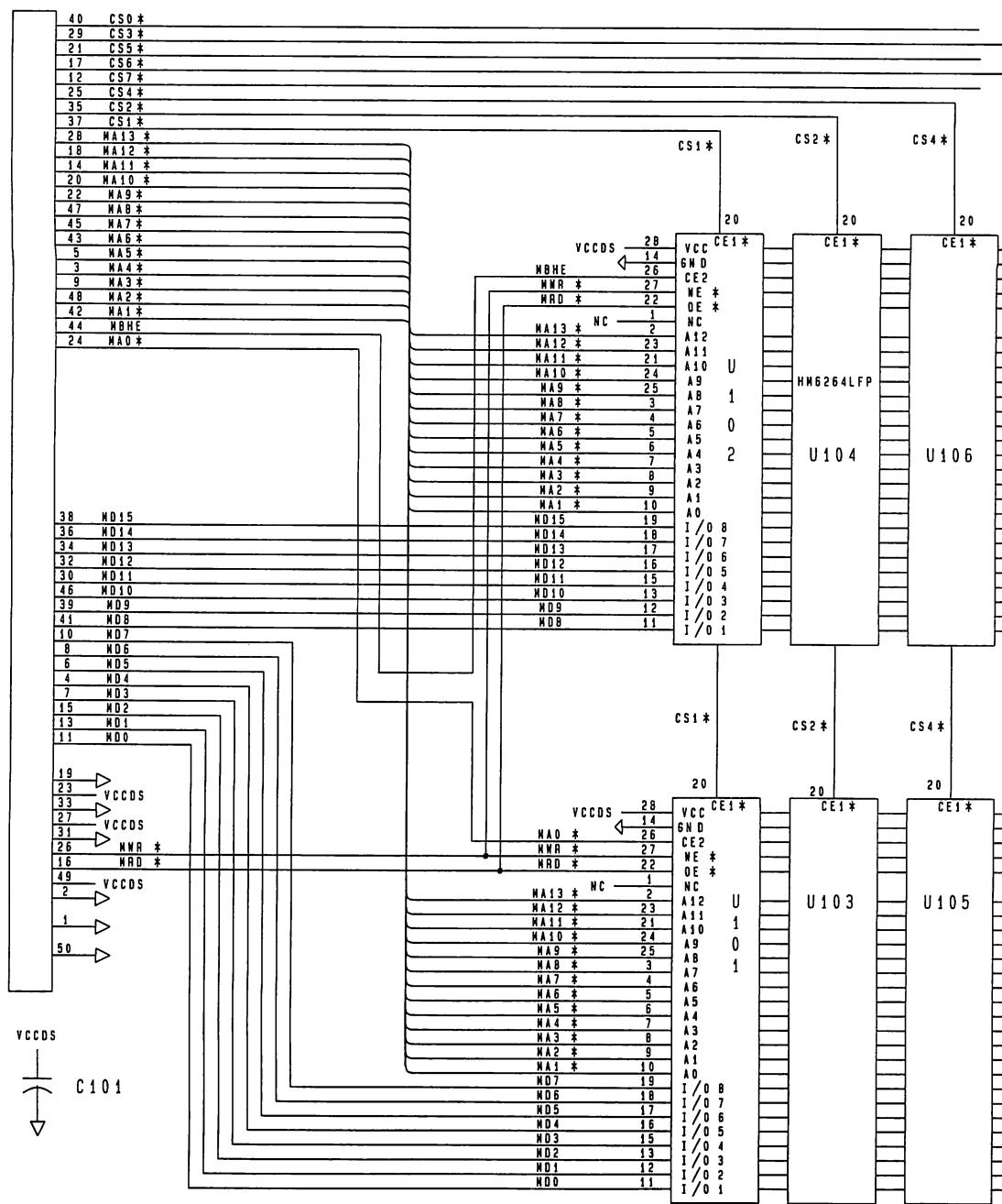


**NOTES**

U23	U24	74 HC 27	TRIPLE 3 - INPUT NOR
U25	74 HC 32	QUAD 2- INPUT OR	
C3 - C14	0.10 UF	DECOUPLING CAPACITOR	VCCS - GND
C15 - C26	0.10 UF	DECOUPLING CAPACITOR	VCCS - GND
R1 - R7	100 K	RESISTOR NETWORK	

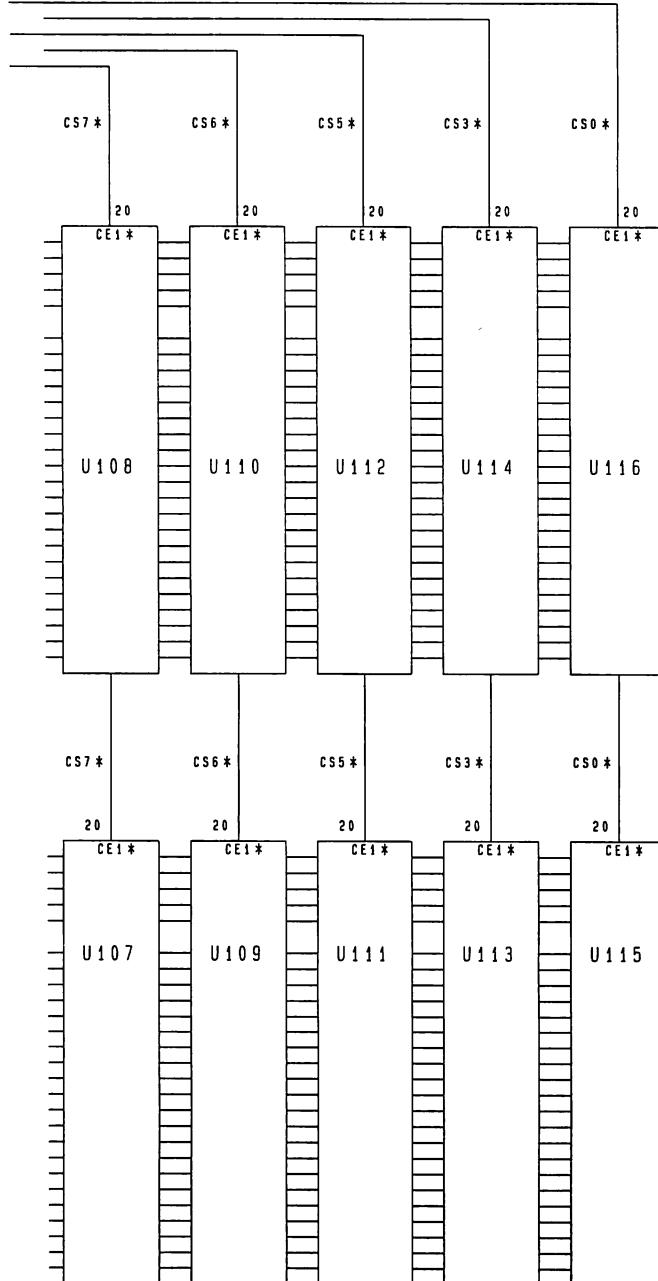


**Figure B-10. Memory Drawer Piggy-Back PCA**



C102 - C109    0 1 UF    DECOUPLING

CAPACITORS





# Assembler Listing for Configuration EPROM

This chapter presents the assembler listing for the U.S. English version of the configuration EPROM code. The listing contains the following parts, which begin on the pages listed.

Page	Information
C-2	Product number and Edisc boot sector.
C-3	Serial number.
C-3	Initial liquid-crystal display contrast.
C-3	Configuration EPROM boot code information.
C-3	Country-specific information.
C-4	Timer reference and key repeat rates.
C-4	Console status indicators.
C-5	PAM variable defaults.
C-6	Mute tables.
C-7	Configuration EPROM checksum.
C-7	Built-in RAM.
C-7	International country code.
C-8	Numeric keypad map.
C-8	System power specifications.
C-8	Baud rate table.
C-9	Initial font load.
C-10	TERM softkey string.
C-10	Keyboard character code maps.
C-11	System string addresses.
C-17	System strings.

```

        name      crom
        page     84,132

*****
CROM.ASM
*****
Portable PLUS Configuration EPROM
--U.S. version [Rev B: 6/20/1985]
*****



0000    CROM      SEGMENT BYTE
ASSUME CS:CROM,DS:CROM,ES:NOTHING,SS:NOTHING
ORG    0

; ######
; Portable PLUS
; #####
; CONFIGURATION EPROM
; #####
= 8000    CROMioaddr    equ    8000h    ; Config EPROM I/O base address
= 0300    KEYMAPorg    equ    0300h    ; HP and Alternate mode keyboard
                                         ; character code maps are assembled
                                         ; at this address, but do not appear
                                         ; in this listing.
= 0C00    STRINGorg    equ    0C00h    ; PAM and TERM strings are assembled
                                         ; at this address.
= 0000    NULL        equ    00h    ; Null character
= 000D    CR          equ    0Dh    ; Carriage return
= 000A    LF          equ    0Ah    ; Line feed
= 001B    ESC         equ    1Bh    ; ESCape

I0addr    macro    addr
local    temp
$call
temp = (offset &addr-offset $Header) shl 1 + CROMioaddr
$call
dw      temp
endm

.list

; ######
; PRODUCT NUMBER AND EDISC BOOT SECTOR
; #####
; This is the information that normally appears in a disc boot
; sector. It must be the first data in the Config EPROM.
; #####



0000  EB 1C 90    $Header      db      0EBh,01Ch,090h    ; (Jump around header)
0003  34 35 37 31 31 20      db      "45711"           ; Product number
                                         ; 20
000A  42          db      'B'       ; Config EPROM rev letter [6/20/85]
0008  0200        dw      512      ; D      Bytes per sector
000D  01          db      1       ; O      Sectors per allocation unit
000E  0001        dw      1       ; R      Reserved sectors
0010  01          db      1       ; N      Number of FATs
0011  0040        dw      64      ; O      Entries in root directory
0013  015F        dw      351      ; T      Number of sectors
0015  FA          db      0FAh    ; A      Media descriptor
0016  0005        dw      5       ; L      Number of FAT sectors
0018  0001        dw      1       ; T      Sectors per track
001A  0001        dw      1       ; E      Number of heads
001C  0000        dw      0       ; R      Number of hidden sectors
001E  F4          hlt
001F  FF          $Spare1    db      -1      ; -unused-

```

```

*****SERIAL NUMBER*****
This 32-byte string contains each
machine's unique serial number.
*****;

= 0020          $SerialNumber    equ      this byte
0020 0A [ FF ]  STRserialnum   db       10 dup (-1)      ; 10-byte Serial Number
002A 20 20          db       20h,20h      two spaces
002C 42 00          db       "B"          hardware version
002D 20 00          db       " "           U.S. designator ("space")
002E 00 00          db       0             zero terminated
002F 12 [ FF ]  serialspares   db       18 dup (-1)      ; -spares-
*****;
; INITIAL LIQUID CRYSTAL DISPLAY CONTRAST
*****;

0041 07          $Contrast      db      7      ; Range is 0 (darkest) to 15 (lightest)

*****CONFIGURATION EPROM BOOT CODE INFORMATION*****
; If the Bootflag is nonzero, the BootAddress and Bootlength
; words indicate the Config EPROM starting I/O address and
; length to be downloaded during system boot.
*****;

= 0042          $BootInfo      equ      this byte
0042 00          BOOTflag       db      0      ; 0=No Boot, 1=Boot
0043 0000        BOOTlength     dw      0      ; Boot code length (in bytes)
0045 +           BOOTaddress:  IOaddr    $BootInfo      ; Boot code I/O address
0045 8084          dw      ??0000

*****COUNTRY-SPECIFIC INFORMATION*****
32 bytes of country-specific data required by MS-DOS.
Tables (0) through (5) are loaded from ROM into RAM during
system boot; then the "$CountryTable" values below overlay
table (5) to allow for changes to one of these ROM tables.
For example, if the U.S. ROM table requires changes, then
"CurrentCountry" should be set to '5' (rather than '0') and
the desired "$CountryTable" values should be substituted.
*****;

0047 00          CurrentCountry db      0      ; Selects U.S. table
                                ; 0: US          3: France
                                ; 1: UK          4: Italy
                                ; 2: Germany     5: Spain (modified below)

= 0048          $CountryTable   equ this byte ; values below overlay table (5)
; For the US,UK,Fra,Ger,Ita & Spa Config EPROMs, table (5) remains
; Spain--unless one of these ROM tables requires changes.

0048 0001        LOCALtimedate  dw      1      ; Time & Date format (2 bytes)
                                                ; 0 - h:m:s m/d/y (US)
                                                ; 1 - h:m:s d/m/y (Europe)
                                                ; 2 - y/m/d h:m:s (Japan)
                                                ; Asciz currency (5 bytes)
004A 50 74 73 00 00  LOCALcurrency   db      'Pts',0,0
004F 2E 00          LOCALthousands db      ',',0
0051 2C 00          LOCALdecimal    db      ',',0
0053 2F 00          LOCALdate       db      ',',0
0055 2E 00          LOCALtime       db      ',',0
0057 03            LOCALbits       db      3      ; Currency symbol position (1 byte)
                                                ; 0 - symbol precedes amount, no space
                                                ; 1 - symbol follows amount, no space
                                                ; 2 - symbol precedes amount, one space
                                                ; 3 - symbol follows amount, one space
                                                ; Number of digits after decimal (1 byte)
                                                ; 12/24 hour flag (1 byte)
                                                ; 0 - 12 hour; 1 - 24 hour
0058 02            LOCALcents      db      2      ; Case mapper offset (4 bytes)
0059 01            LOCAL24hourclk  db      1      ; (do not change)
005A 08C8          LOCALcasemap    dw      08C8h
005C 0000          LOCALdatalist   db      0000h
005E 3B 00          LOCALspares     db      ';' ,0
0060 08 [ FF ]  LOCALspares   db      8 dup (-1) ; Round out to 32 bytes total

```

```
*****
|***** TIMER REFERENCE AND KEY REPEAT RATES *****|
| The one-second reference (3 bytes) is based on a master clock |
| frequency of 2.667 MHz, and is calculated as (2667000/6). |
|***** One Second = 444500 {06C854h} |
|***** 1/50 Second = 8890 {0022Bh} |
*****
```

= 0068	\$Timer50	equ	this byte	
0068 BA	One50thLow	db	0BAh	; 1/50 Second, low byte
0069 22	One50thMiddle	db	022h	; middle byte
006A 00	One50thHigh	db	000h	; high byte
= 006B	\$TimerTick	equ	this byte	
006B 76	HeartLow	db	76h	; Heartbeat (timer tick)
006C 60	HeartMiddle	db	60h	; interval reference
006D 00	HeartHigh	db	0	; (1/18 second)
006E 12	TicksPerSec	db	12h	; Heartbeat rate (18 ticks per second)
006F 1D	KeyRepRates	db	1Dh	Key repeat interval: --1st repeat interval (low nibble) (D ticks = 13/18 second) --succeeding repeats (high nibble) (1 tick = 1/18 second)

```
*****
|***** CONSOLE STATUS INDICATORS *****|
| These 16 bytes define the screen location of the console |
| status indicators (when active). Column numbers range |
| from 0 to 79; row numbers should be 0 or 1 (for the 1st |
| or 2nd line of the softkey label area). 'C' represents |
| 'caps lock on'; 'I' represents 'insert character on'; |
| and 'N' represents 'numeric keypad on'. |
*****
```

= 0070	\$Indicators	equ	this byte	
0070 00 54	INDcursorcol	db	0,42*2	; Cursor's column location displayed here
0072 00 48	INDcursorrow	db	0,36*2	; Cursor's row location displayed here
0074 43 00 4E	INDcapslock	db	'C',0,39*2	'caps lock' indicator on 1st line,col 39
0077 49 00 50	INDinschar	db	'I',0,40*2	'insert char' indicator on 1st line,col 40
0079 4E 01 48	INDnumpad	db	'N',1,36*2	'numeric pad' indicator on 2nd line,col 36
007D 01 4C	INDtimeofday	db	1,38*2	Time-of-day clock positioned here
007F ??	INDSpare	db	?	; -unused-

```

;***** PAM VARIABLE DEFAULTS *****
;***** PAM VARIABLE DEFAULTS *****
0080 2A      $PAMinfo      db    42   ; (Bytes that follow)
0081 04      PAM_partition db    4
0082 01      PAM_extdrives db    1
0083 01      PAM_verify     db    1
0084 00      PAM_powersave  db    0
0085 00      PAM_timeout    db    0
0086 00      PAM_cursor     db    0
0087 00      PAM_commode   db    0
0088 00      PAM_beep       db    0
0089 1F      PAMplotter    db    31
008A 1F      PAMprinter    db    31
008B 00      PAMprmode     db    0
008C 00      PAMprpitch    db    0
008D 00      PAMprspacing  db    0
008E 00      PAMprskip     db    0
008F 0D      PAMserbaud   db    13
0090 00      PAMserword   db    0
0091 00      PAMserstop   db    0
0092 00      PAMserparity db    0
0093 00      PAMserxon    db    0
0094 00      PAMsercts   db    0
0095 00      PAMserdsr   db    0
0096 00      PAMserdcld  db    0
0097 00      PAMserpower  db    0
0098 0D      PAM82164baud db    13
0099 00      PAM82164word  db    0
009A 00      PAM82164stop  db    0
009B 00      PAM82164parity db    0
009C 00      PAM82164xon   db    0
009D 00      PAM82164cts   db    0
009E 00      PAM82164dsr   db    0
009F 00      PAM82164dcld db    0
00A0 00      PAM82164power db    0
00A1 07      PAMmodbaud   db    7
00A2 00      PAMmodword   db    0
00A3 00      PAMmodstop   db    0
00A4 00      PAMmodparity db    0
00A5 00      PAMmodxon    db    0
00A6 00      PAMmodcts   db    0
00A7 00      PAMmoddsr   db    0
00A8 00      PAMmoddcld  db    0
00A9 00      PAMmodpower  db    0
00AA 00      PAMauxdev    db    0
00AB 11 [ FF ] PMspares   db    17 dup (-1) ; -unused-
00BC 00C8      AUX_RetryCount dw    200 ; Retry count for AUX driver (in multiples
                                                ; of 50 ms, so 200 = 10 seconds)
00BE 0F60      PAMaddr     dw    0F60h ; Address of PAM data from offset
                                            ; (used by diagnostics)

```

\*\*\*\*\*  
MUTE TABLES

The first word is the I/O address of an appropriate asciz string of characters (generally vowels) to be muted. The next six words are IOaddr pointers to mute tables for local function keycodes 20h-25h. Each mute table must have the same number of entries as the number of characters in the MuteTables list. The first seven IOaddr pointers apply to HP mode; the second seven IOaddr pointers apply to Alternate (ANSI) mode.

\*\*\*\*\*

= 00C0		\$MuteTableHP	equ	this byte
00C0 81B8	+		IOaddr	HPMutables
			dw	??0001
00C2 81D8	+		IOaddr	HPAccuteAccent
			dw	??0002
00C4 81F8	+		IOaddr	HPGraveAccent
			dw	??0003
00C6 8214	+		IOaddr	HPCircumflex
			dw	??0004
00C8 8232	+		IOaddr	HPUmlaut
			dw	??0005
00CA 8250	+		IOaddr	HPTilde
			dw	??0006
00CC 826E	+		IOaddr	HPMuteSpare
			dw	??0007
= 00CE		\$MuteTableIBM	equ	this byte
00CE 828C	+		IOaddr	IBMMutables
			dw	??0008
00D0 82AC	+		IOaddr	IBMAccuteAccent
			dw	??0009
00D2 82CA	+		IOaddr	IBMGraveAccent
			dw	??000A
00D4 82E8	+		IOaddr	IBMCircumflex
			dw	??000B
00D6 8306	+		IOaddr	IBMUmlaut
			dw	??000C
00D8 8324	+		IOaddr	IBMTilde
			dw	??000D
00DA 8342	+		IOaddr	IBMMuteSpare
			dw	??000E
			;	*****
			;	HP MODE MUTE TABLES
			;	*****
00DC 20 61 65 69 6E 6F 75 79 41 45 49 4E 4F 55 59 00		HPMutables	db	" aeinouyAEINOUY",0
00EC 27 C4 C5 D5 6E 00F1 C6 C7 C9 E0 DC 00F6 E5 4E E7 ED 59		HPAccuteAccent	db	039,196,197,213,110 ; aein
			db	198,199,121,224,220 ; ouyAE
			db	229,078,231,237,089 ; INOUY
00FB 60 C8 C9 D9 6E 0100 CA CB 79 A1 A3 0105 E6 4E E3 AD 59		HPGraveAccent	db	096,200,201,217,110 ; aein ^
			db	202,203,121,161,163 ; ouyAE
			db	230,078,232,173,089 ; INOUY
010A SE C0 C1 D1 6E 010F C2 C3 79 A2 A4 0114 A6 4E DF AE 59		HPCircumflex	db	094,192,193,209,110 ; aein ^
			db	194,195,121,162,164 ; ouyAE
			db	166,078,223,174,089 ; INOUY
0119 20 CC CD DD 6E 011E CE CF EF D8 A5 0123 A7 4E DB EE		HPUmlaut	db	032,204,205,221,110 ; aein
			db	206,207,238,216,165 ; ouyAE
			db	167,078,218,219,238 ; INOUY
0128 7E E2 65 69 B7 012D EA 75 79 E1 45 0132 49 B6 E9 55 59		HPTilde	db	126,226,101,105,183 ; aein ~
			db	234,117,121,225,069 ; ouyAE
			db	073,182,233,085,089 ; INOUY
0137 7E E2 65 69 B7 013C EA 75 79 E1 45 0141 49 B6 E9 55 59		HPMuteSpare	db	126,226,101,105,183 ; aein ~
			db	234,117,121,225,069 ; ouyAE
			db	073,182,233,085,089 ; INOUY

```

;*****;
; ALTERNATE MODE MUTE TABLES ;
;*****;

0146 20 61 65 69 6E 6F IBMMutables db "aeinouyAEINOUY",0
    75 79 41 45 49 4E
    4F 55 59 00

0156 27 A0 82 A1 6E IBMAccuteAccent db 027h,0A0h,082h,0A1h,06Eh ; aein
015B A2 A3 79 41 90 db 0B2h,0A3h,079h,041h,090h ; ouyAE
0160 49 4E 4F 55 59 db 049h,04Eh,04Fh,055h,059h ; INOUY

0165 60 85 8A 8D 6E IBMGraveAccent db 060h,085h,08Ah,08Dh,06Eh ; aein ^
016A 95 97 79 41 45 db 095h,097h,079h,041h,045h ; ouyAE
016F 49 4E 4F 55 59 db 049h,04Eh,04Fh,055h,059h ; INOUY

0174 5E 83 88 8C 6E IBMCircumflex db 05Eh,083h,088h,08Ch,06Eh ; aein ^
0179 93 96 79 41 45 db 093h,096h,079h,041h,045h ; ouyAE
017E 49 4E 4F 55 59 db 049h,04Eh,04Fh,055h,059h ; INOUY

0183 20 84 89 8B 6E IBMUmlaut db 020h,084h,089h,08Bh,06Eh ; aein
0188 94 81 79 8E 45 db 094h,081h,079h,08Eh,045h ; ouyAE
018D 49 4E 99 9A 59 db 049h,04Eh,099h,09Ah,059h ; INOUY

0192 7E 61 65 69 A4 IBMTilde db 07Eh,061h,065h,069h,0A4h ; aein ^
0197 6F 75 79 41 45 db 06Fh,075h,079h,041h,045h ; ouyAE
019C 49 A5 4F 55 59 db 049h,0A5h,04Fh,055h,059h ; INOUY

01A1 7E 61 65 69 A4 IBMmuteSpare db 07Eh,061h,065h,069h,0A4h ; aein
01A6 6F 75 79 41 45 db 06Fh,075h,079h,041h,045h ; ouyAE
01AB 49 A5 4F 55 59 db 049h,0A5h,04Fh,055h,059h ; INOUY

;*****;
; CONFIGURATION EPROM CHECKSUM ;
;*****;

= 01B0 $ROMinfo equ this byte
01B0 3F 3F 3F 3F 3F 2D ROMpartnumber db "?????-?????"; -Reserved-
    3F 3F 3F 3F 3F
01BB 41 30 ROMrevcode db 'A0'; -Reserved-
01BD 00 ROMchecksum dw 0FFFFh; Config EPROM checksum

;*****;
; BUILT-IN RAM
; This byte specifies how much RAM (excluding display memory
; and plug-in RAM drawers) is built into the machine.
;*****;

= 01C0 $RAMinfo equ this byte
01C0 01 RAMbuiltin db 1; Number of 128K byte blocks
01C1 0D [ FF ] RAMspares db 13 dup (-1); -unused-

;*****;
; INTERNATIONAL COUNTRY CODE ;
;*****;

01CE 18 $CountrySize db 24; Bytes of data in $CountryTable
01CF 22 $CountryCode db 34; $CountryCode remains 34 for all
                                ; Config EPROM language versions

