HEWLETT-PACKARD

Technical Reference Manual
Volume 2

HP Vectra Technical Reference Manual
Volume 2: System BIOS

Hewlett-Packard

IBM is a U.S. registered trademark of International
Business Machine Corporation.
Intel is a U.S. registered trademark of Intel Corporation.
Mouse System is a registered trademark of Mouse Systems
Corporation.
Vectra is a registered trademark of Hewlett-Packard Company.
MS-DOS is a registered trademark of Microsoft, Inc.

Notice

The information contained in this document is subject to change without notice.

Hewlett-Packard makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranties 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.

Vectra is a U.S. registered trademark of Hewlett-Packard Company
MS-DOS is a U.S. registered trademark of Microsoft, Incorporated
WordStar is a U.S. registered trademark of MicroPro International Corporation.

© 1985 by Hewlett-Packard Co.
Personal Office Computer Division
974 East Arques Avenue
P.O. Box 486
Sunnyvale, CA 94086, U.S.A.

First Edition - September 1985
Printed in U.S.A.

FCC Statement

Federal Communications Commission Radio Frequency Interference Statement

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

More About Radio and Television Interference

Because the HP Vectra PC generates and uses radio frequency energy, it may cause interference with radio and television reception in a residential installation.

Hewlett-Packard's system certification tests were conducted with HP-supported peripheral devices and HP shielded cables, such as those you receive with your system. The HP Vectra PC meets the requirements for a Class B computing device in accordance with the specifications of Part 15, Subpart J, of protection against interference with radio and television reception in a residential installation.

Hewlett-Packard provides instructions for using this computer in manuals covering setup, connection of peripheral devices, operation, service, and technical reference.

Installing and using the computer in strict accordance with Hewlett-Packard's instructions will minimize the chances that the HP Vectra PC will cause radio or television interference. However, Hewlett-Packard does not guarantee that the computer will not interfere with radio and television reception. If you think your computer is causing interference, turn it off to see if the radio or television reception improves. If the reception:

- Does not improve, your computer is not causing the problem.
- Does improve, your computer is causing the problem.

To correct interference, take one or more of the following steps:

- Relocate the radio or television antenna.
- Move the computer away from the radio or television.
- Plug the computer into a different electrical outlet, so that the computer and the radio or television are on separate electrical circuits.

- Make sure that all of your peripheral devices are certified Class B by the FCC.
- Make sure you use only shielded cables to connect peripheral devices to your computer.
- Consult your computer dealer, Hewlett-Packard, or an experienced radio/television technician for other suggestions.
- Order the FCC booklet called How to Identify and Resolve Radio-TV Interference Problems for the U.S. Government Printing Office, Washington, D.C. 20402.

Warning: Electrical Safety

For the user's safety, the power cords supplied with this product have grounded plugs. The power cords should be used with properly grounded (3-hole) wall outlets to avoid electrical shock. (You can also use multiple-outlet strips that have their own circuit breakers.)

Printed U.S.A.
part number 45961-90001

TABLE OF CONTENTS

Appendices

SECTION 1

TABLE OF CONTENTS

SECTION 1. INTRODUCTION

This manual contains a detailed description of the ROM Basic Input/Output System (BIOS) of the HP Vectra Personal Computer. Entry points, including the industry standard ROM BIOS entry points and function calls, are documented in this manual.

This manual deals extensively with programming and programming concepts. It presumes that the reader is familiar with the Microsoft Macro Assembler (MASM) and the Intel iAPX 80286 processor architecture.

Related documents which may be of interest to programmers and advanced users are listed at the end of this volume in the *References* section.

1.1 System Software

Software operating on the system may be viewed as a three-level hierarchy: application programs, operating system, and ROM BIOS. These three levels are defined as follows:

> Application Programs—An application program is the top level of software. It performs application-specific functions (i.e., spreadsheet or word processing functions). Application programs rely on either DOS or the ROM BIOS for system functions such as character or disc I/O.
>
> Operating System—The operating system provides the control and support functions necessary for an application program to be executed. The operating system provides file-oriented functions, as well as providing basic support for character I/O.
>
> ROM BIOS—The ROM BIOS provides the interface between operating system software and the hardware. The ROM BIOS provides a dual function; it constitutes the low level interface between the hardware and operating system, as well as providing extended functions to application programs.