```

```

***** NUMERIC KEYPAD MAP *****

This string of 32 bytes specifies the scancodes and keycodes
generated when the numeric keypad is active. The keycodes
have an implied attribute of 80h ("transmit character").
(See "Keyboard Operation" in Chapter 6.)
*****
```

<b>01D0</b>	2A 37 2B 38 31 39	\$NumPadMap	db	42,'7',43,'8',49,'9',50,'/'
	32 2F		db	35,'4',44,'5',52,'6',60,'*'
<b>01D8</b>	23 34 2C 35 34 36		db	37,'1',45,'2',53,'3',61,'-'
	3C 2A		db	47,'0',55,'.',54,'.',62,'+'
<b>01E0</b>	25 31 2D 32 35 33			
	3D 2D			
<b>01E8</b>	2F 30 37 2C 36 2E			
	3E 2B			

```

***** SYSTEM POWER SPECIFICATIONS *****

High byte then low byte for each entry, calculated as
actual current (millamps x 61.084). The battery %-age
on PAM's main screen is computed from these values.
*****
```

= 01F0		\$PowerSpecs	equ	this byte	; CURRENT REQUIREMENTS
<b>01F0</b>	26 2D	POWERrun	db	026h,02Dh	; Operating 160 ma
<b>01F2</b>	16 AB	POWERwait	db	016h,0ABh	; Wait .95 ma
<b>01F4</b>	00 23	POWERsleep	db	000h,023h	; Sleep .57 ma
<b>01F6</b>	05 F7	POWERdeepsleep	db	005h,0F7h	; Deep sleep .25 ma
<b>01F8</b>	05 0F	POWERrs232	db	005h,00Fh	; RS-232 21.2 ma
<b>01FA</b>	0A BD	POWERmodem	db	00Ah,0BDh	; Modem .45 ma
<b>01FC</b>	42 D0	POWERcharger	db	042h,0D0h	; Charger input 280 ma
<b>01FE</b>	01 AC	POWERramactive	db	001h,0ACh	; RAM active .7 ma
<b>0200</b>	00 05	POWERramstdby	db	000h,005h	; RAM standby .09 ma
<b>0202</b>	01 01	POWERromactive	db	001h,001h	; ROM active 4.2 ma
<b>0204</b>	00 00	POWERromstdby	db	000h,000h	; ROM standby .003 ma
<b>0206</b>	FF FF	POWERspare	db	-1,-1	; -unused-

```

***** BAUD RATE TABLE *****

Each 3-byte entry <Code,LSB,MSB> specifies a baud
rate based on a 2.667 MHz master clock. The codes
correspond to HP-UX (tm) city codes.
*****
```

= 0208		\$BaudRates	equ	this byte	
<b>0208</b>	00 00 00		db	0,0,0	.
<b>020B</b>	01 2A 68		db	1,42,104	b0
<b>020E</b>	02 71 45		db	2,113,69	b50
<b>0211</b>	03 58 2F		db	3,88,47	b75
<b>0214</b>	04 DD 26		db	4,221,38	b110
<b>0217</b>	05 88 22		db	5,184,34	b134
<b>021A</b>	06 5B 11		db	6,91,17	b150
<b>021D</b>	07 AD 08		db	7,173,8	b300
<b>0220</b>	08 56 04		db	8,86,4	b600
<b>0223</b>	09 E4 02		db	9,228,2	b1200 (default)
<b>0226</b>	0A 2B 02		db	10,43,2	b1800
<b>0229</b>	0B 71 01		db	11,113,1	b2400
<b>022C</b>	0C 15 01		db	12,21,1	b3600
<b>022F</b>	0D B8 00		db	13,184,0	b4800
<b>0232</b>	0E 8A 00		db	14,138,0	b7200
<b>0235</b>	0F 44 00		db	15,68,0	b9600
<b>0238</b>	1E 00 00		db	30,0,0	b19200
<b>023B</b>	1F 00 00		db	31,0,0	external a
<b>023E</b>	FFFF		dw	-1	external b

; END OF TABLE

```

;*****INITIAL FONT LOAD*****
;INITIAL FONT LOAD

FONTsegment is the system ROM segment address where the
default font tables start. The load table follows, and
each entry is four bytes long. The first two bytes give
a font entry's ID (its "name"). The next byte indicates
where the font table begins in the ROM font segment (0 =
first set of 128 chars, 1 = second set of 128 chars, and
so on). The fourth byte defines the size of the font
(0 = 128 characters, and 1 = 256 characters).

As the table is processed, display RAM font areas are
filled in order, starting with the low half of Area 0
and ending with the high half of Area 2.
*****
```

= 0240	\$FontInfo	equ	this byte				
0240 FE00	FONTsegment	dw	0FE00h	; Font table segment address			
				<table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td style="padding: 2px;">FONT ID</td> <td style="padding: 2px;">ROM BLOCK</td> <td style="padding: 2px;">0=128 1=256</td> </tr> </table>	FONT ID	ROM BLOCK	0=128 1=256
FONT ID	ROM BLOCK	0=128 1=256					
				Neither table may exceed 6*128 characters (extras are ignored!)			
				\/			

0242 08 55 00 01	HPFONTbase	db	8,'U', 0, 1	HP Bold (256 chars) #1
0246 00 4C 08 00		db	0,'L', 8, 0	Line Draw (128 chars) #2
024A 00 4D 09 00		db	0,'M', 9, 0	Math Set (128 chars) #3
024E 09 55 02 01		db	9,'U', 2, 1	HP Thin (256 chars) #4
0252 FF FF FF FF		db	-1, -1, -1, -1	-empty- #5
0256 FF FF FF FF		db	-1, -1, -1, -1	-empty- #6
025A FF		db	-1	, END OF HP TABLE (6*128 entries max)
025B 08 41 04 01	ALTFONTbase	db	8,'A', 4, 1	ALT Bold (256 chars) #1
025F 08 FF 04 01		db	8,-1, 4, 1	ALT Bold (again) #2
0263 09 41 06 01		db	9,'A', 6, 1	ALT Thin (256 chars) #3
0267 FF FF FF FF		db	-1, -1, -1, -1	-empty- #4
026B FF FF FF FF		db	-1, -1, -1, -1	-empty- #5
026F FF FF FF FF		db	-1, -1, -1, -1	-empty- #6
0273 FF		db	-1	, END OF ALT TABLE (6*128 entries max)

```

C           include strterm.crm
C           .list
C ****
C TERM SOFTKEY STRING
C ****
C STRTERM.CRM [Rev 3/13/1985]
C
C This asciz string (ioaddr STRING NUMBER 182 of file STRADDR.CRM)
C contains softkey labels for the TERM program. Each label must
C contain 16 characters, and the last character must be a space.
C ****
C
C 0274 20 20 46 72 6F 6D C STRsoftkeys db " From "
C 20 20
C 027C 20 20 48 6F 73 74 C db " Host "
C 20 20
C 0284 20 20 20 54 6F 20 C db " To "
C 20 20
C 028C 20 20 48 6F 73 74 C db " Host "
C 20 20
C 0294 20 46 69 6C 65 20 C db " File "
C 20 20
C 029C 20 4E 61 6D 65 73 C db " Names "
C 20 20
C 02A4 20 52 65 6D 6F 74 C db " Remote "
C 65 20
C 02AC 20 20 4D 6F 64 65 C db " Mode "
C 20 20
C 02B4 20 20 41 75 74 6F C db " Auto "
C 20 20
C 02BC 20 20 20 4C 46 20 C db " LF "
C 20 20
C 02C4 20 4C 6F 63 61 6C C db " Local "
C 20 20
C 02CC 20 45 63 68 6F 20 C db " Echo "
C 20 20
C 02D4 44 69 73 70 6C 61 C db "Display "
C 79 20
C 02DC 46 75 6E 63 74 6E C db "Functns "
C 73 20
C 02E4 20 20 20 20 20 20 C db " "
C 20 20
C 02EC 20 20 45 78 69 74 C db " Exit "
C 20 20
C 02F4 00                 C db 0 ; zero terminated

```

.list

```

C ****
C KEYBOARD CHARACTER CODE MAPS
C ****
C HP and Alternate mode keyboard character
C code maps are assembled at this address,
C but do not appear in this listing. (See
C the character code tables in Chapter 13
C and "Keyboard Operation" in Chapter 6.)
C ****

```

org KEYMAPorg

```

0C00          org      STRINGorg
C           include straddr.crm
C           .list
C           ;*****SYSTEM STRING ADDRESSES*****
C           ; SYSTEM STRING ADDRESSES
C           ; STRADDR.CRM [Rev 3/13/1985]
C           ; Each of these is an I/O address pointer to a corresponding asciz
C           ; (i.e., null-terminated) string of STRINGS.CRM or STRTERM.CRM
C           ; An I/O address pointer is the word offset (low byte, high byte)
C           ; multiplied by two and added to 8000h. The STRING NUMBER which
C           ; identifies each string follows the comment delimiting semicolon.
C           ;*****SYSTEM STRING ADDRESSES*****
= 0C00        C $StringPtrs    equ     this byte
=
C STRProgExited   equ     STRNull      ; string 7 is not used
C ; System strings                                ; STRING NUMBER
0C00 9B00       C+ ioaddr  STRMachineID ; 0 - "45711"
0C02 8040       C+ ioaddr  STRSerialNum ; 1 - Machine serial number
0C04 9AF8       C+ ioaddr  STRSelectKey ; 2 - Select key default: ESC,"&P"
0C06 9B0C       C+ ioaddr  STRMemLost  ; 3 - " WARNING: Memory low! Hit [f1]"
0C08 9C66       C+ ioaddr  STRReformatA; 4 - " Reformatting drive A: ... all
0C0A 9CC6       C+ ioaddr  STRLowBattery; 5 - " Low battery! "
C ; HP Link Messages
0C0C 9CE4       C+ ioaddr  STRAnyKeyExit ; 6 - " Press any key to exit this pro
0C0E 9AFE        C+ ioaddr  STRProgExited ; 7 - Equated to STRNull (not used)
0C10 9D2E       C+ ioaddr  STRCantFormat; 8 - "Disks cannot be formatted using
0C12 9AFE        C+ ioaddr  STRNull      ; 9 - Null string
C ; PAM Messages
0C14 9E66       C+ ioaddr  STRpam_main_msg ; 10 - " Personal Applications Manager
0C16 9EC8       C+ ioaddr  STRpam_sel_msg ; 11 - "Move the pointer to the desired
0C18 9F68       C+ ioaddr  STRreread_msg ; 12 - " Reading all inserted discs to
0C1A 9FE0       C+ ioaddr  STRload1_msg ; 13 - ' Loading '
0C1C 9FF6       C+ ioaddr  STRpam_nxt   ; 14 - "Press [Next]/[Prev] to see more
0C1E A052       C+ ioaddr  STRpam_exit  ; 15 - " Type 'exit' to return to P.A.M
0C20 A096       C+ ioaddr  STRno_print ; 16 - "Background print can be used
0C22 A10C       C+ ioaddr  STRno_set   ; 17 - "This command can only be used
0C24 A1A4       C+ ioaddr  STRpam_max  ; 18 - "Maximum number of applications
0C26 A1F8       C+ ioaddr  STRfree_msg ; 19 - " bytes free on "
C
0C28 A218       C+ ioaddr  STRenv_too_big ; 20 - "PAM.ENV file too long."
0C2A A278       C+ ioaddr  STRparam    ; 21 - "Parameter"
0C2C A28C       C+ ioaddr  STRserial   ; 22 - "Serial"
0C2E A9EC       C+ ioaddr  STRmvptror ; 23 - "Move the pointer to, or type

```

**System String Addresses**

0C30	A2A6	C	ioaddr STRpress_msg ; 24 - "Press Start if information is
		C+	ioaddr STRmodem ; 25 - "Modem"
0C32	A29A	C	ioaddr STRdir_print ; 26 - "Printing directory."
		C+	ioaddr STRpls_wait_msg ; 27 - " Please wait."
0C34	A308	C	ioaddr STRfile_print ; 28 - "Printing file."
		C+	ioaddr STRcntmes ; 29 - "Printer not ready."
0C36	A330	C	
		C+	
0C38	A34E	C	
		C+	
0C3A	9E40	C	
		C+	
0C3C	A36C	C	ioaddr STRdoesnt_exist ; 30 - "File or directory does not
		C+	ioaddr STRdir_delete ; 31 - "Deleting directory."
0C3E	A3B0	C	ioaddr STRdir_del_err ; 32 - "Directory is not empty, or is
		C+	ioaddr STRfile_delete ; 33 - "Deleting file."
0C40	A3D8	C	ioaddr STRfile_wrt_prt ; 34 - "File or directory is write pro
		C+	ioaddr STRdir_make ; 35 - "Making directory."
0C42	A468	C	ioaddr STRmkdir_err ; 36 - "Directory could not be created."
		C+	
0C44	A486	C	ioaddr STRdir_choose ; 37 - "Choosing directory."
		C+	
0C46	A4D2	C	ioaddr STRdisc_format ; 38 - "Formatting disc."
		C+	
0C48	R4F6	C	ioaddr STRtoo_long ; 39 - "Error: line too long."
		C+	
0C4A	A536	C	
		C+	
0C4C	A55E	C	
		C+	
0C4E	A580	C	
		C+	
0C50	A5AC	C	ioaddr STRno_format_b ; 40 - "Cannot format drive B: (read
		C+	ioaddr STRfile_copy ; 41 - "Copying specified file(s)."
0C52	A5FE	C	ioaddr STRfm_press_any ; 42 - "Press any key to return to P.A.M.
		C+	ioaddr STRno_dest ; 43 - "No destination file specified."
0C54	A634	C	ioaddr STRfile_rename ; 44 - "Renaming file."
		C+	
0C56	A67A	C	ioaddr STRalready_ex ; 45 - "New name already exists, is a
		C+	ioaddr STRenter_wcard ; 46 - "Enter new wild card and press
0C58	A6B8	C	ioaddr STRno_wild ; 47 - "No wild cards allowed in this
		C+	ioaddr STRuse_wild ; 48 - "More files in directory than
0C5A	A6D6	C	ioaddr STRdisp_dir ; 49 - "Displayed dir: "
		C+	
0C5C	A75A	C	
		C+	
0C5E	A7AA	C	
		C+	
0C60	A82E	C	
		C+	
0C62	A8AA	C	
		C+	
0C64	A8CA	C	ioaddr STRfm_nxt ; 50 - "Press [Next]/[Prev] to see more
		C+	ioaddr STRno_files ; 51 - "The directory contains no files
0C66	A918	C	ioaddr STRcompany ; 52 - "Hewlett-Packard"
		C+	
0C68	A95A	C	ioaddr STRbattery ; 53 - " Battery"
		C+	
0C6A	A97A	C	ioaddr STRfmgrmsg ; 54 - " File Manager"
		C+	
0C6C	A994	C	ioaddr STRfmain ; 55 - "Main"
		C+	
0C6E	A9B0	C	ioaddr STRselfunc ; 56 - "Select a function."
		C+	
0C70	A9BA	C	ioaddr STRfmprint ; 57 - "Print"
		C+	
0C72	A9E0	C	ioaddr STRtoprint ; 58 - "file or directory to print."
		C+	
0C74	AA4E	C	ioaddr STRprfile ; 59 - "Print file: "
		C+	
0C76	AA86	C	
		C+	
0C78	AAA0	C	ioaddr STRfmdel ; 60 - "Delete"
		C+	
		C	ioaddr STRtodel ; 61 - "file or directory to delete."

### System String Addresses

0C7A	RAAE	C+	ioaddr	STRdelfile	; 62 - "Delete file: "
0C7C	RAE8	C+	ioaddr	STRfmmak	; 63 - "Make Directory"
0C7E	AB04	C+	ioaddr	STRtomake	; 64 - "Type the new directory name."
0C80	AB22	C+	ioaddr	STRmakefile	; 65 - "Directory to make: "
0C82	AB5C	C+	ioaddr	STRfmchs	; 66 - "Choose Directory"
0C84	AB84	C+	ioaddr	STRtochs	; 67 - "directory to display."
0C86	ABA6	C+	ioaddr	STRchsfle	; 68 - "Directory to display: "
0C88	ABD2	C+	ioaddr	STRfmfor	; 69 - "Format" "
0C8A	AC00	C+	ioaddr	STRtofor	; 70 - "Enter the disc to format. (All
0C8C	AC0E	C+	ioaddr	STRLabPrmt	; 71 - "Volume label (11 characters,
0C8E	9D8A	C+	ioaddr	STRdrive	; 72 - "Drive to format: "
0C90	AC90	C+	ioaddr	STRvolume	; 73 - "Volume label: "
0C92	ACB4	C+	ioaddr	STRfmcp	; 74 - "Copy"
0C94	ACD2	C+	ioaddr	STRcpypfr	; 75 - "file to copy."
0C96	ACDC	C+	ioaddr	STRcpypo	; 76 - "destination file."
0C98	ACF8	C+	ioaddr	STRfrom	; 77 - "Copy from file: "
0C9A	AD1C	C+	ioaddr	STRto	; 78 - "Copy to file: "
0C9C	AD3E	C+	ioaddr	STRfmren	; 79 - "Rename"
0C9E	AD5C	C+	ioaddr	STRtoren	; 80 - "file to rename."
0CA0	AD6A	C+	ioaddr	STRname	; 81 - "Type the new file name."
0CA2	AD8A	C+	ioaddr	STRoldnm	; 82 - "Rename file: "
0CA4	ADBE	C+	ioaddr	STRnewnm	; 83 - "Rename file to: "
0CA6	ADDA	C+	ioaddr	STRsyscnf	; 84 - " System Configuration"
0CA8	ADFC	C+	ioaddr	STRcomcnf	; 85 - " Datacom Configuration"
0CAA	AE28	C+	ioaddr	STRclkcnf	; 86 - " Time and Date"
0CAC	AE56	C+	ioaddr	STRalarm	; 87 - " Alarm "
0CAE	AE74	C+	ioaddr	STRmemEdisc	; 88 - "Main Memory / Edisc"
0CB0	AE84	C+	ioaddr	STRextdisc	; 89 - "External Disc Drives"
0CB2	AEAC	C+	ioaddr	STRdiscVer	; 90 - "Disc Write Verify"
0CB4	AED6	C+	ioaddr	STRpwrSav	; 91 - "Power Save Mode"
0CB6	AEFA	C+	ioaddr	STRdisptime	; 92 - "Disp Timeout (min)"
0CB8	AF1A	C+	ioaddr	STRcursor	; 93 - "Cursor Type"
0CBA	AF46	C+	ioaddr	STRconsole	; 94 - "Console Mode"
0CBC	AF5E	C+	ioaddr	STRbeepcnf	; 95 - "Tone Duration"
0CBE	AF78	C+	ioaddr	STRPLTdev	; 96 - "Plotter Interface"
JCC0	AF94	C+	ioaddr	STRprinter	; 97 - "Printer Mode"
0CC2	AFB8	C+	ioaddr	STRPRNdev	; 98 - "Printer Interface"
0CC4	AFD2	C+			

**System String Addresses**

0CC6	AFF6	C	ioaddr	STRpitch	; 99 - "Printer Pitch"
		C+			
0CC8	B012	C	ioaddr	STRlinespc	;100 - "Printer Line Spacing"
		C+	ioaddr	STRskipperf	;101 - "Printer Skip Perforation"
0CCA	B03C	C	ioaddr	STRAuxCnf	;102 - "Datacom Interface"
		C+	ioaddr	STRbaudRate	;103 - "Transmission Rate (BPS)"
0CCC	B06E	C	ioaddr	STRwordlen	;104 - "Word Length (bits)"
		C+	ioaddr	STRstopbits	;105 - "Stop Bits"
0CCE	B092	C	ioaddr	STRparity	;106 - "Parity"
		C+	ioaddr	STRxonxoff	;107 - "XON/XOFF Pacing"
0CD0	B0C2	C	ioaddr	STRctsline	;108 - "CTS Line"
		C+	ioaddr	STRdsrline	;109 - "DSR Line"
0CD4	B0FC	C			
		C+	ioaddr	STRdcpline	;110 - "DCD Line"
0CD6	B10A	C	ioaddr	STRpower	;111 - "Power to Interface"
		C+	ioaddr	STRon	;112 - "On "
0CD8	B12A	C	ioaddr	STRoff	;113 - "Off"
		C+	ioaddr	STRundersc	;114 - "Underscore"
0CDA	B13C	C	ioaddr	STRboxc	;115 - "Box"
		C+	ioaddr	STRlongbeep	;116 - "Long "
0CDC	B14E	C	ioaddr	STRshortbeep	;117 - "Short"
		C+	ioaddr	STRgraphalph	;118 - "Alpha and HP Graphics"
0CDE	B160	C	ioaddr	STRgraph	;119 - "HP Graphics Only"
		C+			
0CE0	B186	C	ioaddr	STRalph	;120 - "Alpha Only"
		C+	ioaddr	STRnoconf	;121 - "No Configuration"
0CE2	B18E	C	ioaddr	STRnorm	;122 - "Normal"
		C+	ioaddr	STRexpan	;123 - "Expanded"
0CE4	B196	C	ioaddr	STRcompr	;124 - "Compressed"
		C+	ioaddr	STRexpcompr	;125 - "Expanded-Compressed"
0CE6	B1AC	C	ioaddr	STReven	;126 - "Even"
		C+	ioaddr	STRodd	;127 - "Odd "
0CE8	B1B4	C	ioaddr	STRnone	;128 - "None"
		C+	ioaddr	STRignore	;129 - "Ignore "
0CEA	B1C0	C			
		C+	ioaddr	STRobserve	;130 - "Observe"
0CEC	B1CC	C	ioaddr	STR82164	;131 - "HP 82164A"
		C+	ioaddr	STR6lines	;132 - "6 lines per inch"
0CEE	B1F8	C	ioaddr	STR8lines	;133 - "8 lines per inch"
		C+	ioaddr	STRhpil	;134 - "HP-IL"
0CF0	B21A	C	ioaddr	STRHP	;135 - "HP"
		C+	ioaddr	STRalt	;136 - "Alternate"
0CF2	B230	C			
0CF4	B252	C			
0CF6	B260	C			
0CF8	B272	C			
0CFA	B288	C			
0CFB	B280	C			
0CFE	B2BA	C			
0D00	B2C4	C			
0D02	B2CE	C			
0D04	B2DE	C			
0D06	B2EE	C			
0D08	B302	C			
0D0A	B324	C			
0DOC	B346	C			
0D0E	B352	C			

### System String Addresses

0D10 B358	C+	ioaddr STRtimezone	;137 - "Time Zone"
0D12 B36C	C+	ioaddr STRhour	;138 - "Hour"
0D14 B380	C+	ioaddr STRminutes	;139 - "Minutes"
0D16 B38A	C+		
	C	ioaddr STRseconds	;140 - "Seconds"
0D18 B39A	C+	ioaddr STRmonth	;141 - "Month"
0D1A B3AA	C+	ioaddr STRday	;142 - "Day"
0D1C B3B6	C+	ioaddr STRyear	;143 - "Year"
0D1E B3BE	C+	ioaddr STRreading	;144 - "Error reading drive "
0D20 90EC	C+	ioaddr STRwriting	;145 - "Error writing drive "
0D22 9E16	C+		
	C ; PAM Softkey Labels		
0D24 B3C8	C+	ioaddr SFKblank	;146 - " " "
0D26 B3EA	C+	ioaddr SFKhelp	;147 - " " " Help "
0D28 B40C	C+	ioaddr SFKchdir	;148 - " Choose " " Dir "
0D2A B42E	C+	ioaddr SFKstover	;149 - " Start " " Over "
0D2C B494	C+	ioaddr SFKmainkeys	;150 - " Start " " Applic "
0D2E B4B6	C+	ioaddr SFKmainf2	;151 - " File " " Manager "
0D30 B4D8	C+	ioaddr SFKmainf3	;152 - " Time & " " Date "
0D32 B4FA	C+	ioaddr SFKmainf4	;153 - " Reread " " Discs "
0D34 B51C	C+	ioaddr SFKmainf5	;154 - " Datacom " " Config "
0D36 B53E	C+	ioaddr SFKmainf6	;155 - " System " " Config "
0D38 B560	C+	ioaddr SFKmainf8	;156 - " " " Off "
0D3A B582	C+	ioaddr SFKcnff3	;157 - " Next " " Choice "
0D3C B5A4	C+	ioaddr SFKcnff4	;158 - " Previous " " Choice "
0D3E B5C6	C+	ioaddr SFKcnff5	;159 - " Default " " Values "
0D40 B450	C+	ioaddr SFKexit	;160 - " " " Exit "
0D42 B472	C+	ioaddr SFKstart	;161 - " " " Start "
0D44 B5E8	C+	ioaddr SFKstprn	;162 - " Stop " " Print "
0D46 B60A	C+	ioaddr SFKdeff1	;163 - " Copy 1 " " char "
0D48 B62C	C+	ioaddr SFKdeff2	;164 - "Copy up " " to char"