The higher the software level, the more powerful the functions provided by the software. However, along with this power often comes additional overhead which reduces performance and flexibility. A system programmer should choose the level of software interface required by the individual set of design constraints. It is good programming practice to use the highest level of system software that gets the job done. Some system functions can only be performed on the highest level, since only system software supports the function. However, other system functions may be performed at more than one level. Using a lower level such as the ROM BIOS provides improved speed of execution and additional flexibility. Using ROM BIOS routines may affect program portability to future HP products, and to other industry standard PC's.

1.2 ROM BIOS

The ROM BIOS provides a powerful set of system functions, allowing application programs full access to the capabilities of the system while maintaining a hardware-independent interface.

The ROM BIOS allows the programmer or system designer to tailor the system to a specific set of design constraints. Some of the tailoring methods provided to the programmer are:

- The number of interrupts can logically expand to fit requirements.
- Adapter cards can obtain a limited amount of RAM from the system BIOS without installing device drivers.
- Applications can expand the features of the keyboard without replacing the industry standard driver (INT 16H).
- The ROM resident mouse driver system can provide the ability to use various input peripherals with applications not specifically written for them.

These methods maintain application compatibility with minimal effect on system performance.

SECTION 2

TABLE OF CONTENTS

SECTION 2. ROM BIOS OVERVIEW

The ROM BIOS is divided into two components, the Standard BIOS (STD-BIOS) and the Extended BIOS (EX-BIOS). The STD-BIOS supports the industry standard set of BIOS functions. The EX-BIOS is unique to the HP Vectra. It provides a wide range of system functions and support for HP peripherals. The STD-BIOS and EX-BIOS are discussed later in this section. Both the STD-BIOS and the EX-BIOS are contained in the system ROM which resides at the top of system memory.

Note

> Throughout the remainder of this manual the terms BIOS, STD-BIOS, and EX-BIOS will be used. STD-BIOS and EX-BIOS are defined above. The term ROM BIOS will be used to indicate the union of STD-BIOS and EX-BIOS.

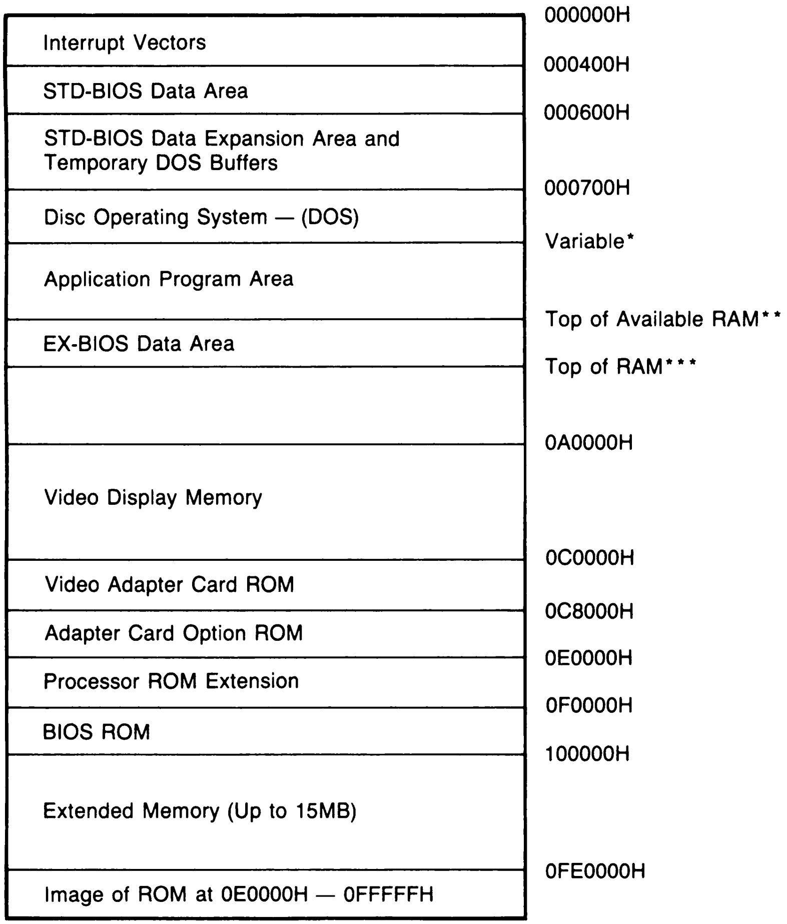
This section contains an overview of the components of the ROM BIOS. These components are the interrupt vectors, code modules, and data structures. Interrupt vectors form the link between the operating system, applications, and the ROM BIOS. The code modules perform the ROM BIOS functions. Data structures provide the means for the ROM BIOS (and to some extent the applications) to maintain driver variables, data buffers, etc.

2.1 Memory Locations

Code modules are accessed through interrupt vectors. The interrupt vectors reside in the first 1KB of system RAM. Usually a code module has an associated data structure. The data structures for the STD-BIOS code modules reside in system RAM in absolute memory locations 00400H through 005FFH. The data structures for the EX-BIOS code module reside at the top of system RAM. *The address of the EX-BIOS data area will vary depending on the particular configuration of the system.*

Figure 2.1 shows the components of the ROM BIOS and their location within the system memory. Each of the ROM BIOS components is discussed in detail in the remainder of this section.

Memory Map Block Diagram



*The length of the operating system is revision dependent.

**The Top of Available RAM is dependent on system configuration, in a 256KB system it is usually 03F000H while in 640KB system it is usually 09F000H.

***The Top of RAM is dependent on system configuration, in a 256KB system it is 03FFFFH while in 640KB system it is 09FFFFH.

Figure 2.1

2.2 Interrupts

The interface to the BIOS is through the interrupt structure of the 80286. The system allows for three types of interrupts.

- Processor Interrupts—These interrupts allow system software to recover from error conditions and other hardware exceptions.

- Hardware Interrupts—These interrupts are generated by the 8259A interrupt controllers on the processor board. Hardware interrupts indicate that a system hardware component or peripheral requires service.

- Software Interrupts—These interrupts are generated through the software 'INT n' instruction. Software interrupts allow system functions to be quickly and easily called by any program.

Interrupt vectors for the processor interrupts are defined by the 80286. Interrupt vectors for the hardware interrupts are mapped by the values programmed into the 8259A interrupt controllers which are initialized by the ROM BIOS. Processor and/or hardware interrupts may be 'simulated' by a software interrupt mapped to the same interrupt vector. For example, Interrupt 0 is mapped by the 80286 for Divide by 0 error. The service routine for this error condition may be executed by an INT 0 instruction.

Each interrupt has an interrupt vector associated with it. The interrupt vector contains the Code Segment and Instruction Pointer of the service routine for that interrupt. Each of these vectors consists of two words (four bytes). The iAPX 80286 architecture supports 256 interrupt vectors which occupy the first 1024 bytes (00000H-003FFH) of system memory.

The interrupt vectors maintain industry standard compatibility while offering the expanded capabilities of the HP EX-BIOS functions. Table 2.1 lists these assignments.

In order for the system to function properly, processor and hardware interrupt vectors are initialized to valid service routines. Most unused vectors point to a null routine in the BIOS which issues an End-of-Interrupt (EOI) signal to the 8259A(s) when required and returns. The Keyboard Break and Timer Tick software interrupt vectors point to an IRET instruction in the BIOS. These vectors are indicated by an IRET in table 2.1. Several software vectors are used as pointers to data blocks instead of interrupt service routines. These vectors are indicated by a PT in table 2.1.