System String Addresses

<b>0D4A</b>	<b>B64E</b>	C	ioaddr SFKdeff3	;165 - " Copy "
		C+		;166 - " all " " Skip 1 "
<b>0D4C</b>	<b>B670</b>	C	ioaddr SFKdeff4	;167 - "Skip up " " char "
		C+		;168 - " to char " " Void "
<b>0D4E</b>	<b>B692</b>	C	ioaddr SFKdeff5	;169 - " Toggle " " input "
		C+		;170 - " insert " " New "
<b>0D50</b>	<b>B6B4</b>	C	ioaddr SFKdeff6	;171 - " Print " " line "
		C+		;172 - "File/Dir" " Delete "
<b>0D52</b>	<b>B6D6</b>	C	ioaddr SFKdeff7	;173 - " Make " "File/Dir"
		C+		;174 - " Dir " "
<b>0D54</b>	<b>B6F8</b>	C	ioaddr SFKfmfgf1	;175 - " Copy " " Format "
		C+		;176 - " File " " Rename "
<b>0D56</b>	<b>B71A</b>	C	ioaddr SFKfmfgf2	;177 - " Set " " File "
		C+		;178 - "Wildcard"
<b>0D58</b>	<b>B73C</b>	C	ioaddr SFKfmfgf3	;179 - " Disc write failure." ;Not softkey
		C+		;180 - "Setting" ;Not softkey
<b>0D5A</b>	<b>B75E</b>	C	ioaddr SFKfmfgf5	;181 - " No "
		C+		;182 - " Change "
<b>0D5C</b>	<b>B780</b>	C	ioaddr SFKfmfgf6	;183 - " DOS Commands " ;Not softkey
		C+		C ; TERM Softkey Labels and Messages
<b>0D5E</b>	<b>B7A2</b>	C	ioaddr STRwrite_fail	C+ ;184 - All eight softkey labels:
		C+		" From " " To "
<b>0D60</b>	<b>B7C4</b>	C	ioaddr STRcur_set	" Host " " Host "
		C+		" File " " Remote "
<b>0D62</b>	<b>B7E6</b>	C	ioaddr STRcancel	" Names " " Mode "
		C+		" Auto " " Local "
<b>0D64</b>	<b>B808</b>	C	ioaddr STRdos_com	" LF " " Echo "
		C+		"Display " " Exit "
<b>0D66</b>	<b>B854</b>	C		C ;185 - " FROM HOST to file: " and
		C+		" TO HOST from file: "
<b>0D68</b>	<b>B832</b>	C		;186 - "termlog"
		C+		;187 - " Cannot open AUX."
<b>0D6A</b>	<b>B864</b>	C		;188 - " Insufficient RAM."
		C+		;189 - ' Cannot access ''
				C \$LastMssg
			equ	this byte

```

C           include strings.crm
C           .list
C ****
C ; SYSTEM STRINGS
C ****
C ; STRINGS.CRM [Rev 6/17/1985]
C ;
C ; These PAM and TERM asciz strings are referenced
C ; by ioaddr pointers in file STRADDR.CRM.
C ****
C ; $StringTexts    equ      this byte
C ;
C ; If the Memomaker editor is used, then GET and SAVE this
C ; file as an 'ASCII File' (rather than 'Document File')
C ; to preserve non-usascii Roman8 characters.
C ;
C ; The number in brackets [ ] following a message refers to
C ; its ioaddr STRING NUMBER in file STRADDR.CRM.
C ;
C ; No message line may exceed 80 characters. Be careful to
C ; preserve leading and trailing spaces exactly as shown.
C ; Additional restrictions on individual messages are noted.
C ;
C ; If CAUTION! appears after a message, then any non-usascii
C ; character in that message will be displayed incorrectly if
C ; "Alternate" Console Mode has been selected.
C ;
C ; ****
C ; System Messages
C ; ****
OD7C 1B 26 50      C ; STRSelectKey db ESC,"&P" ; Do not change.
OD7F 00              C ; [ 2]
OD80 34 35 37 31 31 00   C ; STRNull db 0 ; Do not change.
C ; [ 9]
OD86 20 57 41 52 4E 49   C ; STRMachineID db "45711",0 ; Do not change.
C ; [ 0]
OD8E 4E 47 3A 20 4D 65   C ; STRMemLost db " WARNING: Memory Lost! Press [f1] to reformat drive A: "
4E 47 3A 20 4D 65
6D 6F 72 79 20 4C
6F 73 74 21 20 20
50 72 65 73 73 20
58 66 31 50 20 74
6F 20 72 65 66 6F
72 6D 61 74 20 64
72 69 76 65 20 41
3A 20
ODBE 0D 0A
ODCO 20 28 64 65 73 74   C ; db CR,LF
72 6F 79 69 20 6E 67
20 64 61 74 61 29
20 6F 72 20 70 72
65 73 73 20 74 68
65 20 73 70 61 63
65 20 62 61 72 20
74 6F 20 63 6F 6E
74 69 6E 75 65 20
20 20
ODF8 0D 0A
ODFA 20 77 69 74 68 6F   C ; db CR,LF
75 74 20 72 65 66
6F 72 6D 61 74 74
69 6E 67 20 28 64
61 74 61 20 65 72
72 6F 72 73 20 6F
6E 20 41 3A 20 6D
61 79 20 72 65 73
75 6C 74 29 2E 20
20 20 00
; [ 3] ; Each of the three lines must be exactly the same length.
STRReformatA db " Reformatting Drive A: ... all data destroyed. ",0
OE33 20 52 65 66 6F 72   C
60 61 74 74 69 6E
67 20 44 72 69 76
65 20 41 3A 20 7E
2E 2E 20 61 6C 6C
20 64 61 74 61 20
64 65 73 74 72 6F
79 65 64 2E 20 00
; [ 4]

```

**System Strings**

```

OE63 20 4C 6F 77 20 42 C STRLowBattery db " Low Battery! ",0 ; Max 31 chars.
      61 74 74 65 72 79 C ; [ 5] CAUTION!
      C ;*****;
      C ; HP Link Messages ;
      C ;*****;

OE72 20 50 72 65 73 73 C STRAnyKeyExit db " Press any key to exit this program.",0
      20 61 6E 79 20 6B C
      65 79 20 74 6F 20 C
      65 78 69 74 20 74 C
      68 69 73 20 70 72 C
      6F 67 72 61 6D 2E C
      00 C ; [ 6] CAUTION!

OE97 44 69 73 68 73 20 C STRCantFormat db "Disks cannot be formatted using this program.",0
      63 61 6E 6E 6F 74 C
      20 62 65 20 66 6F C
      72 6D 61 74 74 65 C
      64 20 75 73 69 6E C
      67 20 74 68 69 73 C
      20 70 72 6F 67 72 C
      61 6D 2E 00 C ; [ 8] CAUTION!
      C ;*****;
      C ; PAM Messages ;
      C ;*****;

OECS 56 6F 6C 75 6D 65 C STRLabPrmt db "Volume label (11 characters, [Return] for none)?",0
      20 6C 61 62 65 6C C
      20 28 31 31 20 63 C
      68 61 72 61 63 74 C
      65 72 73 20 20 5B C
      S2 65 74 75 72 6E C
      5D 20 66 6F 72 20 C
      6E 6F 6E 65 29 3F C
      00 C ; [71] CAUTION!

OEF6 45 72 72 6F 72 20 C STRreading db "Error reading drive ",0
      72 65 61 64 69 6E C
      67 20 64 72 69 76 C
      65 20 00 C ; [144]

OF08 45 72 72 6F 72 20 C STRwriting db "Error writing drive ",0
      77 72 69 74 69 6E C
      67 20 64 72 69 76 C
      65 20 00 C ; [145]

OF20 50 72 69 6E 74 65 C STRcntmes db "Printer not ready.",0
      72 20 6E 6F 74 20 C
      72 65 61 64 79 2E C
      00 C ; [29]

OF33 20 50 65 72 73 6F C STRpam_main_msg db " Personal Applications Manager (P.A.M.) Main",0
      6E 61 6C 20 41 70 C
      70 6C 69 63 61 74 C
      69 6F 6E 73 20 4D C
      61 6E 61 67 65 72 C
      20 28 50 2E 41 2E C
      4D 2E 29 20 20 20 C
      20 20 4D 61 69 6E C
      00 C ; [10] ; Maximum 52 characters

OF64 4D 6F 76 65 20 74 C STRpam_sel_msg db "Move the pointer to the desired application, then "
      68 65 20 70 6F 69 C
      6E 74 65 72 20 74 C
      6F 20 74 68 65 20 C
      64 65 73 69 72 65 C
      64 20 61 70 70 6C C
      69 63 61 74 69 6F C
      6E 2C 20 74 68 65 C
      6E 20 C

OF96 70 72 65 73 73 20 C db "press ",27,"&d Start Applic ",27,"&d@.",0
      1B 26 64 42 20 53 C
      74 61 72 74 20 41 C
      70 70 6C 69 63 20 C
      1B 26 64 40 2E 00 C ; [11] ; Escape sequence places Start Applic in inverse video.

```

System Strings

```

0FB4 20 52 65 61 64 69 C STRreread_msg db " Reading all inserted discs to find installed "
6E 67 20 61 6C 6C C
20 69 6E 73 65 72 C
74 65 64 20 64 69 C
73 63 73 20 74 6F C
20 66 69 6E 64 20 C
69 66 73 74 61 6C C
6C 65 64 20 C
61 70 70 6C 69 63 C
61 74 69 6F 6E 73 C
2E 00 C
; [12]

0FE2 20 4C 6F 61 64 69 C
6E 67 20 22 00 C
STRload1_msg db ' Loading ',0
; [13] CAUTION!
STRpam_nxt db "Press [Next]/[Prev] to see more applications.",0

0FFB 50 72 65 73 73 20 C
5B 4E 65 78 74 5D C
2F 5B 50 72 65 76 C
5D 20 74 6F 20 73 C
65 65 20 6D 6F 72 C
65 20 61 70 70 6C C
69 63 61 74 69 6F C
6E 73 2E 00 C
; [14]

1029 20 54 79 70 65 20 C
27 65 78 69 74 27 C
20 74 6F 20 72 65 C
74 75 72 6E 20 74 C
6F 20 50 2E 41 2E C
4D 2E 20 00 C
; [15] CAUTION!

104B 54 68 69 73 20 63 C
6F 6D 61 6E 64 C
20 63 61 6E 20 6F C
6E 6C 79 20 62 65 C
20 75 73 65 64 20 C
66 72 6F 6D 20 C
106E 1B 26 64 42 20 44 C
4F 53 20 43 6F 6D C
6D 61 6E 64 73 20 C
1B 26 64 40 2E 00 C
; [16] CAUTION!
; Escape sequence places " DOS Commands " in inverse video.

1086 50 41 54 48 73 2C C
20 50 52 4F 4D 50 C
54 73 2C 20 61 6E C
64 20 65 6E 76 69 C
72 6F 6E 6D 65 6E C
74 20 76 61 72 69 C
61 62 6C 65 73 20 C
6D 75 73 74 20 62 C
65 20 73 65 74 20 C
75 73 69 6E 67 20 C
61 20 50 41 4D 2E C
45 4E 56 20 66 69 C
6C 65 2E 00 C
; [17] CAUTION!

10D2 4D 61 78 69 6D 75 C
6D 20 6E 75 6D 62 C
65 72 20 6F 66 20 C
61 70 70 6C 69 63 C
61 74 69 6F 6E 73 C
20 69 6E 73 74 61 C
6C 6C 65 64 2E 00 C
; [18]

10FC 20 62 79 74 65 73 C
20 66 72 65 65 20 C
6F 6E 20 00 C
; [19]

```

System Strings

```

110C 50 41 4D 2E 45 4E C STRenv_too_big db "PAM.ENV file too long (maximum 256 characters).",0
56 20 66 69 6C 65 C
20 74 6F 6F 20 6C C
6F 6E 67 20 28 6D C
61 78 69 6D 75 6D C
20 32 35 36 20 63 C
68 61 72 61 63 74 C
65 72 73 29 2E 00 C ; [20]

113C 50 61 72 61 60 65 C STRparam db "Parameter",0
74 65 72 00 C ; [21]

1146 53 65 72 69 61 6C C STRserial db "Serial",0
00 C ; [22]

114D 4D 6F 64 65 6D 00 C STRmodem db "Modem",0

1153 50 72 65 73 73 20 C STRpress_msg db "Press ",27,"&B Start ",27,"&@ if information is "
1B 26 64 42 20 53 C
74 61 72 74 20 1B C
26 64 40 20 69 66 C
20 69 6E 66 6F 72 C
6D 61 74 69 6F 6E C
20 69 73 20 C
117B 63 6F 72 72 65 63 C db "correct.",0
74 2E 00 C ; [24] ; Escape sequence places " Start " in inverse video.

1184 50 72 69 6E 74 69 C STRdir_print db "Printing directory.",0
6E 67 20 64 69 72 C
65 63 74 6F 72 79 C
2E 00 C ; [26]

1198 20 20 50 6C 65 61 C STRpls_wait_msg db " Please wait.",0 ; Must have 2 leading spaces.
73 65 20 77 61 69 C
74 2E 00 C ; [27]

11A7 50 72 69 6E 74 69 C STRfile_print db "Printing file.",0
6E 67 20 66 69 6C C
65 2E 00 C ; [28]

11B6 46 69 6C 65 20 6F C STRdoesnt_exist db "File or directory does not exist.",0
72 20 64 69 72 65 C
63 74 6F 72 79 20 C
64 6F 65 73 20 6E C
6F 74 20 65 78 69 C
73 74 2E 00 C ; [30]

11D8 44 65 6C 65 74 69 C STRdir_delete db "Deleting directory.",0
6E 67 20 64 69 72 C
65 63 74 6F 72 79 C
2E 00 C ; [31]

11EC 44 69 72 65 63 74 C STRdir_del_err db "Directory is not empty, or is write protected, "
6F 72 79 20 69 73 C
20 6E 6F 74 20 65 C
6D 70 74 79 20 20 C
6F 72 20 69 73 20 C
77 72 69 74 65 20 C
70 72 6F 74 65 63 C
74 65 64 2C 20 C
121B 6F 72 20 69 73 20 C db "or is current directory.",0
63 75 72 72 65 6E C
74 20 64 69 72 65 C
63 74 6F 72 79 2E C
00 C ; [32]

1234 44 65 6C 65 74 69 C STRfile_delete db "Deleting file.",0
6E 67 20 66 69 6C C
65 2E 00 C ; [33]

1243 46 69 6C 65 20 6F C STRfile_wrt_prt db "File or directory is write protected.",0
72 20 64 69 72 65 C
63 74 6F 72 79 20 C
69 73 20 77 72 69 C
74 65 20 70 72 6F C
74 65 63 74 65 64 C
2E 00 C ; [34]

```

System Strings

```

1269 4D 61 6B 69 6E 67 C STRdir_make    db "Making directory.",0
20 64 69 72 65 63 C
74 6F 72 79 2E 00 C ; [35]

127B 44 69 72 65 63 74 C STRmkdir_err   db "Directory could not be created.",0
6F 72 79 20 63 6F C
75 6C 64 20 6E 6F C
74 20 62 65 20 63 C
72 65 61 74 65 64 C
2E 00 C ; [36]

129B 43 68 6F 6F 73 69 C STRdir_choose  db "Choosing directory.",0
6E 67 20 64 69 72 C
65 63 74 6F 72 79 C
2E 00 C ; [37]

12AF 46 6F 72 6D 61 74 C STRdisc_format db "Formatting disc.",0
74 69 6E 67 20 64 C
69 73 63 2E 00 C ; [38]

12C0 45 72 72 6F 72 3A C STRtoo_long    db "Error: line too long.",0
20 6C 69 6E 65 20 C
74 6F 6F 20 6C 6F C
6E 67 2E 00 C ; [39]

12D6 43 61 6E 6E 6F 74 C STRno_format_b db "Cannot format drive B: (read only disc).",0
20 66 6F 72 6D 61 C
74 20 64 72 69 76 C
65 20 42 3A 20 28 C
72 65 61 64 20 6F C
6E 6C 79 20 64 69 C
73 63 29 2E 00 C ; [40]

12FF 43 6F 70 79 69 6E C STRfile_copy   db "Copying specified file(s).",0
67 20 73 70 65 63 C
69 66 69 65 64 20 C
66 69 6C 65 28 73 C
29 2E 00 C ; [41]

131A 50 72 65 73 73 20 C STRfm_press_any db "Press any key to return to P.A.M. ",0
61 6E 79 20 6B 65 C
79 20 74 6F 20 72 C
65 74 75 72 6E 20 C
74 6F 20 50 2E 41 C
2E 4D 2E 20 00 C ; [42] CAUTION!

133D 4E 6F 20 64 65 73 C STRno_dest    db "No destination file specified.",0
74 69 6E 61 74 69 C
6F 6E 20 66 69 6C C
65 20 73 70 65 63 C
69 66 69 65 64 2E C
00 C ; [43]

135C 52 65 6E 61 6D 69 C STRfile_rename db "Renaming file.",0
6E 67 20 66 69 6C C
65 2E 00 C ; [44]

136B 4E 65 77 20 6E 61 C STRalready_ex db "New name already exists, is a directory, "
6D 65 20 61 6C 72 C
65 61 64 79 20 65 C
78 69 73 74 73 2C C
20 69 73 20 61 20 C
64 69 72 65 63 74 C
6F 72 79 20 20 C
6F 72 20 63 6F 75 C
6C 64 20 6E 6F 74 C
20 62 65 20 63 72 C
65 61 74 65 64 2E C
00 C ; [45]

1394 45 6E 74 65 72 20 C STRenter_wcard db "Enter new wild card and press [Return].",0
6E 65 77 20 77 69 C
6C 64 20 63 61 72 C
64 20 61 6E 64 20 C
70 72 65 73 73 20 C
58 52 65 74 75 72 C
6E 5D 2E 00 C ; [46]

```

System Strings

```

13D5 4E 6F 20 77 69 6C C STRno_wild db "No wild cards allowed in this function. Please retype "
64 20 63 61 72 64 C
73 20 61 6C 6C 6F C
77 65 64 20 69 6E C
20 74 68 69 73 20 C
66 75 6E 63 74 69 C
6F 6E 2E 20 20 50 C
6C 65 61 73 65 20 C
72 65 74 79 70 65 C
20 C
140C 66 69 6C 65 20 6E C db "file name.",0
61 6D 65 2E 00 C ; [47]
1417 4D 6F 72 65 20 66 C STRuse_wild db "More files in directory than can be displayed--use "
69 6C 65 73 20 69 C
6E 20 64 69 72 65 C
63 74 6F 72 79 20 C
74 68 61 6E 20 63 C
61 6E 20 62 65 20 C
64 69 73 70 6C 61 C
79 65 64 2D 2D 75 C
73 65 20 C
144A 77 69 6C 64 20 63 C db "wild card.",0
61 72 64 2E 00 C ; [48]
1455 44 69 73 70 6C 61 C STRdisp_dir db "Displayed dir: ",0
79 65 64 20 64 69 C
72 3A 20 00 C ; [49]
1465 50 72 65 73 73 20 C STRfm_nxt db "Press [Next]/[Prev] to see more files.",0
58 4E 65 78 74 5D C
2F 5B 50 72 65 76 C
5D 20 74 6F 20 73 C
65 65 20 6D 6F 72 C
65 20 66 69 6C 65 C
73 2E 00 C ; [50]
148C 54 68 65 20 64 69 C STRno_files db "The directory contains no files.",0
72 65 63 74 6F 72 C
79 20 63 6F 6E 74 C
61 69 6E 73 20 6E C
6F 20 66 69 6C 65 C
73 2E 00 C ; [51]
14AD 48 65 77 6C 65 74 C STRcompany db "Hewlett-Packard",0
74 2D 50 61 63 6B C
61 72 64 00 C ; [52]
14BD 20 20 20 20 20 42 C STRbattery db "      Battery",0 ; Exactly 12 spaces, right justified.
61 74 74 65 72 79 C
00 C ; [53]
14CA 20 46 69 6C 65 20 C STRfmgrmsg db " File Manager",0
4D 61 6E 61 67 65 C
72 00 C ; [54]
14D8 4D 61 69 6E 00 C STRfmmain db "Main",0 ; Max 18 chars.
14DD 53 65 6C 65 63 74 C ; [55]
STRselfunc db "Select a function.",0
20 61 20 66 75 6E C
63 74 69 6F 6E 2E C
00 C ; [56]
14F0 50 72 69 6E 74 00 C STRfmprint db "Print",0 ; Max 18 chars.
14F6 1B 4B C ; [57]
STRmvptror db 27,"K" ; Clears to EOL--do not change.
14F8 4D 6F 76 65 20 74 C db "Move the pointer to, or type the name of, the ",0
68 65 20 70 6F 69 C
6E 74 65 72 20 74 C
6F 2C 20 6F 72 20 C
74 79 70 65 20 74 C
68 65 20 6E 61 6D C
65 20 6F 66 2C 20 C
74 68 65 20 00 C ; [23] ; Max 80 chars when combined with "file or directory to print."
C ; or "file or directory to delete." or "directory to display."
C ; or "file to copy." or "destination file." or "file to rename."

```

System Strings

```

1527 66 69 6C 65 20 6F C STRtoprint db "file or directory to print.",0
    72 20 64 69 72 65 C
    63 74 6F 72 79 20 C
    74 6F 20 70 72 69 C
    6E 74 2E 00 C ; [58]

1543 50 72 69 6E 74 20 C STRprfile db "Print file: ",0
    66 69 6C 65 3A 20 C
    00 C ; [59]

1550 44 65 6C 65 74 65 C STRfmdel db "Delete",0 ; Max 18 chars.
    00 C ; [60]

1557 66 69 6C 65 20 6F C STRtodel db "file or directory to delete.",0
    72 20 64 69 72 65 C
    63 74 6F 72 79 20 C
    74 6F 20 64 65 6C C
    65 74 65 2E 00 C ; [61]

1574 44 65 6C 65 74 65 C STRdelfile db "Delete file: ",0
    20 66 69 6C 65 3A C
    20 00 C ; [62]

1582 40 61 6B 65 20 44 C STRfmmak db "Make Directory",0 ; Max 18 chars.
    69 72 65 63 74 6F C
    72 79 00 C ; [63]

1591 54 79 70 65 20 74 C STRtomake db "Type the new directory name.",0
    68 65 20 6E 65 77 C
    20 64 69 72 65 63 C
    74 6F 72 79 20 6E C
    61 6D 65 2E 00 C ; [64]

15AE 44 69 72 65 63 74 C STRmakefile db "Directory to make: ",0
    6F 72 79 20 74 6F C
    20 6D 61 6B 65 3A C
    20 00 C ; [65]

15C2 43 68 6F 6F 73 65 C STRfmchs db "Choose Directory",0 ; Max 18 chars.
    20 44 69 72 65 63 C
    74 6F 72 79 00 C ; [66]

15D3 64 69 72 65 63 74 C STRtochs db "directory to display.",0
    6F 72 79 20 74 6F C
    20 64 69 73 70 6C C
    61 79 2E 00 C ; [67]

15E9 44 69 72 65 63 74 C STRchsfle db "Directory to display: ",0
    6F 72 79 20 74 6F C
    20 64 69 73 70 6C C
    61 79 3A 20 00 C ; [68]

1600 46 6F 72 6D 61 74 C STRfmfor db "Format",0 ; Max 18 chars.
    00 C ; [69]

1607 45 6E 74 65 72 20 C STRtofor db "Enter the drive to format. (All data on the disc"
    74 68 65 20 64 72 C
    69 76 65 20 74 6F C
    20 66 6F 72 6D 61 C
    74 2E 20 20 28 41 C
    6C 6C 20 64 61 74 C
    61 20 6F 6E 20 74 C
    68 65 20 64 69 73 C
    63 20 C

1639 77 69 6C 6C 20 62 C db "will be lost.)",0
    65 20 6C 6F 73 74 C
    2E 29 00 C ; [70]

1648 44 72 69 76 65 20 C STRdrive db "Drive to format: ",0
    74 6F 20 68 6F 72 C
    6D 61 74 3A 20 00 C ; [72]

165A 56 6F 6C 75 6D 65 C STRvolume db "Volume label: ",0
    20 6C 61 62 65 6C C
    3A 20 00 C ; [73]

```

System Strings

```

1669 43 6F 70 79 00 C STRfmcp db "Copy",0 ; Max 18 chars.
C ; [74]
166E 66 69 6C 65 20 74 C STRcpyfr db "file to copy.",0
C ; [75]
167C 64 65 73 74 69 6E C STRcpyro db "destination file.",0
C ; [76]
168E 43 6F 70 79 20 66 C STRfrom db "Copy from file: ",0
C ; [77]
169F 43 6F 70 79 20 74 C STRto db "Copy to file: ",0
C ; [78]
16AE 52 65 6E 61 6D 65 C STRfmren db "Rename",0 ; Max 18 chars.
C ; [79]
16B5 66 69 6C 65 20 74 C STRtoren db "file to rename.",0
C ; [80]
16C5 18 4B 54 79 70 65 20 74 C STRname db 27,"K" ; Clears to EOL--do not change.
C ; [81]
16C7 68 65 20 6E 65 77 20 66 69 6C 65 20 6E 61 6D 65 2E 00 C ; [82]
C ; [83]
16DF 52 65 6E 61 6D 65 20 66 69 6C 65 3A C STRoldnm db "Rename file: ",0
C ; [84]
16ED 52 65 6E 61 6D 65 20 66 69 6C 65 20 74 6F 3A 20 00 C STRnewnm db "Rename file to: ",0
C ; [85]
16FE 20 53 79 73 74 65 6D 20 43 6F 6E 66 69 67 75 72 61 74 6F 6E 00 C STRsyscnf db " System Configuration",0
C ; [86]
1714 20 44 61 74 61 63 6F 6D 20 43 6F 6E 66 69 67 75 72 61 74 69 6F 6E 00 C STRcomcnf db " Datacom Configuration",0
C ; [87]
172B 20 54 69 6D 65 20 61 6E 64 20 44 61 74 65 00 C STRclkcnf db " Time and Date",0
C ; [88]
173A 20 41 6C 61 72 6D 20 00 C STRalarm db " Alarm ",0
C ; [89]
1742 4D 61 69 6E 20 4D 65 6D 6F 72 79 20 2F 20 45 64 69 73 63 00 C STRmemEdisc db "Main Memory / Edisc",0
C ; [90]
1756 45 78 74 65 72 6E 61 6C 20 44 69 73 63 20 44 72 69 76 65 73 00 C STRextdisc db "External Disc Drives",0
C ; [91]
176B 44 69 73 63 20 57 72 69 74 65 20 56 65 72 69 66 79 00 C STRdiscVer db "Disc Write Verify",0
C ; [92]
177D 50 6F 77 65 72 20 53 61 76 65 20 4D 6F 64 65 00 C STRpwrSav db "Power Save Mode",0
C ; [93]
178D 44 69 73 70 6C 61 79 20 54 69 6D 65 6F 75 74 20 28 6D 69 6E 29 00 C STRdisptime db "Display Timeout (min)",0
C ; [94]

```

\*\*\*\*\*  
Messages  
[88] - [102]  
26 chars max.

\*\*\*\*\*  
These are  
[System Config]  
Prompts.  
\*\*\*\*\*

### System Strings

```

17A3 43 75 72 73 6F 72      C STRcursor    db "Cursor Type",0
      20 54 79 70 65 00      C ; [93]
17AF 43 6F 6E 73 6F 6C      C STRconsole   db "Console Mode",0
      65 20 4D 6F 64 65      C ; [94]
178C 54 6F 6E 65 20 44      C STRbeepcnf  db "Tone Duration",0
      75 72 61 74 69 6F      C ; [95]
17CA 50 6C 6F 74 74 65      C STRPLTdev   db "Plotter Interface",0
      72 20 49 6E 74 65      C ; [96]
17DC 50 72 69 6E 74 65      C STRprinter  db "Printer Mode",0
      72 20 4D 6F 64 65      C ; [97]
17E9 50 72 69 6E 74 65      C STRPRNdev   db "Printer Interface",0
      72 20 49 6E 74 65      C ; [98]
17FB 50 72 69 6E 74 65      C STRpitch    db "Printer Pitch",0
      72 20 50 69 74 63      C ; [99]
1809 50 72 69 6E 74 65     C STRlinespc  db "Printer Line Spacing",0
      72 20 4C 69 6E 65
      20 53 70 61 63 69
      6E 67 00      C ; [100]
181E 50 72 69 6E 74 65     C STRskipperf db "Printer Skip Perforation",0
      72 20 53 6B 69 70
      20 50 65 72 66 6F
      72 61 74 69 6F 6E
      00      C ; [101]
1837 44 61 74 61 63 6F      C STRAuxCnf   db "Datacom Interface",0
      6D 20 49 6E 74 65
      72 66 61 63 65 00      C ; [102]
1849 54 72 61 6E 73 6D      C STRbaudRate db "Transmission Rate (BPS)",0 ; ****
      69 73 73 69 6F 6E      C ; Messages
      20 52 61 74 65 20      C ; [103] - [111]
      28 42 50 53 29 00      C ; 25 chars max.
1861 57 6F 72 64 20 4C      C STRwordlen  db "Word Length (bits)",0 ; ****
      65 6E 67 74 68 20
      28 62 69 74 73 29      C ; These are
      00      C ; [104]       [Datacom Config]
                                C ; prompts.
                                C ; ****
1874 53 74 6F 70 20 42      C STRstopbits db "Stop Bits",0
      69 74 73 00      C ; [105]
187E 50 61 72 69 74 79      C STParity    db "Parity",0
      00      C ; [106]
1885 58 4F 4E 2F 58 4F      C STRxonxoff db "XON/XOFF Pacing",0
      46 46 20 50 61 63
      69 6E 67 00      C ; [107]
1895 43 54 53 20 4C 69      C STRctssline db "CTS Line",0 ; CCITT V.24 106
      6E 65 00      C ; [108]
189E 44 53 52 20 4C 69      C STRdsrline  db "DSR Line",0 ; CCITT V.24 107
      6E 65 00      C ; [109]
18A7 44 43 44 20 4C 69      C STRdcldline db "DCD Line",0 ; CCITT V.24 109
      6E 65 00      C ; [110]
18B0 50 6F 77 65 72 20      C STRpower    db "Power to Interface",0
      74 6F 20 49 6E 74
      65 72 66 61 63 65
      00      C ; [111]

```

**System Strings**

```

18C3 4F 6E 20 00      C STRon      db "On ",0           ; [112], [113] *****  

18C7 4F 66 66 00      C STRoff     db "Off",0          ; must be the same length Messages  

18CB 55 6E 64 65 72 73 C STRundersc db "Underscore",0 ; [112] - [125]  

       63 6F 72 65 00    C : [113] 34 chars max.  

18D6 42 6F 78 00      C STRboxc    db "Box",0          *****  

18DA 4C 6F 6E 67 20 00 C STRlongbeep db "Long ",0      ; [114] These are [System Config]  

18E0 53 68 6F 72 74 00 C STRshortbeep db "Short",0   ; [115] responses.  

18E6 41 6C 70 68 61 20 C STRgraphalph db "Alpha and HP Graphics",0 ; [116] *****  

       61 6E 64 20 48 50  
       20 47 72 61 70 68  
       69 63 73 00      C : [117]  

18FC 48 50 20 47 72 61 C STRgraph    db "HP Graphics Only",0 ; [118]  

       70 68 69 63 73 20  
       4F 6E 6C 79 00    C ; [119]  

190D 41 6C 70 68 61 20 C STRalph    db "Alpha Only",0 ; [120]  

       4F 6E 6C 79 00    C ; [121]  

1918 4E 6F 20 43 6F 6E C STRnoconf  db "No Configuration",0 ; [122]  

       66 69 67 75 72 61  
       74 69 6F 6E 00    C ; [123]  

1929 4E 6F 72 6D 61 6C C STRnorm    db "Normal",0 ; [124]  

       00                 C ; [125]  

1930 45 78 70 61 6E 64 C STRexpan   db "Expanded",0 ; [126]  

       65 64 00             C ; [127]  

1939 43 6F 6D 70 72 65 C STRcompr   db "Compressed",0 ; [128]  

       73 73 65 64 00    C ; [129]  

1944 45 78 70 61 6E 64 C STRexpcompr db "Expanded-Compressed",0 ; [130]  

       65 64 2D 43 6F 6D  
       70 72 65 73 73 65  
       64 00             C ; [131]  

1958 45 76 65 6E 00    C STReven    db "Even",0          *****  

195D 4F 64 64 20 00    C STRodd     db "Odd ",0          ; [126] - [130] 15 chars max. Messages  

1962 4E 6F 6E 65 00    C STRnone    db "None",0          ; [127] These are [Datacom Config] responses.  

1967 49 67 6E 6F 72 65 C STRignore  db "Ignore ",0        ; [128] - [129] must be same length.  

       20 00             C ; [129] and [130] must be same length.  

196F 4F 62 73 65 72 76 C STRObserve db "Observe",0 ; [130] *****  

       65 00             C ; [131]  

1977 48 50 20 38 32 31 C STR82164  db "HP 82164A",0 ; [132] *****  

       36 34 41 00         C ; [133]  

1981 38 20 6C 69 6E 65 C STR6lines  db "6 lines per inch",0 ; [134] *****  

       73 20 70 65 72 20  
       69 6E 63 68 00    C ; [135] 2.36 lines/cm  

1992 38 20 6C 69 6E 65 C STR8lines  db "8 lines per inch",0 ; [136] *****  

       73 20 70 65 72 20  
       69 6E 63 68 00    C ; [137] 3.15 lines/cm  

19A3 48 50 2D 49 4C 00 C STRhpil   db "HP-IL",0          *****  

19A9 48 50 00             C STRHP     db "HP",0          ; [138] These are [System Config]  

19AC 41 6C 74 65 72 6E C STRalt    db "Alternate",0    ; [139] prompts.  

       61 74 65 00         C ; [140]  

19B6 54 69 6D 65 20 5A C STRtimezone db "Time Zone",0 ; [141] *****  

       6F 6E 65 00         C ; [142] 22 chars max. These are [Time & Date] prompts.  

19C0 48 6F 75 72 00    C STRhour   db "Hour",0          ; [143]  

19CS 4D 69 6E 75 74 65 C STRminutes db "Minutes",0 ; [144]  

       73 00             C ; [145]

```

System Strings

```

19CD 53 65 63 6F 6E 64 C STRseconds db "Seconds",0
    73 00 ; [140]
19D5 4D 6F 6E 74 68 00 C STRmonth db "Month",0
    ; [141]
19DB 44 61 79 00 C STRday db "Day",0
    ; [142]
19DF 59 65 61 72 00 C STRyear db "Year",0
    ; [143]

C :***** PAM Softkeys (exactly 16 characters) *****
C ;*****
C *****

19E4 20 20 20 20 20 20 C SFKblank db "      "
    20 20 ; [146]           db "      ",0 ; Leave blank
19EC 20 20 20 20 20 20 C SFKhelp db "      "
    20 20 ; [147]           db " Help ",0
19F5 20 20 20 20 20 20 C SFKchdir db " Choose "
    20 20
19FD 20 20 48 65 6C 70 C SFKstover db " Start "
    20 20 00 ; [148]         db " Over ",0
1A06 20 43 68 6F 6F 73 C SFKexit db "      "
    65 20
1A0E 20 20 44 69 72 20 C SFKstart db "      "
    20 20 00 ; [149]         db " Start ",0
1A17 20 53 74 61 72 74 C SFKmainkeys db " Start "
    20 20
1A1F 20 20 4F 76 65 72 C SFKmainf2 db " File "
    20 20 00 ; [161]         db " Manager ",0
1A28 20 20 20 20 20 20 C SFKmainf3 db " Time & "
    20 20
1A30 20 20 45 78 69 74 C SFKmainf4 db " Date ",0
    20 20 00 ; [152]         db " Reread "
1A39 20 20 20 20 20 20 C SFKmainf5 db " Discs ",0
    20 20
1A41 20 53 74 61 72 74 C SFKmainf6 db " Datacom"
    20 20 00 ; [153]         db " Config ",0
1A44 20 53 74 61 72 74 C SFKmainf7 db " System "
    20 20
1A52 20 41 70 70 6C 69 C SFKmainf8 db " Config ",0
    63 20 00 ; [154]         db " Off ",0
1A58 20 20 46 69 6C 65 C
    20 20
1A63 20 4D 61 6E 61 67 C
    65 72 00 ; [155]         db " Data "
1A6C 20 54 69 6D 65 20 C
    26 20
1A74 20 20 44 61 74 65 C
    20 20 00 ; [156]         db " Manager "
1A7D 20 52 65 72 65 61 C
    64 20
1A85 20 44 69 73 63 73 C
    20 20 00 ; [157]         db " Data "
1A8E 20 44 61 74 61 63 C
    6F 6D
1A96 20 43 6F 6E 66 69 C
    67 20 00 ; [158]         db " Manager "
1A9F 20 53 79 73 74 65 C
    6D 20
1AA7 20 43 6F 6E 66 69 C
    67 20 00 ; [159]         db " Data "
1AB0 20 20 20 20 20 20 C
    20 20
1AB8 20 20 4F 66 66 20 C
    20 20 00 ; [160]         db " Manager "

```

**System Strings**

<b>1AC1</b>	20 20 4E 65 78 74	C	SFKcnff3	db " Next "	
	20 20	C		db " Choice ",0	
<b>1AC9</b>	20 43 68 6F 69 63	C	; [157]		
	65 20 00	C			
<b>1AD2</b>	50 72 65 76 69 6F	C	SFKcnff4	db " Previous"	
	75 73	C		db " Choice ",0	
<b>1ADA</b>	20 43 68 6F 69 63	C	; [158]		
	65 20 00	C			
<b>1AE3</b>	44 65 66 61 75 6C	C	SFKcnff5	db " Default "	
	74 20	C		db " Values ",0	
<b>1AEB</b>	20 56 61 6C 75 65	C	; [159]		
	73 20 00	C			
<b>1AF4</b>	20 53 74 6F 70 20	C	SFKstprn	db " Stop "	
	20 20	C		db " Print ",0	
<b>1AFC</b>	20 50 72 69 6E 74	C	; [162]		
	20 20 00	C			
<b>1B05</b>	20 43 6F 70 79 20	C	SFKdeff1	db " Copy 1 "	***** Messages
	31 20	C		db " char ",0	[163] - [170] are MSDOS
<b>1B0D</b>	20 20 63 68 61 72	C	; [163]		softkey labels.
	20 20 00	C			*****
<b>1B16</b>	43 6F 70 79 20 75	C	SFKdeff2	db "Copy up "	
	70 20	C		db " to char",0	
<b>1B1E</b>	20 74 6F 20 63 68	C	; [164]		
	61 72 00	C			
<b>1B27</b>	20 20 43 6F 70 79	C	SFKdeff3	db " Copy "	
	20 20	C		db " all ",0	
<b>1B2F</b>	20 20 61 6C 6C 20	C	; [165]		
	20 20 00	C			
<b>1B38</b>	20 53 6B 69 70 20	C	SFKdeff4	db " Skip 1 "	
	31 20	C		db " char ",0	
<b>1B40</b>	20 20 63 68 61 72	C	; [166]		
	20 20 00	C			
<b>1B49</b>	53 6B 69 70 20 75	C	SFKdeff5	db "Skip up "	
	70 20	C		db " to char",0	
<b>1B51</b>	20 74 6F 20 63 68	C	; [167]		
	61 72 00	C			
<b>1B5A</b>	20 56 6F 69 64 20	C	SFKdeff6	db " Void "	
	20 20	C		db " input ",0	
<b>1B62</b>	20 69 6E 70 75 74	C	; [168]		
	20 20 00	C			
<b>1B6B</b>	20 54 6F 67 67 6C	C	SFKdeff7	db " Toggle "	
	65 20	C		db " insert ",0	
<b>1B73</b>	20 69 6E 73 65 72	C	; [169]		
	74 20 00	C			
<b>1B7C</b>	20 20 4E 65 77 20	C	SFKdeff8	db " New "	
	20 20	C		db " line ",0	
<b>1B84</b>	20 20 6C 69 6E 65	C	; [170]		
	20 20 00	C			
<b>1B8D</b>	20 50 72 69 6E 74	C	SFKfmgf1	db " Print "	
	20 20	C		db "File/Dir",0	
<b>1B95</b>	46 69 6C 65 2F 44	C	; [171]		
	69 72 00	C			
<b>1B9E</b>	20 44 65 6C 65 74	C	SFKfmgf2	db " Delete "	
	65 20	C		db "File/Dir",0	
<b>1BA6</b>	46 69 6C 65 2F 44	C	; [172]		
	69 72 00	C			
<b>1BAF</b>	20 20 4D 61 6B 65	C	SFKfmgf3	db " Make "	
	20 20	C		db " Dir ",0	
<b>1BB7</b>	20 20 44 69 72 20	C	; [173]		
	20 20 00	C			
<b>1BC0</b>	20 20 20 20 20 20	C	SFKfmgf5	db " "	
	20 20	C		db " Format ",0	
<b>1BC8</b>	20 46 6F 72 6D 61	C	; [174]		
	74 20 00	C			

System Strings

```

1BD1 20 20 43 6F 70 79 C SFKfmgf6 db " Copy "
20 20 C
1BD9 20 20 46 69 6C 65 C SFKfmgf7 db " File ",0
20 20 00 C ; [175]
1BE2 20 52 65 6E 61 6D C SFKfmgf7 db " Rename "
65 20 C
1BEA 20 20 46 69 6C 65 C SFKfmgf7 db " File ",0
20 20 00 C ; [176]
1BF3 20 20 53 65 74 20 C SFKchdf3 db " Set "
20 20 C
1BFB 57 69 6C 64 63 61 C SFKchdf3 db " Wildcard",0
72 64 00 C ; [177]
1C04 20 44 69 73 63 20 C STRwrite_fail db " Disc write failure.",0 ; TERM message
77 72 69 74 65 20 C
66 61 69 6C 75 72 C
65 2E 00 C ; [178]
1C19 20 20 20 4E 6F 20 C STRcancel db " No "
20 20 C
1C21 20 43 68 61 6E 67 C STRcancel db " Change ",0
65 20 00 C ; [180]
1C2A 53 65 74 74 69 6E C STRcur_set db "Setting",0 ; Config screens heading
67 00 C ; [179]
1C32 44 4F 53 20 43 6F C STRdos_com db "DOS Commands",0 ; Max 14 chars
6D 6D 61 6E 64 73 C
00 C ; [181]
C ;*****TERM Messages***** ; (STRsoftkeys [182] is in file STRTERM.CRM)
C ;*****TERM Messages***** ; [182]

1C3F 1B 58 73 1B 26 61 C STRhost db ESC,"[s",ESC,"&axY" ; Do not change.
78 59 C
1C47 20 20 46 52 4F 4D C db " FROM HOST to file: " ; Exactly 21 spaces.
20 48 4F 53 54 20 C
74 6F 20 66 69 6C C
65 3A 20 C
1C5C 1B 26 61 58 0A C db ESC,"&ax",LF ; Do not change.
1C61 20 20 54 4F 20 48 C db " to HOST from file: " ; Exactly 21spaces.
4F 53 54 20 66 72 C
6F 6D 20 66 69 6C C
65 3A 20 C
1C76 00 C db 0
; [183]
1C77 74 65 72 60 73 65 C STRsend db "term send",0 ; Default filename for TO HOST from file:
6E 64 00 C ; [184]
1C80 74 65 72 6D 6C 6F C STRlog db "term log",0 ; Default filename for FROM HOST to file:
67 00 C ; [185]
1C88 20 43 61 6E 6E 6F C STRaux db " Cannot open AUX.",CR,LF,0
74 20 6F 70 65 6E C
20 41 55 58 2E 0D C
0A 00 C ; [186]
1C9C 20 49 6E 73 75 66 C STRram db " Insufficient RAM.",CR,LF,0
66 69 63 69 65 6E C
74 20 52 41 4D 2E C
0D 0A 00 C ; [187]
1CB1 20 54 72 61 6E 73 C STRtransmit db " Transmit failure.",0
6D 69 74 20 66 61 C
69 6C 75 72 65 2E C
00 C ; [188]
1CC4 20 43 61 6E 6E 6F C STRaccess db ' Cannot access "",0
74 20 61 63 63 65 C
73 73 20 22 00 C ; [189]

```

.list

CRM ENDS  
END



# D

# Character Sets

---

This chapter lists the character sets provided by the computer.

## HP Mode:

- Roman8 set.
- Line-drawing/math sets.

## Alternate Mode:

- Alternate set.

The Roman8 and alternate character sets are present in normal (bold) form and in light ("halfbright") form. (Refer to "Fast Video Interrupt" in chapter 5.)

**Table D-1. Roman8 Character Set**

DECIMAL	0	16	32	48	64	80	96	112	128	144	160	176	192	208	224	240
HEX	0	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
0	0	8	9	0	@	P	'	p	8	9	-	a	A	A	P	
1	1	5	6	!	1	A	Q	q	5	6	A	8	E	f	A	
2	2	5	5	"	2	B	R	b	r	5	5	8	8	8	8	5
3	3	E	x	#	3	C	S	c	s	E	+	E	0	E	D	
4	4	E	9	\$	4	D	T	d	t	E	+	E	C	a	d	
5	5	E	5	%	5	E	U	e	u	E	+	E	s	e	f	
6	6	2	5	&	6	F	V	f	v	2	5	N	6	e	t	
7	7	E	E	'	7	G	W	g	w	E	L	I	6	4		
8	8	E	E	(	8	H	X	h	x	E	L	I	6	0	5	
9	9	4	E	)	9	I	Y	i	y	4	L	I	6	1	0	
10	A	7	6	*	:	J	Z	j	z	7	T	8	0	8	0	
11	B	7	E	+	;	K	L	k	l	7	T	8	£	0	5	
12	C	F	F	,	<	L	\	l	l	F	-	~	ä	€	5	
13	D	F	F	-	=	M	J	m	3	F	I	0	ö	0	»	
14	E	F	F	.	>	N	^	n	~	F	+	O	ö	8	+	
15	F	F	F	/	?	O	—	o	~	F	■	E	ö	8	9	

## Example:

$$M = 64 + 13 = 77$$

$$= 40h + Dh = 4Dh$$

**Table D-2. Line-Drawing/Math Character Sets**

DECIMAL	0	16	32	48	64	80	96	112	128	144	160	176	192	208	224	240
HEX	0	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
0 0				+	F	J	F	J			o	T	π	π	π	π
1 1			T	T	L	r	T	r			√	1	α	γ	α	γ
2 2			T	T	+	r	T	r			I	2	3	0	3	0
3 3			T	T	█	█	T	█			δ	3	ψ	σ	ψ	σ
4 4			T	T	█	█	T	█			π	4	♦	τ	♦	τ
5 5				T	I	I	I	I			±	5	ε	∞	ε	∞
6 6				T	L	+	L	+			Δ	6	Θ	Δ	6	Δ
7 7			≡	T	L	■	L	■			Γ	7	λ	8	λ	8
8 8			≡	T	L	█	L	█			÷	8	η	χ	η	χ
9 9				=	7	7	7	7			≈	9	ι	υ	ι	υ
10 A			+	I	T	-	T	-			Π	Ω	θ	ξ	θ	ξ
11 B			+	-	□	F	□	F			Γ	Λ	κ	†	κ	†
12 C			-	#	m	--	m	--			Ψ	∞	w	+	w	+
13 D			L	Γ	+	π	+	π			≡	J	μ	T	μ	T
14 E			-	#	+	-	+	-			Φ	†	v	+	v	+
15 F			+	#	≡	π	≡	π			Σ	Σ	P	+	P	%

**Example:**

$$\Gamma = 64 + 13 = 77$$

$$= 40h + Dh = 4Dh$$

Table D-3. Alternate Character Set

DECIMAL		0	16	32	48	64	80	96	112	128	144	160	176	192	208	224	240
	HEX	0	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
0	0	█			█	█	█	█	█	█	█	█	█	█	█	█	█
1	1	▀	▀	▀	!	▀	▀	▀	▀	▀	▀	▀	▀	▀	▀	▀	▀
2	2	▀	▀	▀	"	▀	▀	▀	▀	▀	▀	▀	▀	▀	▀	▀	▀
3	3	▀	▀	▀	#	▀	▀	▀	▀	▀	▀	▀	▀	▀	▀	▀	▀
4	4	▀	▀	▀	\$	▀	▀	▀	▀	▀	▀	▀	▀	▀	▀	▀	▀
5	5	▀	▀	▀	%	▀	▀	▀	▀	▀	▀	▀	▀	▀	▀	▀	▀
6	6	▀	▀	▀	&	▀	▀	▀	▀	▀	▀	▀	▀	▀	▀	▀	▀
7	7	▀	▀	▀	'	▀	▀	▀	▀	▀	▀	▀	▀	▀	▀	▀	▀
8	8	▀	▀	▀	(	▀	▀	▀	▀	▀	▀	▀	▀	▀	▀	▀	▀
9	9	▀	▀	▀	)	▀	▀	▀	▀	▀	▀	▀	▀	▀	▀	▀	▀
10	A	▀	▀	▀	*	▀	▀	▀	▀	▀	▀	▀	▀	▀	▀	▀	▀
11	B	▀	▀	▀	+	▀	▀	▀	▀	▀	▀	▀	▀	▀	▀	▀	▀
12	C	▀	▀	▀	,	▀	▀	▀	▀	▀	▀	▀	▀	▀	▀	▀	▀
13	D	▀	▀	▀	-	=	▀	▀	▀	▀	▀	▀	▀	▀	▀	▀	▀
14	E	▀	▀	▀	.	▀	▀	▀	▀	▀	▀	▀	▀	▀	▀	▀	▀
15	F	▀	▀	▀	?	▀	▀	▀	▀	▀	▀	▀	▀	▀	▀	▀	▀