Table 2.1

Interrupt Vector Assignments

Address	Int	Function		Type*	Service	Routine**
000-003H	0	Divide by Zero		PI	STD-BIOS	(UI)
004-007H	1	Single Step		PI	STD-BIOS	(UI)
008-00BH	2	Nonmaskable Interrupt		PI	STD-BIOS	
00C-00FH	3	Breakpoint		PI	STD-BIOS	(UI)
010-013H	4	Arithmetic Overflow		PI	STD-BIOS	(UI)
014-017H	5	Print Screen		SW	STD-BIOS	(DRVR)
018-01BH	6	Invalid Opcode		PI	STD-BIOS	(UI)
01C-01FH	7	Reserved		PI	STD-BIOS	(UI)
020-023H	8	Timer Interrupt	(IRQ 0)	HW	STD-BIOS	
024-027H	9	Keyboard ISR	(IRQ 1)	HW	STD-BIOS	
028-02BH	A	Reserved	(IRQ 2)	HW	STD-BIOS	
02C-02FH	B	Serial Port 1 ISR	(IRQ 3)	HW	STD-BIOS	(UI)
030-033H	C	Serial Port 0 ISR	(IRQ 4)	HW	STD-BIOS	(UI)
034-037H	D	Printer Port 1 ISR	(IRQ 5)	HW	STD-BIOS	(UI)
038-03BH	E	Diskette ISR	(IRQ 6)	HW	STD-BIOS	
03C-03FH	F	Printer Port 0 ISR	(IRQ 7)	HW	STD-BIOD	(UI)
040-043H	10	Video		SW	STD-BIOS	(DRVR)
044-047H	11	Equipment Check		SW	STD-BIOS	(DRVR)
048-04BH	12	Memory Size		SW	STD-BIOS	(DRVR)
04C-04FH	13	Diskette/Hard Disc		SW	STD-BIOS	(DRVR)
050-053H	14	Serial		SW	STD-BIOS	(DRVR)
054-057H	15	System Functions		SW	STD-BIOS	(DRVR)
058-05BH	16	Keyboard		SW	STD-BIOS	(DRVR)
05C-05FH	17	Printer		SW	STD-BIOS	(DRVR)
060-063H	18	Reserved		SW	N/A	(IRET)
064-067H	19	Boot		SW	STD-BIOS	(DRVR)
068-06BH	1A	Time and Date		SW	STD-BIOS	(DRVR)
06C-06FH	1B	Keyboard Break		SW	STD-BIOS	(IRET)
070-073H	1C	Timer Tick		SW	STD-BIOS	(IRET)
074-077H	1D	Video Parameter Table		PT	STD-BIOS	
078-07BH	1E	Diskette Parameter Table		PT	STD-BIOS	
07C-07FH	1F	Graphics Character Table		PT	STD-BIOS	
080-083H	20	Program Terminate		SW	DOS	
084-087H	21	DOS Function Calls		SW	DOS	
088-08BH	22	DOS Terminate Address		PT	DOS	

Address	Int	Function	Type*	Service	Routine**
08C-08FH	23	DOS <CTRL>-<Break> Address	SW	DOS	
090-093H	24	DOS Critical Error	SW	DOS	
094-097H	25	DOS Absolute Disc Read	SW	DOS	
098-09BH	26	DOS Absolute Disc Write	SW	DOS	
09C-09FH	27	DOS Terminate Stay Resident	SW	DOS	
0A0-0CBH	28-32	Reserved for DOS	SW	DOS	
0CC-0CFH	33	HP Mouse	SW	EX-BIOS	(DRVR)
0D0-0FFH	34-3F	Reserved for DOS	SW	DOS	
100-103H	40	Alternate Diskette	SW	STD-BIOS	
104-107H	41	Hard Disc Parameter Table (0)	PT	STD-BIOS	
108-117H	42-45	Reserved	SW	STD-BIOS	
118-11BH	46	Hard Disc Parameter Table (1)	PT	STD-BIOS	
11C-17FH	47-5F	Reserved	SW	STD-BIOS	
180-19FH	60-67	Reserved for User Programs	SW	N/A	
1A0-1A3H	68	8041 Service Request ISR	HW	EX-BIOS	
1A4-1A7H	69	Keyboard OBF ISR	HW	EX-BIOS	
1A8-1ABH	6A	Reserved	HW	EX-BIOS	
1AC-1AFH	6B	Reserved	HW	EX-BIOS	
1B0-1B3H	6C	HP-HIL Controller ISR	HW	EX-BIOS	
1B4-1B7H	6D	Reserved	HW	EX-BIOS	
1B8-1BBH	6E	Reserved	HW	EX-BIOS	
1BC-1BFH	6F	EX-BIOS Entry Point	SW	EX-BIOS	(DRVR)
1C0-1C3H	70	Real-time Clock ISR (IRQ 8)	HW	STD-BIOS	
1C4-1C7H	71	SW Redirected (IRQ 9)	HW	STD-BIOS	
1C8-1CBH	72	Reserved (IRQ 10)	HW	STD-BIOS	(UI)
1CC-1CFH	73	Reserved (IRQ 11)	HW	STD-BIOS	(UI)
1D0-1D3H	74	Reserved (IRQ 12)	HW	STD-BIOS	(UI)
1D4-1D7H	75	Coprocessor (IRQ 13)	HW	STD-BIOS	
1D8-1DBH	76	Hard Disc ISR (IRQ 14)	HW	STD-BIOS	(UI)
1DC-1DFH	77	Reserved (IRQ 15)	HW	STD-BIOS	(UI)
1E0-1FFH	78-7F	Not Used	SW	N/A	
200-3C3H	80-F0	Reserved	SW	N/A	
3C4-3FFH	F1-FF	Not Used	SW	N/A	