Example:

$$M = 64 + 13 = 77$$

$$= 40h + Dh = 4Dh$$

**Table D-4. Character Sets - Numeric Listing**

HP Char	Line Char	Alt Char	Code		HP Char	Line Char	Alt Char	Code	
			Dec	Hex				Dec	Hex
8			0 00		!	!	!	32	20
9			1 01		"	"	"	33	21
;			2 02		#	#	#	34	22
*			3 03		\$	\$	\$	35	23
*			4 04		%	%	%	36	24
*			5 05		&	&	&	37	25
*			6 06		-	-	-	38	26
.			7 07		(	)	)	39	27
■			8 08		=	=	=	40	28
○			9 09		*	*	*	41	29
□			10 0A		+	+	+	42	2A
□			11 0B		,	,	,	43	2B
□			12 0C		-	-	-	44	2C
□			13 0D		_	_	_	45	2D
□			14 0E		.	.	.	46	2E
□			15 0F		/	/	/	47	2F
□			16 10		0	0	0	48	30
□			17 11		1	1	1	49	31
□			18 12		2	2	2	50	32
□			19 13		3	3	3	51	33
□			20 14		4	4	4	52	34
□			21 15		5	5	5	53	35
□			22 16		6	6	6	54	36
□			23 17		7	7	7	55	37
□			24 18		8	8	8	56	38
□			25 19		9	9	9	57	39
□			26 1A		:	:	:	58	3A
□			27 1B		;	;	;	59	3B
□			28 1C		<	<	<	60	3C
□			29 1D		=	=	=	61	3D
□			30 1E		?	?	?	62	3E
□			31 1F		#	#	#	63	3F

Table D-4. Character Sets - Numeric Listing (Continued)

HP Char	Line Char	Alt Char	Code		HP Char	Line Char	Alt Char	Code	
			Dec	Hex				Dec	Hex
@	£	@	64	40	`	£	`	96	60
A	£	A	65	41	a	£	a	97	61
B	+	B	66	42	b	+	b	98	62
C	■	C	67	43	c	■	c	99	63
D	■	D	68	44	d	■	d	100	64
E	—	E	69	45	e	—	e	101	65
F	—	F	70	46	f	—	f	102	66
G	—	G	71	47	g	—	g	103	67
H	Г	H	72	48	h	Г	h	104	68
I	І	I	73	49	i	І	i	105	69
J	І	J	74	4A	j	І	j	106	6A
K	Ї	K	75	4B	k	Ї	k	107	6B
L	Ї	L	76	4C	l	Ї	l	108	6C
M	+	M	77	4D	m	+	m	109	6D
N	+	N	78	4E	n	+	n	110	6E
O	—	O	79	4F	o	—	o	111	6F
P	—	P	80	50	p	—	p	112	70
Q	—	Q	81	51	q	—	q	113	71
R	—	R	82	52	r	—	r	114	72
S	—	S	83	53	s	—	s	115	73
T	—	T	84	54	t	—	t	116	74
U	—	U	85	55	u	—	u	117	75
V	+	V	86	56	v	+	v	118	76
W	—	W	87	57	w	—	w	119	77
X	■	X	88	58	x	■	x	120	78
Y	—	Y	89	59	y	—	y	121	79
Z	—	Z	90	5A	z	—	z	122	7A
£	—	£	91	5B	€	—	€	123	7B
/	—	/	92	5C	—	—	—	124	7C
]	—	]	93	5D	]	—	]	125	7D
^	-	^	94	5E	~	-	~	126	7E
-	π	-	95	5F	¤	¤	¤	127	7F

**Table D-4. Character Sets – Numeric Listing (Continued)**

HP Char	Math Char	Alt Char	Code		HP Char	Math Char	Alt Char	Code	
			Dec	Hex				Dec	Hex
¢	¢	¢	128	80			á	160	A0
¤	¤	¤	129	81			í	161	A1
₩	₩	₩	130	82			ó	162	A2
₹	₹	₹	131	83			ú	163	A3
₪	₪	₪	132	84			ñ	164	A4
₭	₭	₭	133	85			⌐	165	A5
₱	₱	₱	134	86			⌐	166	A6
₵	₵	₵	135	87			⌐	167	A7
₵	₵	₵	136	88			⌐	168	A8
₵	₵	₵	137	89			⌐	169	A9
₵	₵	₵	138	8A			⌐	170	AA
₵	₵	₵	139	8B			⌐	171	AB
₵	₵	₵	140	8C			⌐	172	AC
₵	₵	₵	141	8D			⌐	173	AD
₵	₵	₵	142	8E			⌐	174	AE
₵	₵	₵	143	8F			⌐	175	AF
₵	₵	₵	144	90			⌐	176	B0
₵	₵	₵	145	91			⌐	177	B1
₵	₵	₵	146	92			⌐	178	B2
₵	₵	₵	147	93			⌐	179	B3
₵	₵	₵	148	94			⌐	180	B4
₵	₵	₵	149	95			⌐	181	B5
₵	₵	₵	150	96			⌐	182	B6
₵	₵	₵	151	97			⌐	183	B7
₵	₵	₵	152	98			⌐	184	B8
₵	₵	₵	153	99			⌐	185	B9
₵	₵	₵	154	9A			⌐	186	BA
₵	₵	₵	155	9B			⌐	187	BB
₵	₵	₵	156	9C			⌐	188	BC
₵	₵	₵	157	9D			⌐	189	BD
₵	₵	₵	158	9E			⌐	190	BE
₵	₵	₵	159	9F			⌐	191	BF

Table D-4. Character Sets – Numeric Listing (Continued)

HP Char	Math Char	Alt Char	Code		HP Char	Math Char	Alt Char	Code	
			Dec	Hex				Dec	Hex
à	¶	¤	192	C0	À	¶	¤	224	E0
è	¤	¤	193	C1	È	¤	¤	225	E1
ò	¤	¤	194	C2	Ò	¤	¤	226	E2
ó	¤	¤	195	C3	Ó	¤	¤	227	E3
á	+	-	196	C4	Á	+	+	228	E4
é	€	+	197	C5	É	€	€	229	E5
ö	¤	¤	198	C6	Ö	¤	¤	230	E6
ú	¤	¤	199	C7	Ú	¤	¤	231	E7
à	ñ	ñ	200	C8	À	ñ	ñ	232	E8
è	ñ	ñ	201	C9	È	ñ	ñ	233	E9
ò	ø	ø	202	CA	Ò	ø	ø	234	EA
ó	ø	ø	203	CB	Ó	ø	ø	235	EB
ä	w	w	204	CC	Ä	w	w	236	EC
ë	þ	þ	205	CD	Ë	þ	þ	237	ED
ö	v	v	206	CE	Ö	v	v	238	EE
ü	p	p	207	CF	Ü	p	p	239	EF
A	π	π	208	D0	À	π	π	240	F0
à	y	¬	209	D1	È	y	+	241	F1
ò	ø	ø	210	D2	Ò	ø	ø	242	F2
é	ç	ç	211	D3	Ó	ç	ç	243	F3
í	‑	‑	212	D4	Í	‑	‑	244	F4
í	‑	‑	213	D5	Í	‑	‑	245	F5
œ	Δ	Π	214	D6	Œ	Δ	Δ	246	F6
œ	ð	#	215	D7	Œ	ð	ð	247	F7
œ	x	#	216	D8	Œ	x	x	248	F8
œ	v	#	217	D9	Œ	v	v	249	F9
œ	‑	r	218	DA	Œ	‑	‑	250	FA
œ	†	■	219	DB	Œ	†	†	251	FB
œ	+	■	220	DC	Œ	■	■	252	FC
œ	+	■	221	DD	Œ	†	†	253	FD
œ	+	■	222	DE	Œ	†	†	254	FE
œ	+	■	223	DF	Œ	†	†	255	FF

# Using TERM in Batch Files

You can somewhat automate file uploading by using TERM in an MS-DOS batch file. If you specify a filename on the command line when you invoke TERM, the first thing TERM will do when it comes up is attempt to open that file and transmit it to the host. The effect is exactly the same as if you had run TERM, pressed the "File Names" function key, typed a filename into the "TO HOST" field, pressed "File Names" again to save the filename definition, and then pressed "To Host" to start the transfer.

**term filename**

The various upload protocols that are normally accessed by using the **(Shift)** and **(CTRL)** keys together with "To Host" can be specified by prefixing the command line filename with a special protocol selection character:

- = "Equals" indicates a *Wait For Echo* upload.  
After each character is transmitted, TERM waits for the host to echo back the exact same character before proceeding with the next character. This is the same as pressing **(Shift)** along with the "To Host" key.
- + "Plus" indicates an *XON-triggered* upload.  
Whenever TERM sends out a carriage return, it stops transmitting data until it sees an XON (DC1) from the host. This is the standard handshake used by an HP3000. An equivalent keystroke would be to press **(CTRL)** together with the "To Host" key.
- , "Comma" indicates that after the upload completes, TERM should exit and leave the datacom device power turned on. This is the same as pressing **(Shift)** along with the "Exit" key.

- ; "Semicolon" indicates that after the upload completes, TERM should exit and turn off the datacom device power.  
This is the same as simply pressing the "Exit" key.

You can combine these special characters in any order to specify the upload action you desire; the only restriction is that they appear immediately before the name of the file to be uploaded (without any separating spaces).

**term [+ = , ; ]filename**

Note that these options only affect the way by which a file is sent to the host--TERM does not provide any means of checking incoming data. Consequently, you cannot configure TERM to wait, as an example, for the host to send a particular string of characters, or to recognize errors in the incoming data and request the host to retransmit. Along the same lines, TERM assumes that what it sends out is always sent correctly; there is no facility for retransmitting data if the host detects an error.



If the file you specify as a TERM command line parameter contains, as the first thing, commands to the built-in modem, it may be necessary to begin the file with a blank line. For example, if you wanted to autodial Information by typing TERM INFO, the file should contain two lines:

```
<blank line>
ATDT1,5551212
```

(The blank line may or may not be necessary, depending on the version of modem firmware and system AUX driver.)

---

**Example:** Upload an assembler listing to the HP3000 using EDIT/3000.

```
echo off
rem ****
rem BATCH FILE TO UPLOAD TEXT FILES TO AN HP3000
rem
rem      to3000 localfile remotefile [localfile remotefile ...]
rem
rem You should already be logged onto the 3000 before running this
rem batch file; each local file uploaded can be a maximum of 10000
rem lines in length, with each line not to exceed 132 characters.
rem This batch file requires a small amount of Edisc to work,
rem as it sends commands to the 3000 editor via a file on drive A.
rem Also, the file being uploaded should NOT contain a "//", since
rem this will cause the 3000's editor to stop accepting text.
rem ****
:BEGIN
if x%1==x goto help
if x%2==x goto badparm2
if not exist %1 goto badparm1
echo >a:upload.ctl editor
echo >>a:upload.ctl set length=132
echo >>a:upload.ctl set right=132
echo >>a:upload.ctl set size=10000
echo >>a:upload.ctl addq
term +,a:upload.ctl
term +,%1
echo >a:upload.ctl //
echo >>a:upload.ctl :purge %2
echo >>a:upload.ctl keepq %2,unn
echo >>a:upload.ctl exit
term +,a:upload.ctl
del a:upload.ctl
goto nextparm
:BADPARM2
echo No remote file specified to receive "%1"
:HELP
echo usage: to3000 localfile remotefile [localfile remotefile ...]
goto end
:BADPARM1
echo "%1" does not exist
:NEXTPARM
```

```
shift  
shift  
if not x%1==x goto begin  
:END
```

# F Mass Storage

---

## F.1 Disc Drive Options

There are several options for adding disc drives to the Portable PLUS. Four alternatives are discussed below. The assigning of drive letters (such as A; B;) may seem confusing at first, but it follows a very specific scheme.

The built-in disc drives are always A: and B:. The external disc drives specified in the PAM System Configuration Menu use the next letters. For example, if there are two discs selected they will be C: and D:, even if there are no physical disc drives attached. The disc drives accessed by these letters are the HP-IL disc drives and the newer HP-IB disc drives. If a device driver is downloaded using a CONFIG.SYS file to talk to other disc drives (such as the Amigo driver discussed later), it will take the next available letters.

### F.1.1 Built-In Disc Drives

The Portable PLUS has a built-in disc drive consisting of two separate drives. The first is the Edisc (drive A:, also referred to as the RAM disc or Electronic disc), which is similar to any removable disc except that it works at lightning speed and is silent. There is also a ROM disc (drive B:), which contains the system files used in the Portable PLUS. This disc is similar to any removable disc that is write-protected. You can read from the ROM disc but you can not write to it.

### F.1.2 HP 9114A HP-IL Disc Drive

The HP 9114A is a portable, battery-powered disc drive that packs over 700,000 characters on one double-sided 3.5-inch disc. Simply connect it with HP-IL cables and set the PAM System Configuration Menu entry for External Discs to one. This disc drive becomes drive C:.

### F.1.3 HP-IB Disc Drives

The HP-IB disc drives may be used with the Portable PLUS with an HP 82169A HP-IL/HP-IB Interface for hardware compatibility.



**Note**

If you put everything together correctly and you get a "bad unit" error message, you probably have an older HP-IL/HP-IB converter. It will need to be updated with a new processor. Converters with serial numbers before 2401A00000 have the old processors. Contact Hewlett-Packard regarding the update.

In the United States:

Hewlett-Packard Company  
Corvallis Service Center  
P.O. Box 999  
Corvallis, Oregon 97339  
(503) 757-2002

In other countries:

Contact a local HP office  
for information.

The device driver for the newer HP-IB disc drives is built into the BIOS. The newer HP-IB disc drives include the HP 9122, the HP 9125, the HP 9133/4D, the HP 9133/4H, the HP 9133/4L, and the HP 9153/4. These disc drives are connected as follows:

1. Connect the disc drives via HP-IB to the HP-IL/HP-IB converter and connect the converter at any position in the HP-IL loop.
2. Set all the switches on the converter to zero, then connect the converter to ac power.
3. Set the number of external disc drives in the PAM System Configuration Menu to the appropriate number (depending on the configuration) to include the newer HP-IB disc drives.

The older HP-IB disc drives implement a different command set (called "Amigo" protocol) than the newer disc drives. An Amigo device driver is included with the Portable PLUS which will support the following Amigo disc drives: the HP 9121D/S, the HP 82901M/2M, all versions of Winchester drive (5MB 1 or 4 volume, 10MB, and 15MB), and the HP 9895A 8-inch floppy. To connect any of these disc drives, proceed as follows:

1. Connect the disc drives via HP-IB to the HP-IL/HP-IB converter and connect the converter at any position in the HP-IL loop.
2. Set all the switches on the converter to zero, then connect the converter to ac power.
3. If the disc drive(s) are all dual floppy drives, include the following line in a CONFIG.SYS file on the A: drive:

DEVICE=B:\BIN\AMIGO.SYS

If the disc drive(s) are not dual floppies, the HP-IB address and number of volumes at each address must be provided. The following line should be included in a CONFIG.SYS file on the A: drive:

DEVICE=B:\BIN\AMIGO.SYS /0#4,2#1

The arguments imply that there is a four-volume disc at HP-IB address zero and a single-volume disc at address two. Note the space after "SYS".

4. Re-boot the Portable PLUS by pressing the **(CTRL)** **(Shift)** **(Break)** key combination to install the driver. The driver will automatically assign the appropriate drive letter to each disc drive. "Amigo" drives are assigned letters that follow the letters for "newer" external disc drives specified in the System Configuration Menu.

#### F.1.4 Portable PLUS-Desktop Link

If you already own an IBM PC, IBM XT, or HP 150, the best solution may be to buy one of the Portable PLUS-Desktop Links. These inexpensive products include an HP-IL interface for the desktop, plus software for the desktop that makes it behave like a peripheral for the Portable PLUS. The Portable PLUS can use the desktop's disc drives or printer as though they were directly connected. This allows easy file interchange (*although desktop programs will not necessarily run on the Portable PLUS*) and gives the Portable PLUS access to IBM-formatted discs. The software also allows the desktop to connect to HP-IL printers, disc drives, tape drives, and plotters.

The Portable PLUS-Desktop Link is described in further detail in appendix H, "Portable PLUS-Desktop Link."

## F.2 Media Compatibility

### F.2.1 Reading Other Discs on the Portable PLUS

The Portable PLUS, when equipped with the appropriate external disc drive, will read discs that are "HP MS-DOS" format. This includes both single and double-sided 3.5-inch flexible discs, 5.25-inch discs, and Winchester discs. All of these should have been formatted by an HP MS-DOS computer such as the HP 110 or HP 150. Data files will be directly usable, *but programs are not necessarily compatible.* (Refer to appendix A, "Comparisons With Other Computers.")

The Portable PLUS will not read non-MS-DOS discs such as the Integral or HP-86 discs, and may not directly read non-HP MS-DOS discs. However, the Portable PLUS-Desktop Link described in appendix H will allow the Portable PLUS to read and write on IBM PC discs.

The HP 9114A Disc Drive is capable of reading either single- or double-sided discs, so single-sided discs from an HP 150A can be used by the Portable PLUS. A single-sided disc modified by the Portable PLUS will still be usable by the HP 150 on a single-sided drive.

To transfer data from a computer with incompatible discs, use the Portable PLUS modem or RS-232 interface to do a file transfer.

### F.2.2 Reading Portable PLUS Discs on Other Computers

The HP 150B and HP 150C supports either single- or double-sided discs. Therefore discs formatted by the Portable PLUS are readable by these computers. Again, data on the disc is usable by either machine, *but programs are not necessarily compatible.* (Refer to appendix A, "Comparisons With Other Computers.")

With a Portable PLUS-Desktop Link, an IBM PC can connect to an HP 9114A and read discs formatted by the Portable PLUS. This gives PC owners access to a more rugged and portable storage media.

The HP 150A does not support double-sided, 3.5-inch disc drives. However, an HP 150-readable single-sided disc can be formatted on a double-sided drive by following this procedure:

1. Insert the blank disc to be formatted in drive C:.
2. Type **format c: /w** and press the **(Return)** key twice. There will be a delay while one side of the disc is formatted and checked for bad sectors.
3. When prompted, type any valid volume label and press the **(Return)** key or simply press the **(Return)** key for no volume label.
4. Press the **(Y)** key to format another single-sided disc or press the **(N)** key to stop the formatting process.

Anything subsequently stored on this disc will be readable by an HP 150A with a 3.5-inch single-sided disc drive.



**Note**

The HP 150's disc drive must be turned off and on again before switching to the Portable PLUS discs. This forces the HP 150 to reconfigure for the new disc format.



# Configuring Serial Printers

---

## G.1 Introduction

This section provides information on using seven different serial printers with the Portable PLUS. The printers covered are:

- HP 2225D ThinkJet Printer.
- HP 2601A Letter Quality Printer.
- HP 2686A LaserJet Printer.
- IDS-560 Impact Printer.
- NEC Spinwriter 3510.
- Xerox 610C1 Memorywriter.
- Xerox 625C Memorywriter.

Other serial printers may be used with the Portable PLUS by following procedures similiar to those described in this section.

---

## G.2 The HP 2225D ThinkJet Printer

**Hardware Requirements.** The minimum equipment required to use this printer with the Portable PLUS is listed below:

The Portable PLUS.

HP 2225D ThinkJet Printer with power cord, printhead, and paper.

HP 92221P Printer Cable.

**Connecting the Printer.** To connect the ThinkJet Printer to the Portable PLUS, plug the 9 pin connector of the HP 92221P Printer cable into the serial port on the back of the computer. Then plug the 25-pin connector into the back of the printer.

**Configuring the Printer.** To configure the ThinkJet Printer for use with the Portable PLUS, use the information in appendix I of the *ThinkJet Printer Owner's Manual*. Change the switch settings listed in table G-1 to the values specified.

**Table G-1. ThinkJet Switch Settings**

Segment	Setting	Description
1	DOWN	Enable Software Handshaking {XON/XOFF}
2	DOWN	Segment 2 & 3
3	DOWN	No Parity Checking, 8 bit word
4	DOWN	Segment 4 & 5
5	DOWN	Select 9600 Baud

**Configuring the Portable PLUS.** To configure the Portable PLUS, the following values must be set up in the PAM Datacom Configuration Menu and System Configuration Menu.

#### **Datacom Configuration**

Transmission Rate (BPS)	9600
Word Length (bits)	8
Stop Bits	1
Parity	None
XON/XOFF Pacing	On
CTS Line	Ignore
DSR Line	Ignore
DCD Line	Ignore

## System Configuration

G

Printer Interface	Serial
Printer Mode	Alpha and HP Graphics
Datacom Interface	Serial

---

### G.3 The HP 2601A Printer

**Hardware Requirements.** The minimum equipment required to use this printer with the Portable PLUS is listed below:

- The Portable PLUS.
- HP 2601A Printer with power cord and paper.
- HP 92221P Printer Cable.

**Connecting the Printer.** To connect the HP 2601A Printer to the Portable PLUS, plug the 9-pin connector of the HP 92221P Printer cable into the serial port on the back of the computer. Then plug the 25-pin connector into the back of the printer.

**Configuring the Printer.** To configure the HP 2601A for use with the Portable PLUS, use the information on pages 2-3 and 2-4 in section 2 of the *2601A Installation and Reference Manual*. Change the switch settings listed in table G-2 to the values specified. The switches that need to be changed are found under the front access cover.

**Table G-2. HP 2601A Switch Settings**

<b>Segment</b>	<b>Setting</b>	<b>Description</b>
1	ON	Select FULL Duplex mode
2	ON	Enable parity checking & transmission
3	ON	Select 300 Baud
4	OFF	This segment is not used
5	OFF	1200 Baud is not selected
6	ON	Select Even Parity
7	OFF	Paper out sensing is enabled
8	OFF	This segment is not used

**Configuring the Portable PLUS.** To configure the Portable PLUS, the following values must be set up in the PAM Datacom Configuration Menu and System Configuration Menu.

#### **Datacom Configuration**

Transmission Rate (BPS)	300
Word Length (bits)	7
Stop Bits	1
Parity	Even
XON/XOFF Pacing	On
CTS Line	Ignore
DSR Line	Ignore
DCD Line	Ignore

#### **System Configuration**

Printer Interface	Serial
Printer Mode	Alpha Only
Datacom Interface	Serial

## G.4 The HP 2686A LaserJet Printer

**Hardware Requirements.** The minimum equipment required to use this printer with the Portable PLUS is listed below:

The Portable PLUS.

HP 2686A LaserJet Printer with power cord and paper.

HP 92221P Printer Cable.

**Connecting the Printer.** To connect the LaserJet Printer to the Portable PLUS, plug the 9-pin connector of the HP 92221P Printer cable into the serial port on the back of the computer. Then plug the 25-pin connector into the back of the printer.

**Configuring the Printer.** To configure the LaserJet Printer for use with the Portable PLUS, there is nothing that the user needs to (or can) do. This printer comes from the factory set to the following specifications:

Baud Rate	9600
Parity	None
Databits	8
Stop Bits	1
Handshake	Both Software {XON/XOFF} & Hardware {DTR}

**Configuring the Portable PLUS.** To configure the Portable PLUS, the following values must be set up in the PAM Datacom Configuration Menu and System Configuration Menu.

#### Datacom Configuration

Transmission Rate (BPS)	9600
Word Length (bits)	8
Stop Bits	1
Parity	None
XON/XOFF Pacing	On
CTS Line	Ignore
DSR Line	Ignore
DCD Line	Ignore

#### System Configuration

Printer Interface	Serial
Printer Mode	Alpha Only
Datacom Interface	Serial

---

## G.5 The IDS-560 Impact Printer – The Paper Tiger

**Hardware Requirements.** The minimum equipment required to use this printer with the Portable PLUS is listed below:

The Portable PLUS.

IDS-560 Paper Tiger with power cord and paper.

HP 92221P Printer Cable.

HP 92222F female-to-female gender converter.

**Connecting the Printer.** To connect the IDS-560 Paper Tiger to the Portable PLUS, plug the 9-pin connector of the HP 92221P Printer cable into the serial port on the back of the computer then connect the HP 92222F female to female gender converter to the 25-pin connector. Finally, plug the 25-pin connector of the HP 92222F converter into the back of the printer.

**Configuring the Printer.** To configure the Paper Tiger for use with the Portable PLUS, use the information in section 3 of the printer manual to change the option items listed in table G-3 to the values specified.

**Table G-3. IDS-560 Switch Settings**

Segment	Setting	Description
1	ON	Segments 1, 2 & 3 select 11" paper
2	OFF	
3	ON	
4	ON	Segments 4 & 5 combine to select
5	OFF	9600 Baud transmission rate
6	OFF	Select Even Parity
7	ON	Enable Parity Checking



**Note**

This printer comes with a DB-25S (female 25-pin RS-232) connector and can be configured as either a serial or parallel printer. The instructions for selecting serial or parallel operation are covered in the printer manual in section 2.3.2 "Interface Configuration Strapping."

**Configuring the Portable PLUS.** To configure the Portable PLUS, the following values must be set up in the PAM Datacom Configuration Menu and System Configuration Menu.

#### Datacom Configuration

Transmission Rate (BPS)	9600
Word Length (bits)	7
Stop Bits	1
Parity	Even
XON/XOFF Pacing	On
CTS Line	Ignore
DSR Line	Ignore
DCD Line	Ignore

#### System Configuration

Printer Interface	Serial
Printer Mode	Alpha Only
Datacom Interface	Serial

---

## G.6 The NEC Spinwriter 3510

**Hardware Requirements.** The minimum equipment required to use this printer with the Portable PLUS is listed below:

- The Portable PLUS.
- NEC Spinwriter 3510 with power cord and paper.
- HP 92221P Printer Cable.

**Connecting the Printer.** To connect the NEC Spinwriter 3510 to the Portable PLUS, plug the 9-pin connector of the HP 92221P Printer cable into the serial port on the back of the computer. Then plug the 25-pin connector into the back of the printer.

**Configuring the Printer.** To configure the NEC Spinwriter 3510 for use with the Portable PLUS, use the information in section 3 of the printer manual to change the option items listed in table G-4 to the values specified.

**Table G-4. Spinwriter 3510 Switch Settings**

Segment	Setting	Description
1	OFF	Local Line Feed
2	ON	Select Even Parity
3	ON	Enable Parity Checking
4	ON	Select FULL Duplex

**Configuring the Portable PLUS.** To configure the Portable PLUS, the following values must be set up in the PAM Datacom Configuration Menu and System Configuraton Menu.

#### Datacom Configuration

Transmission Rate (BPS)	1200
Word Length (bits)	7
Stop Bits	1
Parity	Even
XON/XOFF Pacing	On
CTS Line	Ignore
DSR Line	Ignore
DCD Line	Ignore

#### System Configuration

Printer Interface	Serial
Printer Mode	Alpha Only
Datacom Interface	Serial

## G.7 The Xerox 610C1 Memorywriter

**Hardware Requirements.** The minimum equipment required to use this printer with the Portable PLUS is listed below:

The Portable PLUS.

Xerox 610C1 Memorywriter with power cord and paper.

HP 92221P Printer Cable.

**Connecting the Printer.** To connect the 610C1 Memorywriter to the Portable PLUS, plug the 9-pin connector of the HP 92221P Printer cable into the serial port on the back of the computer. Then plug the 25-pin connector into the back of the printer.

**Configuring the Printer.** To configure the Memorywriter for use with the Portable PLUS, follow the steps below:

1. Turn on the Memorywriter.
2. Use the information in the reference section on page 21 of the *610C1 Memorywriter Communications Handbook* for the Memorywriter to change the option items listed in table G-5 to the values specified.

**Table G-5. 610C1 Memorywriter Option Settings**

Option	Setting	Memorywriter Code	Default
Baud Rate	1200	6	300 [(4)]
Parity	Even	e	Even
Echo	Off	[Space Bar]	Off
Caplock	Off	[Space Bar]	Off
Carrier Return	On	x	Off [Space Bar]
Auto Linefeed	Off	[Space Bar]	On
ESC	On	x	On (Allow Esc. Seq's)
Answer Back (ENQ)	Off	[Space Bar]	Off

3. After pressing the printer's **(STOP)** key to exit the modification routine, the Memorywriter will be set to the proper state to communicate with the Portable PLUS. Note that the printer will stay in this state until you deliberately change an option or leave the Memorywriter turned off or unplugged for five days. (An internal battery maintains the settings.)

**Configuring the Portable PLUS.** To configure the Portable PLUS, the following values must be set up in the PAM Datacom Configuration Menu and System Configuraton Menu.

#### Datacom Configuration

Transmission Rate (BPS)	1200
Word Length (bits)	7
Stop Bits	1
Parity	Even
XON/XOFF Pacing	On
CTS Line	Ignore
DSR Line	Ignore
DCD Line	Ignore

#### System Configuration

Printer Interface	Serial
Printer Mode	Alpha Only
Datacom Interface	Serial

## G.8 The Xerox 625C Memorywriter

**Hardware Requirements.** The minimum equipment required to use this printer with the Portable PLUS is listed below:

The Portable PLUS.

Xerox 625C Memorywriter with power cord and paper.

HP 92221P Printer Cable.

**Connecting the Printer.** To connect the 625C Memorywriter to the Portable PLUS, plug the 9-pin connector of the HP 92221P Printer cable into the serial port on the back of the computer. Then plug the 25-pin connector into the back of the printer.

**Configuring the Printer.** To configure the Memorywriter for use with the Portable PLUS, follow the steps below:

1. Turn on the Memorywriter.
2. Using the procedure described under "Changing Options Settings" on page 26 of the *625C Memorywriter Communication Handbook* for the Memorywriter, change the option items listed in table G-6 to the values specified.

**Table G-6. 625C Memorywriter Option Settings**

Option	Alpha Identifier	Setting	Setting Code	Default
Echo	a	OFF	0	OFF
Cap Lock	b	OFF	0	OFF
Line Cntrl Char	c	CR	0	CR
Line Control	d	OFF	0	OFF
Point-to-Point	e	OFF	0	OFF
Baud Rate	f	1200	5	300 [(3)]
Duplex	g	FULL	1	FULL
Parity	h	EVEN	1	EVEN
Auto Line Feed	i	OFF	0	ON [(1)]
Stop Bits	j	1	0	2 [(2)]
Answer Back	k	OFF	0	OFF

3. After pressing the printer's **STOP** key to exit the modification routine, the Memorywriter will be set to the proper state to communicate with the Portable PLUS. Note that the printer will stay in this state until you deliberately change an option or leave the Memorywriter turned off or unplugged for five days. (An internal battery maintains the settings.)

**Configuring the Portable PLUS.** To configure the Portable PLUS, the following values must be set up in the PAM Datacom Configuration Menu and System Configuration Menu.

### Datacom Configuration

Transmission Rate (BPS)	300
Word Length (bits)	7
Stop Bits	1
Parity	Even
XON/XOFF Pacing	On
CTS Line	Ignore
DSR Line	Ignore
DCD Line	Ignore

### System Configuration

Printer Interface	Serial
Printer Mode	Alpha Only
Datacom Interface	Serial



## Portable PLUS- Desktop Link

The Portable PLUS-Desktop Link (PDL) is much more than a wire connection between a desktop computer and the Portable PLUS. It is the key to a powerful portable and desktop companion relationship. Compatibility of machines is brought about by the sharing of information and peripherals. The investment made in higher capability peripherals such as Winchester discs and letter quality printers can be spread across several machines by using the PDL with a desktop computer. The PDL makes it easy to transfer information to or from the Portable PLUS.

The PDL is the necessary hardware and software on a desktop computer that enables the Portable PLUS to easily communicate with the desktop computer. The hardware provides a low cost communications link using HP-IL. The software, which is provided by the PDL, is executed on the desktop computer.

Information can be transferred from the Portable PLUS to or from a disc that is attached to the desktop. The desktop computer simply passes the information to or from its disc drive to the Portable PLUS. The Portable PLUS can also use a printer which is attached to the desktop computer as if it were directly tied to the Portable PLUS. In those circumstances when it is more convenient to display the information on the screen rather than print it, you can treat the display on the desktop like a printer. Another capability which is provided for the desktop computer is the capability of using HP-IL peripherals such as the Hewlett-Packard ThinkJet Printer with the HP-IL interface. The PDL deals only with the connection between the Portable PLUS and the desktop.

## H.1 Portable PLUS to HP 150

The Portable PLUS-Desktop Link is implemented on the HP 150 by the HP 45643A Extended I/O Accessory. This product also provides a parallel printer interface. A cable for the printer is available separately.

The PDL provides the capability to pass information between the Portable PLUS and the HP 150 and allows the Portable PLUS to access the desktop's peripherals. In addition, the HP 150 may drive HP-IL peripherals.

**Example:** An executive who is out of the office a few days each week needs to write letters and reports while she is on the road. In her office, her secretary has an HP 150 with a letter quality printer. Her problem is to use the printer to print her letters without copying them first to the disc drives on the HP 150. The executive has no need to retain copies of most of her letters on a disc. She just wants to get letter quality printouts of her text using the printer connected to the HP 150.

Equipment needed:

HP 150 with dual disc drives.

Printer attached to the HP 150 that is installed as the PRN device.

HP 45643A Extended I/O Accessory.

The Portable PLUS.

2 HP-IL Cables.

1. Install the HP 45643A Extended I/O Accessory in the HP 150.
2. Connect the HP 150 to the Portable PLUS using HP-IL cables.
3. On the HP 150 insert the HP 45643A disc into drive A: and run HPLINK. When requested to select a device, type (2) **(Return)** for printer.
4. On the Portable PLUS, create and store a memo in the file A:MEMO.
5. Print the file MEMO on the Portable PLUS using PAM's File Manager Print File function.
6. After the memo has been printed, exit HPLINK on the HP 150 by typing (E) on the HP 150 keyboard.

## H.2 Portable PLUS to IBM PC/XT

On the IBM PC/XT, the Portable PLUS-Desktop Link is implemented by the HP 82973A HP-IL Interface Card. The hardware combined with the software provided with the PDL enables the Portable PLUS to efficiently transfer information to or from a desktop and utilize peripherals on the desktop computer. Using the driver provided, the IBM PC/XT may drive HP-IL peripherals.

**Example:** A food brokerage firm has five sales representatives who call on different wholesalers. The firm has developed a spreadsheet using 1-2-3 from Lotus on their IBM PC. This spreadsheet contains ordering information, discount levels, and other relevant information about the products represented by this brokerage. Each of the sales representatives has the Portable PLUS, which they carry with them on their sales calls. When the sales representative leaves the call, he will have a completed record of the call. The problem that the firm has is how to transfer a spreadsheet template from the IBM PC's disc drives to the Portable PLUS.

### Equipment needed:

- IBM PC with two flexible disc drives.
- HP 82973A HP-IL Interface Card.
- 1-2-3 from Lotus for the IBM PC.
- The Portable PLUS.
- 2 HP-IL Cables.

1. Install the HP 82973A interface card in the IBM PC.
2. Using 1-2-3 from Lotus, create a spreadsheet on the IBM PC and store it as the file "CALLER" on a disc in drive B.
3. Connect the IBM PC to the Portable PLUS using HP-IL cables.
4. On the IBM PC, insert the HP 82973A disc into drive A and run HPLINK. When requested to select a device, type **(D)** (**Return**) for disc.
5. On the Portable PLUS, copy D:CALLER to A: (Drive D: addressed from the Portable PLUS is the same as drive B: as addressed from the IBM PC.)
6. After the file has been copied and a message to that effect is displayed, return control of the IBM PC to the operating system by typing **(E)** on the IBM PC.

---

### **H.3 Portable PLUS to IBM AT**

The Portable PLUS-Desktop Link interface card for the IBM PC/XT (HP 82973A) can be used in the IBM AT. However, due to the differences in the disc formats in MS-DOS 3.00 and MS-DOS 2.11, *it is necessary that the Portable PLUS be configured as a disc drive that can be read and written to by the IBM AT.* Do this by running the built-in HPLINK program. The IBM AT can load the HPIL.SYS driver provided by the PDL and access the Portable PLUS and other HP-IL peripherals.

---

### **H.4 Portable PLUS to Portable (or Portable PLUS)**

The Portable PLUS can be linked to the Portable (HP 110) or to another Portable PLUS via HP-IL cables. The software necessary to pass information is built into both computers. Simply run HPLINK on one of the computers to simulate a disc drive and provide access to its built-in RAM and ROM discs through HP-IL. (Refer also to chapter 6 in *Using the Portable PLUS*.)

# Parts

This appendix lists accessories and parts that you can use with the Portable PLUS. The lists are separated into three groups:

- Accessories used with the Portable PLUS.
- Custom Hewlett-Packard parts, which may be made available to independent hardware vendors through HP sales offices. For information about becoming an independent hardware vendor, contact your local Hewlett-Packard sales office.
- Standard parts, which are available from non-HP vendors.

**Table I-1. Portable PLUS Accessories**

Model Number	Description
HP 82983A	300/1200 BPS Modem (internal)
HP 82982A	Software Module (ROM drawer)
HP 82981A	128K Memory Module
HP 82984A	128K Memory Card (add-on)
HP 82985A	Video Interface
HP 9114A	Portable 3-1/2" Disc Drive (battery-powered)
HP 88014A	Replacement battery
HP 2225B	ThinkJet Personal Printer (battery-powered)
HP 82199A	Replacement battery
HP 92205D	Acoustic Coupler
HP 92189A	Keyboard overlays (package of five)
	Serial interface cables:
HP 92221M	Modem cable (5-foot)
HP 92221P	Printer cable (5-foot)
HP 92221F	Female adapter (gender converter)
HP 92204E	Interface reconfiguration device

**Table I-1. Portable PLUS Accessories (Continued)**

HP 82973A	Portable-Desktop Link (HP-IL card for IBM PC/XT)
HP 45643A	Portable-Desktop Link (HP-IL card for HP 150)
	HP-IL cables:
HP 82167A	Cable (0.5-meter)
HP 82167B	Cable (1-meter)
HP 82167D	Cable (5-meter)
HP 82164A	HP-IL/RS-232-C Interface
HP 82169A	HP-IL/HP-IB Interface
HP 13269U	Leather conformal case
HP 13269V	System carrying bag
HP 13269W	Attache computer/brief case
	AC Rechargers:
HP 82159D	U.S. (110 Vac)
HP 82066B	Europe (220 Vac)
HP 82067B	U.K. (220 Vac)
HP 82067B Opt 001	U.K. with RSA plug (220 Vac)
HP 82068B	Australia (220 Vac)
HP 82069B	Europe (110 Vac)
HP 45419C	Programmer's Tools
45711-60055	Owner's documentation (English)
45711-60056	Owner's documentation (French)
45711-60057	Owner's documentation (German)
45711-60058	Owner's documentation (Italian)

**Table I-2. Custom HP Parts**

Part Number	Description
5001-6303	Software Module: ROM DIP carrier
45711-80021	Modem parts: Ground strap
0535-0063	Flanged hex nut (M3x0.5, 3.7 mm thick)
5061-4323	Plug-in module parts: Plastic drawer with inserts
5001-6301	Memory (RAM) module cover
82982-00001	Software (ROM) module cover
0515-0346	PCB screw (M2x0.4 x 4 mm, Torx T6)
0515-1369	Cover screw (M2x0.4 x 5 mm, Torx T6)
5001-6302	ESD ground spring
5061-2211	Recharger: Cable with built-in plug

**Table I-3. Standard Parts**

Part Number	Description
520252-2	RJ11C phone jack, right angle, PC mount: AMP Inc., Harrisburg, PA
10-91-2624	Plug-in drawer connector, 62-pin male: Molex Products Co., Lisle, IL
15-29-5012	Modem board connector, 12-pin male: Molex Products Co., Lisle, IL
10375-1	AMP Inc., Harrisburg, PA
65000-116	Dupont Connector Systems, New Cumberland, PA



# Escape Sequence Summary

This appendix summarizes the following escape sequences and control characters:

- Portable PLUS Control Characters
- HP Two-Character Escape Sequences
- HP Alpha & Graphics Escape Sequences

**Table J-1. Control Characters**

Char	Description
CTRL @	Null
CTRL G	Bell
CTRL H	Backspace
CTRL I	Horizontal Tab
CTRL J	Line Feed
CTRL M	Carriage Return
CTRL N	Select Alternate Font
CTRL O	Select Primary Font
CTRL Q	Resume Output
CTRL S	Stop Output
CTRL [	Escape

**Table J-2. Two-Character Escape Sequence Summary**

Seq	Description
ESC 0	Print Screen
ESC A	Cursor Up
ESC B	Cursor Down
ESC C	Cursor Right
ESC D	Cursor Left
ESC E	Reset Console
ESC F	Home Down
ESC H	Home Up
ESC I	Tab Forward
ESC J	Clear to End Of Display Memory
ESC K	Clear to End Of Line
ESC L	Insert Line
ESC M	Delete Line
ESC P	Delete Character
ESC Q	Insert Character ON
ESC R	Insert Character OFF
ESC S	Roll Up
ESC T	Roll Down
ESC U	Next Page
ESC V	Previous Page
ESC Y	Display Functions ON
ESC Z	Display Functions OFF
ESC ^	Read Primary Status
ESC `	Read Relative Cursor Position
ESC a	Read Absolute Cursor Position
ESC d	Enter Line
ESC h	Home Up
ESC i	Tab Backward

**Table J-3. HP Alpha Escape Sequence Summary**

Escape Sequence			Description
ESC & a	0..79	C	Move to Absolute Column
ESC & a	0..61	R	Move to Absolute Row
ESC & a	0..79	X	Move to Screen Column
ESC & a	0..24	Y	Move to Screen Row
ESC & a	+/-n	C	Move to Relative Column
ESC & a	+/-n	R	Move to Relative Row
ESC & a	1..2	D	Arabic/Latin Mode Select
ESC & d	{@-0}		Select Display Enhancement
ESC & d	@		Normal
ESC & d	A		Blinking only
ESC & d	B		Inverse Video only
ESC & d	D		Underline only
ESC & d	H		Halfbright only
ESC & f	0..2	A	Softkey Attribute
ESC & f	-1..80	D	Softkey Label Length
ESC & f	1..8	K	Softkey Key Select
ESC & f	-1..80	L	Softkey String Length
ESC & j		@	Softkey Labels OFF
ESC & j		B	Softkey Labels ON
ESC & k	0..1	\	Alt Mode Keyboard OFF/ON
ESC & k	0..1	A	Auto Linefeed OFF/ON
ESC & k	0..1	D	Bell OFF/ON
ESC & k	0..3	O	Select Keyboard Mode - 0 Turn off keyboard modes 1 Turn on Numeric Keypad 2 Turn on Scancode Mode 3 Turn on Modifier Mode
ESC & k	0..1	P	Caps Lock OFF/ON
ESC & s	0..1	A	Transmit Functions OFF/ON
ESC & s	0..1	C	End-Of-Line Wrap ON/OFF

**Table J-4. HP Graphics Escape Sequence Summary**

Escape Sequence		Description	
ESC * d		A	Clear Graphics Memory
ESC * d		B	Set Graphics Memory
ESC * d		C	Graphics ON
ESC * d		D	Graphics OFF
ESC * d		E	Alpha ON
ESC * d		F	Alpha OFF
ESC * d		K	Block Cursor/Gcursor ON
ESC * d		L	Uline Cursor/Gcursor OFF
ESC * d	x,y	O	Position Gcursor Absolute
ESC * d	x,y	P	Position Gcursor Relative
ESC * d	0..1	Q	Alpha Primary/Secondary Cursor
ESC * d		R	Alpha Cursor OFF
ESC * m	0..4	A	Set Drawing Mode
ESC * m	1..10	B	Set Line Type
ESC * m	p,s	C	Set Line Pattern and Scale
ESC * m	x,y	J	Set Relocatable Origin
ESC * m		K	Set Rel Origin At Pen Position
ESC * m		L	Set Rel Origin At Gcursor
ESC * m		R	Set Graphics Defaults
ESC * p		A	Lift Pen
ESC * p		B	Lower Pen
ESC * p		C	Move Pen to Gcursor Position
ESC * p		D	Draw Dot at Pen Position
ESC * p		E	Set Rel Origin at Pen Position
ESC * p		F	Use ASCII Absolute Format
ESC * p		G	Use ASCII Incremental Format
ESC * p		H	Use ASCII Relocatable Format
ESC * p		Z	NOP/Synch
ESC * p	x,y		Move Pen
ESC * s	1/110	S	Read Model/Serial Number

**Table J-5. ANSI Escape Sequence Summary**

Escape Sequence		Description	
ESC [	p1	A	Cursor Up <p1> Lines (CUU)
ESC [	p1	B	Cursor Down <p1> Lines (CUD)
ESC [	p1	C	Cursor Right <p1> Columns (CUR)
ESC [	p1	D	Cursor Left <p1> Columns (CUL)
ESC [	p1;p2	H	Move to Absolute Position (CUP)
ESC [	p1	J	Erase Display - <ul style="list-style-type: none"> <li>0 Clear to end of display</li> <li>1 Clear to start of display</li> <li>2 Clear entire display</li> </ul>
ESC [	p1	K	Erase Line - <ul style="list-style-type: none"> <li>0 Clear to end of line</li> <li>1 Clear to beginning of line</li> <li>2 Clear entire line</li> </ul>
ESC [	p1	L	Insert <p1> Lines (IL)
ESC [	p1	M	Delete <p1> Lines (DL)
ESC [	p1	P	Delete <p1> Characters (DCH)
ESC [	p1;p2	R	Cursor Position Report (CPR)
ESC [	p1	U	Next <p1> Pages (NP)
ESC [	p1	V	Previous <p1> Pages (PP)
ESC [	p1;p2	f	Move to Absolute Position (HVP)
ESC [	p1	h	Set Mode - <ul style="list-style-type: none"> <li>4 Insert Character ON</li> <li>=0 Alpha Mode ON</li> <li>=1 Alpha Mode ON</li> <li>=2 Alpha Mode ON</li> <li>=3 Alpha Mode ON</li> <li>=4 Graphics Mode ON</li> <li>=5 Graphics Mode ON</li> <li>=6 Graphics Mode ON</li> <li>=7 End-Of-Line Wrap ON</li> <li>=8 Alpha Mode ON</li> <li>=10 Graphics Mode ON</li> <li>?7 End-Of-Line Wrap ON</li> </ul>

**Table J-5. ANSI Escape Sequence Summary (Continued)**

ESC [	p1	1	Reset Mode - 4 Insert Character OFF =0 Alpha Mode OFF =1 Alpha Mode OFF =2 Alpha Mode OFF =3 Alpha Mode OFF =4 Graphics Mode OFF =5 Graphics Mode OFF =6 Graphics Mode OFF =7 End-Of-Line Wrap OFF =8 Alpha Mode OFF =10 Graphics Mode OFF ?7 End-Of-Line Wrap OFF
ESC [	p1	m	Set Graphics Rendition - 0 All attributes OFF 1 Halfbright ON 4 Underline ON 5 Blinking ON 7 Inverse ON 10 Use HP Fonts 11 Use Alt Fonts
ESC [	6	n	Device Status Report (DSR)
ESC [		s	Save Cursor Position (SCP)
ESC [		u	Restore Cursor Position (RCP)

# K

## Software Module Configurations

---

### K.1 Overview

The HP 82982A Software Plug-in Module for the Portable PLUS provides a means for integrating additional ROM-based software into the computer system. This module has a storage capacity of 1.5M-bytes and supports up to twelve "ROM-disc" software packages.

The configuration of the module can be modified to accept a variety of combinations of ROMs and EPROMs. The ROM/EPROM Module is shipped from the factory configured to accept 6 pairs of ROMs. These ROMs can be a variety of sizes, as long as they meet the pin-out and performance requirements described in a following section. They can be matched pairs (allowing the possibility for executing code directly out of ROM/EPROM) or a pair can be two independent applications, each formatted as a stand-alone directory of files.

The configuration of the module can be modified by re-positioning one or more of the six jumpers wires mounted between the ROM/EPROM sockets. The positioning of these jumpers for all possible configurations is described in table K-1.

Two, four, or six of the ROM-socket pairs can be configured to accept 32Kx8 EPROMs instead of ROMs. These banks can be treated independently (as if they were ROMs) or the jumpers can be configured to allow 4 pairs of EPROMs to be cascaded, allowing them to be mapped into the entire 256K-bytes of memory space allocated for a ROM bank. This feature allows emulation of two 128K-byte ROMs using eight 32K-byte EPROMs (or potentially eight 32K-byte ROMs). The remaining two socket pairs always operate as independent banks.