* PI—Processor interrupt
HW—Hardware interrupt
SW—Software interrupt
PT—Interrupt vector used as pointer to data.
N/A—Not applicable

** UI—Unused interrupt ISR
IRET—Interrupt returned
DRVR—Application callable entry point

2.3 ROM BIOS, Drivers and Functions

The ROM BIOS is comprised of many drivers. For example, there is a driver to perform video functions, one to perform disc functions, etc. The ROM BIOS drivers are organized into two components. One component contains the STD-BIOS drivers that support the STD-BIOS functions. The second component contains EX-BIOS drivers that support unique HP features.

Each driver supports one or more functions. A function can be viewed as a specific task. For example, the Video Driver supports 22 separate functions that perform tasks such as setting the display mode, moving the cursor, and displaying characters.

2.3.1 STD-BIOS Drivers

Drivers in the STD-BIOS are accessed through an interrupt. STD-BIOS drivers are accessed through interrupts 05H and 10H through 1CH. Drivers are accessed by performing a software INT n instruction, where n is the interrupt number assigned to the driver (refer to table 2.1.)

The function code and any required data are passed in the 80286 registers. Data passing conventions for STD-BIOS drivers vary, however, there are aspects which are common.

- Most of the STD-BIOS drivers support more than one function. Therefore, multi-function drivers must have the desired function code passed as part of the data. The AH register is used on all multi-function drivers to pass the function code.

- Byte and word data is passed in the internal registers of the 80286. Registers AL, BX, CX, and DX are usually used for this purpose. The register assignments and number of registers used depend on the driver and driver function.

- If the amount of data cannot fit in the internal registers of the 80286, a data buffer in system memory is used. This buffer is usually pointed to by ES:BX, ES:BP or ES:SI.

- Drivers may modify one or more registers. The registers which are maintained and the registers which are modified vary from driver to driver. The registers which are modified are listed in each function description.

Calling STD-BIOS Drivers

The following program example demonstrates accessing a typical STD-BIOS driver. The function sets the position of the cursor on display page 0 to row 20, column 10. The function code (02H) is passed in register AH. The row position, the column position, and the page number are passed respectively in DH, DL, and BH.

```
MOV     AH,02H          ;Function number
MOV     DH,14H          ;Row number (Row 20)
MOV     DL,0AH          ;Column number (Column 10)
MOV     BH,0H           ;Page number
INT     10H             ;Call Video driver
```

The STD-BIOS drivers support all industry standard BIOS functions. In addition, many of the drivers have additional functions that support enhanced features. These functions are referred to as 'HP extensions' throughout the remainder of this manual. These enhancements are accessed through function code (06FH) of their respective driver. Most of these extended functions are further divided into subfunctions. For example, the HP extended function for the Video driver has six subfunctions which allow access to the enhanced features of the Multimode Video Display Adapter. The function code (06FH) is placed in the AH register and the subfunction code in AL for all HP extensions.

The following example uses HP extensions to turn off the cursor control pad on the keyboard.

```
MOV     AH,6FH          ; HP Function
MOV     AL,07H          ; Switch Keyboard
MOV     BL,02H          ; Disable CCP: Turn Cursor Control
                        ; Pad Off
INT     16H             ; Call Keyboard Driver
```

2.3.2 EX-BIOS Drivers

The EX-BIOS drivers provide a wide range of functions not found in the STD-BIOS drivers. The EX-BIOS drivers are accessed through a single software interrupt vector. This interrupt (06FH) will be referred to as INT HP__ENTRY. Due to the large number of EX-BIOS drivers, it would be impossible to give each driver its own interrupt vector and still maintain industry standard compatibility. Therefore, each driver is assigned its own number which is placed in the BP register. This manual refers to these numbers by the names assigned in Appendix E.

Calling EX-BIOS Drivers

As with the STD-BIOS drivers, each EX-BIOS driver may support one or more functions. A function code placed in the AH register selects the desired function within the driver. In addition, a subfunction code passed in the AL register is required by many EX-BIOS functions.

The following program example demonstrates access to a typical EX-BIOS driver. The function executes a 'beep' on the speaker.

```
MOV     AH,3AH        ; Function: F__SND__BEEP
MOV     BP,12H        ; Driver Name: V__SYSTEM
PUSH    DS            ;
INT     6FH           ; EX-BIOS Call: HP__ENTRY
POP     DS            ;
```

On leaving the EX-BIOS driver the BP and DS registers will be modified while the AH register usually contains the return status of the driver call.

2.3.3 EX-BIOS Standard Functions

Many EX-BIOS drivers support a standard set of functions and subfunctions as listed in table 2.2. While these functions and subfunctions are defined, it is not required that they all be implemented by every driver. In addition, EX-BIOS drivers may implement functions other than those listed. Most EX-BIOS drivers use a standard set of return status codes reported in the AH register at the completion of a driver's function call. Some of these return status codes and their definitions are listed in table 2.3. A driver may report a return status code of RS__UNSUPPORTED (02H) for a given function.

Function codes and return statuses are described in detail in Appendix G.

Table 2.2

EX-BIOS Defined Functions

Function	Register		
Subfunction	AH	AL	Definition
F_ISR	00		Responds to a logical Interrupt Service Request (ISR).
F_SYSTEM			Executes one of several standard subfunctions.
SF_INIT	02	00	Starts the initialization of a driver.
SF_START	02	02	Completes the initialization process of the driver.
SF_REPORT_STATE	02	04	Reports the state of the driver.
SF_VERSION_DESC	02	06	Reports the revision number and datecode of the driver.
SF_DEF_ATTR	02	08	Reports the default configuration of the driver.
SF_GET_ATTR	02	0A	Reports the current configuration of the driver.
SF_SET_ATTR	02	0C	Overrides the current configuration of the driver.
SF_OPEN	02	0E	Reserves the driver for exclusive access. Requests any resources required by the driver.
SF_CLOSE	02	10	Releases the driver from exclusive access.
SF_TIMEOUT	02	12	Reports to the driver that a requested timeout has occurred.
SF_INTERVAL	02	14	Reports to the driver that a requested 60 Hz interval has expired.
SF_TEST	02	16	Performs a hardware test.
F_IO_CONTROL			Executes the following subfunctions and any driver dependant subfunctions.
SF_LOCK	04	00	Reserves the sub-address device specified for exclusive access.
SF_UNLOCK	04	02	Releases the sub-address specified from the exclusive access.
F_PUT_BYTE	06		Writes a byte of data.
F_GET_BYTE	08		Reads a byte of data.
F_PUT_BUFFER	0A		Writes a variable length buffer of data (supported by character devices).
F_PUT_BLOCK	0A		Writes a fixed length buffer of data (supported by block devices).
F_GET_BUFFER	0C		Reads a variable length buffer of data (supported by character devices).
F_GET_BLOCK	0C		Reads a fixed length block of data (supported by block devices).
F_PUT_WORD	0E		Writes a word of data.
F_GET_WORD	10		Reads a word of data.