**Table K-1. Wire Jumper Connections for ROMs/EPROMs**

Configuration	Wire Jumper Connections					
	XW1	XW2	XW3	XW4	XW5	XW6
<b>Small Group:</b>						
Banks 0 and 1 = Independent ROM Pairs	A	-	-	-	-	-
= Independent EPROM Pairs	B	-	-	-	-	-
<b>Large Group:</b>						
Banks 4, 5, 6, 7 = Independent ROM Pairs	-	A	A	A	A	A
= Independent EPROM Pairs	-	B	-	A	A	A
Bank 7*= Up to 4 cascaded pairs of 32Kx8 ROMs	-	A	B	B	B	B
= Up to 4 cascaded pairs of 32Kx8 EPROMs	-	B	B	B	B	B
<b>Notes:</b> A "—" in a particular column means that the jumper does not affect the bank configuration.						
Banks 2 and 3 are not present in this module.						
The factory configuration is for all independent ROM pairs (all jumpers in position A).						
* When the jumpers are configured for cascaded pairs of 32K-byte ROMs/EPROMs, the ROMs/EPROMs must be installed in the order 5-6-7-4 (that is, sockets 5H, 6H, 7H, 4H become bank 7H, and sockets 5L, 6L, 7L, 4L become bank 7L). Banks 4, 5, and 6 are not present in this configuration.						

## K.2 Plug-In ROMs and EPROMS

ROMs (or EPROMs) that are to be installed in the HP 82982A Software Drawer must meet certain specifications to ensure proper system operation. These specifications are listed in table K-2.

Table K-2. Plug-In ROM Specifications

Parameter	Specification
<b>Physical:</b>	
IC package	ROM: 128K x 8 through 8K x 8 EPROM: 32K x 8
Operating temperature	Jedec 28-pin dual in-line
Storage temperature	0° to 55°C (32° to 131°F) -25° to 55°C (-13° to 131°F)
<b>Electrical:</b>	
Access time (CS* to data valid)	363 ns max. (150 pF load)
Output enable time (OE* to data valid)	363 ns max. (150 pF load)
Data hold time	0 ns min.
Deselect time (CS* high)	90 ns min.
Cycle time	935 ns
Address setup time (address valid to CS*)	0 ns min.
Output capacitance (data lines)	15 pF max.
Output capacitance (address lines)	10 pF max.
Power supply voltage	5 Vdc ± 5%
Operating current	50 mA max.
Stand-by current	1 mA max.

Figure K-1 illustrates the pin configuration for ROMs and EPROMs used in the software drawer. A jumper in the drawer determines the connection for pin 1 based on whether the IC is a ROM or an EPROM.

Figure K-1. Pin Configuration for Plug-In ROM

++	1	28	Vcc
A12	2	27	A14
A7	3	26	A13
A6	4	25	A8
A5	5	24	A9
A4	6	23	A11
A3	7	22	A16/0E*
A2	8	21	A10
A1	9	20	CS*
A0	10	19	D7
D0	11	18	D6
D1	12	17	D5        ++ Set by jumper
D2	13	16	ROM: A15
GND	14	15	EPROM: Vpp

K  
Key:  
CS\* = Chip Select (asserted low)  
OE\* = Output Enable (asserted low)  
VPP = EPROM secondary supply voltage (operating: 5 volts)

Note: The numbering scheme used for the address signals shown on this diagram and used by integrated circuit manufacturers is offset by one in relation to the signal names shown on the schematic diagram of the ROM module. The pin labeled A0 in this diagram is connected to signal MA1 on the schematic diagram, A1 to MA2, etc.



It is generally possible to use a ROM/EPROM with secondary "Chip Select\*" (asserted low) lines in the positions of "No Connect" pins which are otherwise used as address lines by larger memory components. Mirror images are not required by the system and these address lines will be driven low when accessing these ROMs/EPROMs.

## K.3 Detailed Description

### K.3.1 ROM/EPROM Organization Options

A software package can be stored in a variety of configurations depending on its size, whether or not code is to be executed directly out of ROM/EPROM, and whether the package is to be stored in ROM or EPROM. The typical storage configurations of a single software package are described in the following table.

Memory sizes shown in table K-3 refer to the number of bytes of storage in the ROM/EPROM (memory components organized to have 8-bit parallel access). These memory sizes are rounded up to correspond to the size of typically-used parts. The abbreviations "hb" and "fb" refer to "Half-Bank" and "Full-Bank", respectively. These terms are described in the text following the table. Note that only 32K-byte EPROMs can be used in this module (any size ROM can be used).

Table K-3. ROM/EPROM Organization Options

Software Package Size	Typical Storage Options:	
	ROM	EPROM
32K	1 hb 32K ROM	1 hb 32K EPROM
33K-64K	1 hb 128K ROM	2 fb or hb 32K EPROMs
65K-128K	1 hb 128K ROM	3-4 hb 32K EPROMs
129K-256K	2 fb 128K ROMs	5-8 fb 32K EPROMs
257K+	Software package must be partitioned into 2 or more <i>pseudo-independent</i> packages, each <= 256K in size.	

The term "pseudo-independent" refers to the fact that each package must independently support the firmware-interface requirements of the Portable PLUS system, yet firmware support is provided to allow the code from one partitioned section of a package to "call" code from another section (refer to chapter 9).

A ROM "bank" refers to a section of ROM-based memory which can be mapped into a 256K-byte memory space of the Portable PLUS system. Only one bank can be mapped into the system at a time. The terms "Half-Bank" and "Full-Bank" refer to the memory organization of sequential bytes of a file stored in ROM disc. A selected ROM "bank"

can be either empty, it can contain a "full-bank", or it can be composed of 1 or 2 "half-banks". The following is a description of these two forms of the Portable PLUS ROM disc memory organization. (Generally, references to a "ROM" can be interpreted to refer to either a ROM or an EPROM.)

The architecture of the Portable PLUS uses a 16-bit parallel data bus. The typical memory organization for this type of data bus would utilize two 8-bit wide memory storage components for each section of memory, one component being selected on the even addresses and the other component on the odd addresses. Thus sequential bytes of a file stored in such a memory organization would be contained in alternate memory storage components.

ROM-based software packages using this form of memory organization are referred to as "full-bank" ROMs. Full-bank ROMs always come in pairs, with one ROM designated to go in a "Low-Byte" socket (even addresses) and the other to go in the adjacent "High-Byte" socket (odd addresses). Because this form of memory organization is consistent with the architecture of the CPU, this is the only form of ROM-based software which has the option of executing directly out of ROM. For further details about executing code directly out of ROM, refer to chapter 9.

When a ROM-based software package is organized as a "half-bank" ROM, sequential bytes in the files of the package are stored sequentially within the ROM memory component itself. Since this form of organization is inconsistent with the architecture of the CPU, the firmware of the Portable PLUS must re-align the bytes of any of these files when transferring data from a data file to a requesting process or when downloading an executable file into system memory before executing it. Since half-bank ROMs are not directly executable and since the firmware can handle these files at either even or odd addresses, a half-bank ROM can be placed in either a "High-Byte" or "Low-Byte" socket. The adjacent socket can either be empty or it can contain another half-bank software package.

### K.3.2 Jumper and Socket Labeling

The Portable PLUS Software Module provides 12 sockets for either ROMs or EPROMs. Since the use of EPROMs generally involves different circuitry than for ROMs, the Software Module has 6 wire jumpers which can be used to re-configure the hardware for the various modes of operation.

The Software Module comes configured from the factory for use with ROMs only. When the plug-in module is used in this mode, the only circuit board labels needed to install ROMs are the 6 large "L"s and 6 "H"s located on each socket of the Software

Module circuit board. The ROMs of full-bank software packages will be labeled with either an "H" or an "L", corresponding to the label on the socket in which those ROMs are to be installed. The only other requirement is that the two ROMs of a full-bank ROM pair must be installed in adjacent sockets. The ROMs of half-bank software packages will not be labeled with either an "L" or an "H"; these can be placed in any available socket.

The use of EPROMs on the Portable PLUS Software Module requires a more detailed inspection of the jumper and socket labeling on the Software Module circuit board. The 6 jumpers are labeled "XW1" through "XW6" with an "A" and a "B" on each end of each jumper socket. The "A" and "B" correspond to the two possible positions of the wire jumper installed in the jumper socket. Each of the 12 ROM sockets are labeled at the opposite end of the socket opposite from its nearest jumper. These small socket labels are single-digit numbers followed by either an "L" or an "H".

It is generally necessary to locate the labels just described in order to use the Software Module with EPROMs.

### K.3.3 Jumpers and Socket Groups

There are two ways to use EPROMs in the Software Module. First, an EPROM can be used as if it were a 32K ROM. In this mode, an EPROM can be used for a 32K half-bank software package or 2 EPROMs can be used for a 64K full-bank package. The other mode of EPROM usage provides a means to emulate ROMs which are larger than 32K-bytes using multiple 32K-byte EPROMs.

To allow for the possibility of using both ROMs and EPROMs simultaneously, the configuration circuitry associated with selecting between ROM and EPROM usage controls two groups of ROM/EPROM sockets independently. Since the configuration of either one of the two groups has no effect on the operation of the other group, the configuration procedure for each group will be described separately.

### K.3.4 Configuration of the Small Group

The "small group" of ROM sockets consists of those sockets with the following labels: "0L", "0H", "1L", "1H". These 4 sockets reside in the upper left-hand corner of the plug-in module (with the module connector facing towards you). Regardless of jumper settings, these four sockets are *always configured as two banks*, "bank 0" and "bank 1"--the bank number corresponds to the first digit of the socket labels.

This group is affected by only the jumper labeled "XW1". "XW1" selects whether this group is to be used with ROMs or EPROMs.

When the jumper is in the "A" position, the sockets of this group can be used with 32K-, 64K-, or 128K-byte ROMs.

When jumper "XW1" is in the "B" position, the sockets of this group can be used with 32K-byte EPROMs only. (32K-byte ROMs can be used if pin 1 is an active-high chip select or is not connected.) These EPROMs can contain a software package that is organized as either a 32K half-bank package or a 64K full-bank package.

This group cannot be used to emulate ROMs larger than 32K-bytes.

### K.3.5 Configuration of the Large Group

The "large group" of ROM sockets consists of those sockets with the following labels: "4L", "4H", "5L", "5H", "6L", "6H", "7L", "7H". Six of these eight sockets reside in the right half of the Software Module, with the remaining two sockets residing in the lower left-hand corner of the plug-in module (with the plug-in oriented such that the connector is facing towards you). This group can be configured as four separate banks (banks 4 through 7) or as a single bank (bank 7), depending on the position of jumpers "XW3" through "XW6".

Unlike the "small group" of sockets previously described, this group has three modes of operation. When jumpers "XW3", "XW4", "XW5", and "XW6" are all in position "A", jumper "XW2" controls this group in exactly the same way that "XW1" controls the small group (selecting either ROMs or EPROMs). When "XW2" is in position "A", sockets 4-7L,H can be used only with ROMs. When "XW2" is in position "B", sockets 4-7L,H can be used only with 32K EPROMs that are configured as 32K-byte half-bank singles or 64K-byte full-bank pairs (including combinations of these).

The remaining mode of operation is provided to allow 32K-byte EPROMs to be cascaded to emulate half-bank or full-bank software packages that are larger than 32K- or 64K-bytes, respectively. This mode is selected by positioning the six jumpers "XW2"- "XW6" all in the "B" position. *There is no valid configuration where jumpers "XW3"- "XW6" are not either all in position "A" or all in position "B".*

In this mode of operation the sockets labeled "5L", "6L", "7L", and "4L" are sequentially combined to form a single 128K-byte half-bank on the Low-byte data bus. Each socket can have a 32K-byte EPROM installed which comprises one-fourth of a 128K-byte half-bank. *The order of the sockets that comprise this half-bank is*

"5-6-7-4". The sockets "5H", "6H", "7H", and "4H" correspond to the High-byte half-bank and are organized in the same way as the Low-byte half-bank just described.

You can emulate 128K-byte ROMs using 32K-byte EPROMs. The EPROMs can be used in place of one or two 128K-byte half-bank software packages or one 256K-byte full-bank package. Four 32K-byte EPROMs are required for each 128K-byte ROM to be emulated. Fewer than four EPROMs can be used to emulate smaller ROMs or for 128K-byte ROMs having 32K-bytes of unused memory. The only requirements for this are that the EPROMs are installed in the proper order and that their software configuration accurately reflects the ROM-based software package being emulated.



# Index

---

## A

Accessories, I-1 thru I-2  
Accessory IDs  
    reading on HP-IL, 5-71  
    searching on HP-IL, 5-68, searching on  
        HP-IL, 5-70  
Alarm  
    affected by time zone, 5-49, 5-50  
    affects PPU internal status, 5-60, 5-63  
    enabling interrupts for, 5-59  
    from CLOCK device, 6-22  
    generates interrupt 46h, 5-25  
    operation, 10-16 thru 10-17  
    reading and setting, 5-48  
Alpha mode  
    initializing, 5-38, 5-89  
    selecting, 6-36  
    use of display RAM, 5-96 thru 5-98,  
        7-29 thru 7-34  
Alternate character set, 6-23, 6-45,  
    10-10, D-2 thru D-8  
Alternate mode  
    affects keycodes, 6-47, 13-1  
    affects operation of system keys, 5-81  
        thru 5-84  
    character sets, 5-97, D-1  
CONsole driver mode, 6-45  
keyboard-break interrupt active, 5-21  
reading status of, 6-52  
selecting, 6-34, 6-50, 10-10  
ANSI escape sequences. *See* Escape  
    sequences

Application programs, using with PAM,  
    10-4 thru 10-5  
Arabic mode, selecting, 5-98, 6-32  
Arena, 5-94, 5-99  
Attribute byte  
    about to be stored, 5-90  
    IBM compatibility, A-4  
    operation, 5-6, 7-33  
    reading and writing, 5-9, 5-102 thru  
        5-109, 5-111  
    stored in display RAM, 7-29  
    used by fast alpha, 5-96  
Autoanswering, 10-17 thru 10-18  
AUTOEXEC.BAT files, boot execution,  
    10-2 thru 10-3, 10-5, 11-1, 11-2,  
    11-5  
Automotive recharger, 2-22  
AUX device  
    controlling, 6-9 thru 6-15  
    reading status, 6-11, 6-15  
    relation to interrupt 14h, 5-13  
    setting communications parameters,  
        6-13 thru 6-15  
    specifying, 6-1, 10-12  
AUX driver  
    processing incoming data, 5-85 thru  
        5-86  
    provides modem control, 2-25, 12-2  
    uses interrupt 14h, 5-13 thru 5-16  
AUX expansion interrupt. *See* Int 5Dh  
Awake mode, 2-5, 2-10, 2-45

## B

**Battery.** *See also* Battery indicator  
conserving power, 6-7, 6-8, 6-9, 10-7,  
    10-9  
enabling interrupts when low, 5-59  
level affects sleep mode, 5-45  
level activates interrupts, 5-26, 5-29  
low-level condition, 4-1, 5-26, 5-29  
PPU status when low, 5-60  
recharging, 2-19 thru 2-23  
specifications, 2-2

**Battery indicator**  
affects PPU internal status, 5-63  
level after stop mode, 2-7  
operation, 2-20, 5-59, 10-18  
reading, 5-60  
setting current estimates, 5-47, 5-57

**Baud rate**  
CPU counting, 7-13  
determines transmission period, 6-8  
setting, 5-14, 6-14, 7-12 thru 7-13,  
    10-13, 12-3 thru 12-4

**Beeper**  
activating, 5-62  
part of mainframe, 2-7  
setting frequency and duration, 5-56,  
    10-10  
specifications, 2-2

**BIOS, interrupt descriptions**, 5-1 thru  
    5-127

**Blinking enhancement**  
operation, 7-33, 7-34  
reading status of, 6-52  
selecting, 6-33, 6-44  
set by attribute byte, 5-6, 5-96

**Boot**  
doesn't occur after sleep, 5-80  
effects, 5-97 thru 5-98, 6-9, 10-1,  
    10-2  
sequence of execution, 11-1 thru 11-5

## Boot code

contained in ROM, 8-1  
executing at boot, 11-1, 11-2 thru  
    11-4  
in plug-in ROMs, 9-1, 9-3, 9-4, 9-5,  
    9-7

**Break, sending**, 6-9

**Break key interrupt.** *See* Int 58h

**Bus cycles, description**, 2-9

## C

**CCITT V.24 standard**, 2-13 thru 2-18

**CCITT V.28 standard**, 2-13 thru 2-18

**Character fonts.** *See* Fonts

**Character sets, listed**, D-2 thru D-8

**Characters.** *See also* Attribute byte  
about to be displayed, 5-90  
fast display, 1-2, 1-3, 5-39  
muted, 13-1, 13-42 thru 13-43  
printing, 5-19

reading and writing, 5-5, 5-9, 5-10,  
    5-23, 5-37, 5-101 thru 5-109,  
    5-111

reading from font, 5-111

reading from key queue, 5-17

sending and receiving on datacom, 5-15

**Clock.** *See* Time and date

**CLOCK device**, 6-1, 6-22

**Clocking circuit**, 2-9

**COM1 device**, 6-1, 6-9 thru 6-15

**COM2 device**, 6-1, 6-9 thru 6-15

**COM3 device**, 6-1, 6-9 thru 6-15

**Communications devices.** *See* AUX device

**Communications interrupt.** *See* Int 14h

**CON driver.** *See* CONsole driver

**CON expansion interrupt.** *See* Int 5Eh

**CONFIG.SYS files**

effects, 10-21, 11-1, 11-4, F-1, F-3  
structure, 11-5 thru 11-6

**Configuration EPROM**

address range, 2-10 thru 2-12

affects keycodes, 6-47  
assembler listing, C-2 thru C-29  
can contain boot code, 11-1, 11-3,  
    11-4  
displaying string, 5-40  
maps keyboard, 13-1  
reading string from, 5-40  
revision shown in reboot display, 4-5  
sets default fonts, 5-97  
types of information, 2-10 thru 2-11,  
    C-1

Configuration files. *See* CONFIG.SYS files  
Configurations, 10-6 thru 10-15. *See also*  
    System configuration; Datacom  
    configuration

CONsole driver

- data just processed, 5-90
- defined, 6-1
- IOCTL functions, 6-50 thru 6-52
- mode affects keycodes, 13-1
- operation, 6-23 thru 6-52
- preprocessing data, 5-87 thru 5-90
- reading status of, 6-29, 6-50, 6-52
- resetting, 6-26
- uses fast alpha, 5-96

CONsole output, display option, 1-2 thru  
    1-3

Contrast, setting for display, 5-58

Control characters, 6-24 thru 6-25, J-1

Control-C, break code, 4-1, 5-21, 6-48

Country specification, 2-11

CPU, 2-2, 2-9, 5-59

Cursor

- defined, 5-94
- in graphics mode, 5-113
- low-level operation, 7-33 thru 7-34
- moving, 5-7, 5-101, 5-118, 6-26,  
    6-27, 6-31, 6-32, 6-37, 6-41,  
    6-43, 7-36
- reading position, 5-8, 5-100, 5-117,  
    6-30, 6-42, 7-36

reading type of, 6-52  
selecting type, 5-7, 5-101, 6-37, 7-34,  
    7-35, 10-10  
turning on and off, 6-37, 7-36  
underscore at boot, 5-38, 10-3

## D

Datacom. *See also* Int 14h

- handshake line status, 5-16
- reading error conditions, 5-16
- reading status, 5-16
- sending and receiving characters, 5-15
- setting parameters, 5-14
- turning port on and off, 5-51, 5-52

Datacom configuration, 10-12 thru  
    10-15, 10-21, 11-6

Datacom devices. *See* AUX device

Date. *See* Time and date

Death interrupt, 5-26

Desktop link, F-3, H-1 thru H-4, I-2

Device drivers, 5-65, 5-87, 6-1 thru  
    6-52

Diagnostics, 8-1, 11-1, 11-2

Dimensions, 2-2, 3-3

Disc drives

- accessing, F-1 thru F-5
- Amigo drives, F-2 thru F-3
- media compatibility, F-4 thru F-5
- presence, 5-12
- setting number, 10-7

Discs, compatibility, F-4 thru F-5

Display (hardware), 2-2, 2-7, 6-23, 7-26

Display (image). *See also* Alpha mode;  
    Graphics mode; Display modes;  
    Display RAM

- blanking, 7-35
- clearing, 5-89, 6-26, 6-27, 6-41
- contrast, 5-58
- erasing graphics within, 5-114
- fast write, 1-2, 1-3, 5-39
- initializing, 5-114

- modes. *See* Display modes  
options for accessing, 1-2 thru 1-3  
printing, 6-26  
reading attribute bytes from, 5-102  
    thru 5-106  
reading characters from, 5-101 thru  
    5-106  
scrolling, 5-8, 5-9, 6-28  
enhancements. *See* Blinking; Inverse;  
    Underlining; Halfbright  
selecting page, 5-8  
showing configuration EPROM string,  
    5-40  
turning on and off, 6-36  
writing attribute bytes to, 5-103 thru  
    5-109, 5-111  
writing characters to, 5-37, 5-103 thru  
    5-109, 5-111
- Display controller, low-level operation,  
    7-26 thru 7-37
- Display functions, 6-28, 10-3
- Display modes. *See also* Alpha mode;  
    Graphics mode; Display  
accessed by CONsole driver, 6-23  
initialized, 5-89  
provided by display controller, 7-26  
reading, 5-11  
selecting, 5-7, 5-114, 6-43, 7-35
- Display RAM  
arena for fast alpha, 5-94  
character about to be stored, 5-90  
clearing, 5-38, 6-26, 6-27, 6-41  
contained in display controller, 7-26  
CPU access, 7-26, 7-35  
operation, 5-5, 5-96, 5-113, 7-26 thru  
    7-32  
separate from other RAM, 8-1  
used for boot code, 9-7, 11-2 thru  
    11-4
- Drawers. *See* Plug-in drawers
- Drive A. *See* Edisc
- Drive B. *See* ROM disc
- E**
- EBCDIC, 5-85
- Edisc  
as device driver, 6-1  
boot sector, 8-5 thru 8-6  
boundary controlled by PAM, 8-3  
built-in disc drive, F-1  
can contain help file, 10-19  
checking integrity, 5-35  
checksums, 8-1, 8-4  
contained in RAM, 8-1  
default drive at boot, 11-4  
default environment source, 10-1  
erased by reset button, 4-3  
erased by security violation, 5-52, 5-53  
formatting, 5-35, 8-2  
not included by interrupt 12h, 5-13  
organization, 8-3 thru 8-6  
partition, 10-6  
reading data limit, 5-36  
secured sleep condition, 5-52, 5-53
- EIA RS-232-C standard, 2-13 thru 2-18
- 82164A device, 6-2, 6-9 thru 6-15
- Electrical design, 2-1 thru 2-47
- ENQ/ACK protocol, 6-7, 6-14
- EPROMs. *See* Plug-in drawers; Software  
    drawers; Plug-in ROMs
- Error conditions, reading on datacom,  
    5-16
- Escape sequences  
    adding new, 5-87, 5-88, 5-90  
    descriptions, 6-26 thru 6-44  
    summary, J-2 thru J-6
- F**
- Fast alpha. *See* Int 5Fh
- Fast graphics. *See* Int 5Fh
- Fast video interrupt. *See* Int 5Fh
- Files, loading, E-1 thru E-2

Firmware interrupts, 5-4  
Fonts  
    adding new, 5-39  
    alternate font, 6-23  
    contained in configuration EPROM,  
        2-11  
    default, 5-38, 10-3  
    HP font, 5-38, 6-23  
    in graphics mode, 5-23, 5-97  
    loading, 5-97, 5-111  
    organization in display RAM, 7-31  
        thru 7-32  
    reading character from, 5-111  
    replacing, 5-87, 5-89, 5-97 thru 5-98  
    selecting, 6-25, 6-44  
    stored in display RAM, 5-96, 7-29  
    structure, 5-97 thru 5-98  
Fuel gauge. *See* Battery indicator  
Function keys. *See* Softkeys

**G**  
Graphics, options for displaying, 1-2  
Graphics mode. *See also* Display modes  
    attributes, 5-115, 5-116  
    clearing memory, 6-36  
    copying area, 5-119  
    cursor operation, 5-113  
    cursor position, 5-117, 5-118  
    default condition, 6-39  
    drawing lines, 5-115  
    drawing mode, 5-89, 5-115, 5-116,  
        5-119, 6-37  
    initialized, 5-89  
    moving cursor, 6-37  
    pen operation, 5-113, 5-117, 6-39,  
        6-40  
    reading and setting pixels, 5-10, 5-116  
    reading area, 5-118  
    selecting line type, 5-89, 6-38  
    turning display on and off, 6-36  
use of display RAM, 5-99, 5-113, 7-26  
    thru 7-28

user character font, 5-5, 5-23  
using alpha characters, 5-5  
writing area, 5-119

## H

Halfbright enhancement  
    character set, D-1  
    in attribute byte, 5-6  
    reading status of, 6-52  
    selecting, 6-33, 6-44  
    uses font 2, 5-97  
Handshake, 5-16, 6-2 thru 6-7, 6-15,  
    10-14  
Hardware, low-level interface, 1-2 thru  
    1-3, 7-1 thru 7-37  
Hardware handshake, setting, 6-15  
Hardware interrupts, 5-3, 5-4, 5-21  
Hayes compatibility, modem, 12-20 thru  
    12-22  
Heartbeat timer  
    generating interrupts, 7-4  
    invokes interrupt 1Ch, 5-22  
    low-level operation, 7-4 thru 7-5  
    multi-controller registers, 7-16 thru  
        7-19  
    reading and setting, 5-20 thru 5-21  
    restarting, 5-41  
Help, PAM, 10-19 thru 10-20  
Hooks, type of interrupt, 5-1  
HP 110  
    comparison, A-1 thru A-2  
    compatibility, 5-4, 5-31, 6-12, 6-23,  
        F-4  
    connecting via modem, 12-23  
    using desktop link, H-4  
HP 150. *See also* Desktop link  
    accessing its disc drives, F-3  
    comparison, A-2 thru A-3  
    disc compatibility, F-4  
    using desktop link, H-2  
HP 3421A Data Acquisition Unit, 5-66,  
    5-77

**HP 82164A HP-IL/RS-232-C Interface**  
as AUX device, 6-9, 10-12  
as COM2 device, 6-1  
as datacom device, 1-3 thru 1-4  
as plotter, 10-11  
can't turn on and off, 10-15  
not supported by interrupt 14h, 5-13  
setting parameters, 10-12 thru 10-14  
uses ENQ/ACK protocol, 6-14

**HP 82169A HP-IL/HP-IB Interface**  
accessing HP-IB devices, 5-65  
accessing disc drives, F-2  
addressing, 5-68, 5-69  
as plotter, 10-11

**HP 9114A Disc Drive**, accessing, F-1

**HP escape sequences**. *See Escape sequences*

**HP font**. *See Fonts*

**HP mode**  
affects fonts, 5-97  
affects keycodes, 6-47, 13-1  
affects system keys, 5-81 thru 5-84  
character sets, D-1  
CONsole driver mode, 6-45  
default, 5-38, 10-3  
reading status of, 6-52  
selecting, 6-34, 6-50, 10-10

**HP-IB devices**  
addressing, 5-68, 5-69  
controlling, 5-65  
disc drives, F-2 thru F-3

**HP-IL controller (IC)**, 2-14, 7-22 thru 7-25

**HP-IL disc drives**, F-1

**HP-IL interface**  
configuring loop, 5-67, 5-69  
connects defined devices, 6-22  
electrical standard, 2-19  
enabling interrupts, 5-44  
error byte, 5-67  
**IRQ interrupt**. *See Int 4Ch*  
primitives, 5-65 thru 5-80. *See also Int 54h*

reading current status, 5-76  
receiving, 5-74, 5-75  
sending, 5-72, 5-73, 5-75  
service requests, 5-76  
talkers and listeners, 5-72  
timeout, 5-67, 5-76  
turning on and off, 5-51

**HP-IL primitives**. *See Int 54h*

**Humidity specifications**, 2-2

**I**

**IBM computers**. *See also Desktop link*  
accessing their disc drives, F-3  
comparison, A-3 thru A-7  
in alternate mode, 6-23, 6-45  
interrupt compatibility, 1-4, 5-2, 5-19  
softkey compatibility, 6-45  
using desktop link, H-3 thru H-4  
video compatibility, 1-3, 5-5

**IEEE-488 standard**, 5-65

**Int 5h (print screen)**, 5-5, 10-11

**Int 10h (video I/O)**

accessing fonts, 5-23, 5-97  
can conflict with fast alpha, 5-96  
compared to interrupt 5Fh, 5-113  
controls CONsole output, 6-23  
display option, 1-2 thru 1-3  
operation, 5-5 thru 5-11  
used within CON expansion, 5-87

**Int 11h (equipment check)**, 5-12

**Int 12h (memory)**, 5-13

**Int 14h (communications)**

access serial port or modem, 1-4  
operation, 5-13 thru 5-16  
relation to multi-controller registers, 7-13

turns on modem, 6-8, 12-22

turns on serial interface, 6-7

**Int 16h (keyboard I/O)**, 5-17 thru 5-18, 6-23

**Int 17h (print byte)**, 5-19

Int 19h (reboot), 5-20, 5-55  
Int 1Ah (heartbeat), 5-20 thru 5-21  
Int 1Bh (keyboard break), 4-1, 5-21  
Int 1Ch (timer tick), 5-22, 7-4  
Int 1Fh (graphics character extensions),  
    5-5, 5-23, 5-97  
Int 21h (MS-DOS function request)  
    display option, 1-2  
    for serial and modem ports, 6-2  
    not within CON expansion, 5-87  
    number of open files, 11-6  
    sends IOCTL commands, 10-13  
    turns on device, 6-7, 6-8  
Int 40h (modem transmit), 5-24  
Int 42h (modem ring/carrier), 5-24, 5-84,  
    10-17  
Int 43h (timer 2), 5-24  
Int 44h (plug-in 1), 5-25  
Int 45h (plug-in 2), 5-25  
Int 46h (PPU alarm), 5-25, 10-16  
Int 47h (battery cutoff), 5-26  
Int 49h (keyboard), 5-27  
Int 4Ah (serial transmit), 5-28  
Int 4Bh (serial ring/carrier), 5-28, 5-84,  
    10-17  
Int 4Ch (HP-IL IRQ), 5-28, 7-22  
Int 4Dh (low battery), 5-29  
Int 4Eh (modem input), 5-29  
Int 4Fh (serial input), 1-2 thru 1-3, 5-30  
Int 50h (system services)  
    affects power usage, 10-8, 10-9, 10-18  
    causes sleep mode, 11-2  
    changing registers, 7-10, 7-20  
    display option, 1-2 thru 1-3  
    enables interrupt 4Ch, 5-28  
    identifies virtual modules, 2-43  
    operation, 5-31 thru 5-53  
    related to CLOCK device, 6-22  
    summary, 5-31 thru 5-32  
    used within CON expansion, 5-87  
Int 52h (modifier key), 5-64 thru 5-65

Int 53h (print key), 5-65, 6-26, 6-46,  
    10-11  
Int 54h (HP-IL primitives), 5-65 thru  
    5-80, 7-22  
Int 55h (sleep), 5-26, 5-80, 11-2  
Int 56h (menu key), 5-81 thru 5-82, 6-46  
Int 57h (system key), 5-82 thru 5-83,  
    6-46  
Int 58h (break key), 5-83 thru 5-84  
Int 59h (enable/disable ring), 5-84, 6-13  
Int 5Dh (AUX expansion), 1-4, 5-85 thru  
    5-86  
Int 5Eh (CON expansion)  
    can affect fonts, 5-97 thru 5-98  
    controls CONsole output, 6-23  
    operates with simulated input, 5-38  
    operation, 5-87 thru 5-90  
Int 5Fh (fast video)  
    controls CONsole output, 6-23  
    fast alpha operations, 5-94 thru 5-112  
    fast graphics operations, 5-94, 5-113  
        thru 5-119  
    operation, 5-94 thru 5-119  
    used within CON expansion, 5-87  
Interrupts,  
    8086 type, 5-2  
    avoid nesting, 5-1  
    detailed descriptions, 5-1 thru 5-127  
    firmware type, 5-4  
    hardware type, 5-4  
    IBM type, 5-2, A-3 thru A7  
    MS-DOS type, 5-3  
    pseudo-hardware type, 5-3  
    waiting for, 5-41  
Inverse enhancement  
    selecting, 6-33  
    set by attribute byte, 5-6, 5-96  
    operation, 7-33, 7-34  
    reading status of, 6-52  
    selecting, 6-44  
I/O control commands. *See* IOCTL  
    commands

I/O memory. *See* Memory

**IOCTL commands**

- controlling devices, 1-4, 6-9 thru 6-15
- operation, 6-50 thru 6-52
- relation to multi-controller registers, 7-13
- sent by interrupt 21h, 10-13
- setting communications conditions, 6-2

**K**

Key queue. *See also* Keyboard

- affected by transmit functions, 6-46
- affects power-save mode, 10-7 thru 10-8
- bytes from special keys, 5-64, 5-81 thru 5-84
- flushing, 5-21, 5-37, 6-50
- numeric keypad operation, 6-49
- reading from, 5-17, 6-50, 6-51
- receives scancodes, 6-47

Keyboard. *See also* Key queue

- CONsole input device, 6-23
- controlled by multi-controller, 7-3
- debounce, 7-3 thru 7-4, 7-11 thru 7-12
- description, 2-7, 2-10
- enabling interrupts, 5-45
- generates interrupt 49h, 5-27, 7-3
- interrupt registers, 7-9 thru 7-11
- I/O. *See* Int 16h
- keyboard-break interrupt. *See* Int 1Bh
- keycodes, 13-1 thru 13-43
- keys that cause interrupts, 6-46
- language options, 13-1 thru 13-43
- low-level operation, 7-3 thru 7-4
- operation, 6-45 thru 6-49
- matrix key to be processed, 5-89
- multi-controller registers, 7-9 thru 7-12
- numeric keypad, 6-48 thru 6-49
- simulating input, 5-38

specifications, 2-2

status, 5-18, 7-11

Keyboard interrupt. *See* Int 49h

**Keyboard modes**

- affect CONsole driver, 6-23
- affect interrupts, 5-64, 5-81 thru 5-84
- affect simulated input, 5-38
- reading status of, 6-52
- operation, 6-47 thru 6-49
- selecting, 6-35, 6-50, 6-51

Keycodes, 13-1 thru 13-43

Keypad. *See* Numeric keypad

**L**

Languages, each has unique configuration

EPROM, 2-11

LCD. *See* Display (hardware)

Line-drawing character set, 6-23, D-2 thru D-8

**Lines**

- drawing, 5-115
- reading pattern type, 5-115
- setting pattern type, 5-89, 5-116, 6-38

Liquid-crystal display. *See* Display (hardware)

Listeners (HP-IL), 5-72

Logical rectangle

- defined, 5-94
- filling, 5-111
- size and position, 5-100, 5-101
- scrolling, 5-110

Low battery interrupt. *See* Int 4Dh

Low-level hardware interface, 1-2 thru 1-3, 7-1 thru 7-37

LPT1 device, 6-1, 6-22

LPT2 device, 6-1, 6-22

LST device, 6-1, 6-22

**M**

Main memory. *See* Memory; System RAM

Mainframe, dimensions, 2-2, 3-3

Mass storage. *See Disc drives*  
Math character set, 6-23, D-2 thru D-8  
Mechanical design, description, 3-1 thru 3-7  
Media, compatibility, F-4 thru F-5  
Memory. *See also* Edisc; RAM; ROM  
    addressing, 2-3 thru 2-4, 7-1 thru 7-2  
    erased, 4-3, 5-52, 5-53  
    initializing, 5-35  
    multi-controller registers, 7-7 thru 7-8  
    operation, 8-1 thru 8-15  
    specifications, 2-2  
Memory board PCA, 2-7, B-8 thru B-9  
Memory drawers, 8-4, B-14 thru B-19, I-1, I-3. *See also* Plug-in drawers  
Memory lost condition, 4-3 thru 4-4  
Menu interrupt. *See* Int 56h  
Menus (softkey labels). *See* Softkey labels  
Modem. *See also* AUX device; Modem interface  
    as AUX device, 6-9, 10-12  
    autoanswering, 12-4 thru 12-5, 12-15  
    carrier interrupt, 5-45. *See also* Int 42h  
    checking presence, 6-13  
    commands, 12-2 thru 12-3, 12-6 thru 12-12  
    connecting to another modem, 12-23  
    connects to mainframe, 2-8  
    controlled by multi-controller, 7-3  
    controlling lines, 5-58  
    data registers, 7-15  
    default conditions, 12-5 thru 12-6  
    determines baud rate, 12-3 thru 12-4, 12-6  
    determines parity, 12-3  
    device driver operation, 6-7 thru 6-8  
    dialing, 6-12, 12-12 thru 12-13  
    format registers, 7-12 thru 7-15  
    hanging up, 6-12  
    input interrupt. *See* Int 4Eh  
    interrupt registers, 7-15 thru 7-16  
loopback mode, 7-6, 7-13, 12-19  
low-level operation, 7-5 thru 7-6  
modes, 12-1 thru 12-2, 12-5  
multi-controller registers, 7-12 thru 7-16  
operation, 12-1 thru 12-23  
part numbers, I-1, I-3  
possible datacom assignment, 1-3 thru 1-4  
processing incoming data, 5-85 thru 5-86  
ring interrupt, 5-84, 6-13, 10-17  
reading identity of, 12-9  
related to COM3 device, 6-1  
relation to interrupt 14h, 5-13  
resetting, 5-55  
ring interrupt, 5-45. *See also* Int 42h  
S-registers, 12-6, 12-10, 12-11, 12-15 thru 12-20  
schematic, B-10 thru B-11  
setting power estimates, 5-47, 5-57  
setting originate mode, 6-12  
status, 5-86, 12-9  
transmission settings, 10-12 thru 10-14, 12-4  
transmit interrupt. *See* Int 40h  
turning on and off, 5-51, 5-52, 5-61, 5-62, 5-85, 6-10, 10-15  
uses timer 2, 7-5  
verbosity options, 12-5, 12-11  
vocabulary options, 12-3, 12-5, 12-11, 12-13 thru 12-15  
Modem interface, 2-25 thru 2-27, 3-4.  
    *See also* Modem  
Modem PCA, 3-4 thru 3-5, B-10 thru B-11  
Modes, display. *See* Display modes  
Modes, system power, 2-4 thru 2-7. *See also* Awake mode; Sleep mode; Stop mode  
Modifier keys. *See also* Int 52h

affect keyboard registers, 7-9 thru  
7-11  
can invoke interrupt 52h, 5-64  
coding as scancodes, 6-47 thru 6-48  
don't affect interrupt 49h, 5-27  
in modifier mode, 6-50  
low-level operation, 7-3  
reading status, 5-18  
selecting interrupts, 5-45  
simulating, 5-38  
states, 5-81, 5-82, 5-89  
Modifier mode, 6-49. *See also* Keyboard  
modes  
Motherboard PCA, 2-7, B-2 thru B-7  
MS-DOS interrupts, 1-2, 5-3  
Multi-controllers  
    changing interrupt registers, 5-44 thru  
        5-45  
    internal registers, 7-7 thru 7-21  
    multi-purpose port, 7-7, 7-19 thru  
        7-21  
    operation, 7-3 thru 7-21  
    shares serial function, 2-14  
Muted characters, 13-1, 13-42 thru  
    13-43

## N

NUL device, defined, 6-21  
Numeric keypad, 2-11, 5-18, 6-48 thru  
    6-49, 6-50, 6-51

## P

Page (display), 5-8, 5-9  
PAM  
    environment, 10-1 thru 10-3  
    help facility, 10-19 thru 10-20  
    interface to plug-in ROMs, 9-8  
    operation, 10-1 thru 10-21  
    replacing, 10-21  
    startup state, 10-3  
    using applications, 10-4 thru 10-5

PAM.ENV files, 9-8, 10-1 thru 10-2,  
    10-4, 10-7  
Parity, 5-14, 6-13, 10-13  
Parts, I-1 thru I-3  
PATH variable, 9-8, 10-1, 10-2  
Pen (graphics), 5-113, 5-117, 6-39, 6-40  
Personal Application Manager. *See* PAM  
Plotter, specifying, 10-11  
PLT device, 6-1, 6-22, 10-11  
Plug-in drawers. *See also* Memory  
    drawers; Software drawers; Plug-in  
    ports; Virtual modules  
affect battery indicator, 10-18  
architectural requirements, 2-41 thru  
    2-47  
cause interrupts 44h and 45h, 5-25  
checking presence, 5-43  
connect to mainframe, 2-8  
description, 2-28  
electrical requirements, 2-28 thru 2-41  
enabling interrupts, 5-45  
mechanical design, 3-6 thru 3-7  
modes of operation, 2-45 thru 2-47  
PCA dimensions, 3-7  
RAM requirements, 2-44 thru 2-45  
reading IDs, 5-43  
registers, 2-41 thru 2-45  
removal causes stop mode, 2-7, 2-47  
ROM/EPROM requirements, 2-44  
setting power estimates, 5-47, 5-57  
Plug-in ports. *See also* Plug-in drawers  
    accept plug-in drawers, 2-8, 3-6  
    connector description, 2-29 thru 2-33  
    electrical description, 2-28 thru 2-47  
    timing, 2-34 thru 2-38  
Plug-in RAM, 8-2. *See also* Memory  
    drawers  
Plug-in ROMs  
    affect PATH variable, 9-8, 10-2  
    boot sector, 9-2 thru 9-6, 9-7  
    calling, 5-34

can be in ROM disc, 8-1  
can contain boot code, 11-1, 11-2,  
    11-3  
can contain help file, 10-19  
checksums, 9-5  
configurations, K-1 thru K-6  
controlling, 8-3  
enabling and disabling, 5-33  
implementations, 9-5, K-1 thru K-6  
interface to PAM, 9-8  
internal design, 9-1 thru 9-8  
organization in ROM disc, 8-10 thru  
    8-14  
requirements, K-3 thru K-4  
Portable (HP 110). *See* HP 110  
Power, setting estimates, 5-47, 5-57  
Power supplies, 2-4 thru 2-5, 2-10  
Power-save mode, 5-42, 5-43, 10-7 thru  
    10-8, 11-2  
Power-up. *See* Boot  
PPU  
    alarm interrupt, 5-25  
    commands, 5-46, 5-54 thru 5-63  
    controls operating mode, 2-4 thru 2-7  
    description, 2-9  
    enabling interrupt of CPU, 5-59  
    informed of CPU condition, 5-59  
    reading internal status, 5-60, 5-63  
    revision, 4-5, 5-59  
    shares serial function, 2-14  
Print direction, 6-26, 6-32  
Print interrupt. *See* Int 53h  
Print screen interrupt. *See* Int 5h  
Printers  
    configuring, G-1 thru G-14  
    initializing, 5-19  
    operating conditions, 10-11 thru 10-12  
    presence, 5-12  
    reading status, 5-19  
    sending characters to, 5-19  
PRN device, 6-2, 6-22, 10-11

Product number, 2-11, 4-4  
Protocols, for serial interface, 6-2 thru  
    6-7

## R

RAM. *See* Memory; Plug-in RAM;  
    Memory drawers  
RAM disc. *See* Edisc  
RAM drawers. *See* Memory drawers  
Reboot. *See* Boot  
Recharger  
    automotive, 2-22  
    setting current estimates, 5-47, 5-57  
Recharger interface, 2-19 thru 2-23  
Reference documents, 1-4 thru 1-5  
Reset, 4-1 thru 4-5, 5-20  
Reset button, 4-3  
ROM disc  
    accessing, 8-14 thru 8-15  
    alternative drive at boot, 11-4  
    as device driver, 6-1  
    boot sector, 8-9  
    built-in disc drive, F-1  
    derived from ROM, 8-1  
    in plug-in ROMs, 9-1  
    maintaining, 8-3  
    organization, 8-7 thru 8-14  
    root directory, 8-11 thru 8-12  
ROM drawers. *See* Software drawers;  
    Plug-in ROMs  
ROM-executable code, 5-34, 9-1, 9-2,  
    9-3, 9-6  
Roman8 character set, 6-23, 6-45, 10-10,  
    D-2 thru D-8  
ROMs, 4-4, 4-5. *See also* Memory  
RS-232 interface. *See* Serial interface

## S

Scancode mode, 5-17, 6-49. *See also*  
    Keyboard modes  
Scancodes, 5-17, 5-89, 6-47 thru 6-48

Schematic diagrams, B-1 thru B-19  
Secured sleep mode, 5-52, 5-53  
Serial cables, 2-18  
Serial interface. *See also* AUX device  
    as AUX device, 6-9, 10-12  
    as datacom device, 1-3 thru 1-4  
    as plotter, 10-11  
    cable descriptions, 2-18  
    carrier interrupt, 5-44. *See* Int 4Bh  
    connector description, 2-14  
    controlled by multi-controller, 7-3  
    controlling lines, 5-54 thru 5-55  
    data registers, 7-15  
    device driver operation, 6-2 thru 6-7  
    electrical description, 2-13 thru 2-18  
    enabling ring interrupt, 5-84, 6-13  
    format registers, 7-12 thru 7-15  
    input interrupt. *See* Int 4Fh  
    interrupt registers, 7-15 thru 7-16  
    loopback mode, 7-6, 7-13  
    low-level operation, 7-5 thru 7-6  
    multi-controller registers, 7-12 thru  
        7-16  
    processing incoming data, 5-85 thru  
        5-86  
    related to COM1 driver, 6-1  
    relation to interrupt 14h, 5-13  
    ring interrupt, 5-44. *See* Int 4Bh  
    setting communications parameters,  
        10-12 thru 10-14  
    setting current estimates, 5-57  
    status byte, 5-86  
    transmission interrupt. *See* Int 4Ah  
    turning on and off, 5-51, 5-52, 5-54,  
        6-10, 10-15  
    uses HP-IL registers, 7-25  
    XON/XOFF protocol, 6-7  
Serial number, 2-11, 4-5, 6-40  
Service requests (HP-IL), 5-76  
Services, type of interrupt, 5-1  
Shell, 11-6

Sleep mode. *See also* Timeout  
    activating, 5-45  
    affects plug-in drawers, 2-45 thru  
        2-47  
    initiated by interrupt 55h, 5-80  
    low-level keyboard operation, 7-4  
    methods of initiating, 2-5 thru 2-6  
    multi-controller interrupts, 7-3  
    recovering at boot, 11-1, 11-2  
    secured condition, 5-52, 5-53  
    system condition, 2-5, 2-9, 2-10, 7-22,  
        7-26

Slot 7. *See* Software drawers

Softkey labels  
    affect page length, 6-42  
    defined by PAM.MNU files, 10-4  
    DOS commands label, 10-4  
    not part of window, 5-94, 5-96  
    operation, 7-37  
    reading status of, 6-52  
    stored in display RAM, 7-29  
    turning on and off, 5-81, 6-34

Softkeys, 6-33, 6-34, 6-45

Software drawers. *See also* Plug-in  
    drawers  
    configuring, K-1 thru K-9  
    description, K-1 thru K-9  
    part numbers, I-1, I-3  
    reading slot 7 subdirectory, 5-50  
    schematic, B-12 thru B-13

Speaker. *See* Beeper

Specifications, 2-2

Stop bits, 5-14, 6-15, 10-13

Stop mode, 2-5, 2-7, 2-47

Strings. *See* Characters

Subdirectories, 5-50

System configuration, 5-12, 10-6 thru  
    10-12, 10-21, 11-6

System key interrupt. *See* Int 57h

System RAM, 5-13, 5-36, 8-1, 10-6

System services interrupt. *See* Int 50h

## T

Talkers (HP-IL), 5-72  
Temperature specifications, 2-2  
TERM, loading files, E-1 thru E-2  
Time and date  
    affected by time zone, 5-49, 5-50  
    affects PPU internal status, 5-63  
    configuration, 10-15  
    from CLOCK device, 6-22  
    not affected by interrupt 1Ah, 5-20  
    reading and setting, 5-48, 5-61, 10-15  
    updating display, 5-41  
Time zone, 5-49, 5-50, 5-63  
Timeout  
    affects PPU internal status, 5-63  
    enabling and disabling, 5-43  
    initiates sleep mode, 2-6  
    operation, 10-9 thru 10-10  
    reading and setting, 5-42  
Timer 2  
    generating interrupts, 7-4  
    interrupt registers, 7-16 thru 7-17  
    invokes interrupt 43h, 5-24  
    low-level operation, 7-4 thru 7-5  
    multi-controller registers, 7-16 thru  
        7-19  
    timer-tick interrupt. *See* Int 1Ch  
    used by modem, 5-24  
Transmit functions  
    active in modifier mode, 6-49  
    at PAM boot, 10-3  
    available in HP mode, 6-45  
    operation, 6-46  
    reading status of, 6-52  
    turning on and off, 6-35

## U

Underline enhancement  
    in attribute byte, 5-6, 5-96  
    operation, 7-33, 7-34  
    reading status of, 6-52  
    selecting, 6-33, 6-44

V

Video interface, 2-23 thru 2-24  
Video interface accessory, I-1  
Video I/O interrupt. *See* Int 10h  
Virtual modules, 2-42 thru 2-45, 5-43

## W

Window  
    defined, 5-94  
    low-level positioning, 7-36  
    position, 5-114  
    scrolling, 5-109  
    size, 5-99, 5-100, 5-114  
Word length, 5-14, 6-15

## X

XON/XOFF protocol  
    affected by interrupt 5Dh, 5-85  
    defining characters, 6-14  
    enabling, 6-10, 10-14  
    status, 6-13  
    supported for serial interface, 6-7  
    supported for modem, 6-7  
    using, 6-25





Mfg. No. 45559-90001

Printed in U.S.A.