2.3.4 EX-BIOS Parameter Passing Conventions

When calling EX-BIOS drivers, the function code is placed in the AH register, and the subfunction code (if any) in the AL register. Note that the function and subfunction codes are multiples of two in order to facilitate decoding by the drivers.

The general parameter passing conventions used by the EX-BIOS drivers are also defined. These register conventions are as follows:

On Entry:	BP	=	V__DRIVER__NAME
	AH	=	F__FUNC__CODE
	AL	=	SF__FUNC__CODE (if required by driver)
	CX	=	On write: byte count (if required by driver) On read: maximum permissible byte count (if required by driver)
	ES:DI	=	Buffer pointer or context area (if required by driver)
On Exit:	AH	=	Return status
	CX	=	On read: byte count (if required by driver) On write: number of bytes written (if required by driver)
	ES:DI	=	Buffer pointer or context area (if required by driver)
	DS,BP		Always modified (unless otherwise indicated)

2.3.5 EX-BIOS Return Status Codes

EX-BIOS drivers are expected to report a Return Status Code upon completion. This code is returned in the AH register. Several status codes have been defined and are listed in table 2.3.

Table 2.3

EX-BIOS Return Status Codes

Return Status	Code	Indication
RS_SUCCESSFUL	000H	The requested function executed correctly.
RS_UNSUPPORTED	002H	The requested function or subfunction is not implemented or is unsupported.
RS_FAIL	0FEH (-02H)	The driver failed the operation in an error state.
RS_BAD_PARAMETER	0FAH (-06H)	The driver received a bad parameter.
RS_BUSY	0F8H (-08H)	The requested driver is busy.
RS_NO_VECTOR	0F6H (-0AH)	EX-BIOS Vector table is out of RAM or room for more drivers.
RS_OFFLINE	0F4H (-0CH)	Device is offline.
RS_OUT_OF_PAPER	0F2H (-0EH)	Device is out of paper.

If additional drivers are installed in the system, they should conform to the defined statuses wherever possible. However, to maintain coding efficiency and/or functional accuracy, a driver may create a return status other than those listed in Table 2.3.

Note

Return status conditions are always multiples of two. Negative return status codes indicate error conditions, while positive status codes indicate exceptional conditions to the caller. For example, the status code RS_UNSUPPORTED indicates the driver does not support a function which may or may not be an error, while RS_OUT_OF_PAPER requires some kind of response by the caller.

2.4 Data Structures

BIOS drivers require RAM data area to perform their functions. The layout and placement of the data areas for the STD-BIOS and EX-BIOS drivers differ. This is discussed in the following subsections.

2.4.1 STD-BIOS Data Structures

The data area for the STD-BIOS is in absolute memory locations 00400H through 005FFH, which conforms to the industry standard. Table 2.4 summarizes the assignments within this block of memory. Refer to Appendix B for a detailed description of these data fields.

Table 2.4

STD-BIOS Data Area Summary

Address	Function
400H-407H	RS-232 Communications Port Addresses
408H-40FH	Parallel Printer Port Addresses
410H-416H	System Data and Flags
417H-43DH	Keyboard Data Area
43EH-448H	Flexible Disc Data Area
449H-466H	Video Display Data Area
467H-46BH	System Data and Flags
46CH-470H	Timer Data Area
471H-473H	System Data Flags
474H-477H	Hard Disc Data Area
478H-47BH	Printer Timeout Counters
47CH-47FH	RS-232 Communications Port Timeout Counters
480H-483H	Keyboard Data Area
48BH-496H	Diskette/Hard Disc Data Area
498H-504H	System Data and Flags
505H-5FFH	Reserved

2.4.2 EX-BIOS Data Structures

Data structures for the EX-BIOS drivers are located in a block of memory at the top of system RAM. The address of this block varies depending on the amount of RAM contained in the system and the hardware configuration.

There are three types of data structures in the EX-BIOS data area. These structures are: the HPbrVECTOR__TABLE and its associated HP__ENTRY__CODE, the driver data areas, and the EX-BIOS global data area.

HP__VECTOR__TABLE

Each of the 80286 interrupt vectors contains the Code Segment (CS) and Instruction Pointer (IP) of its associated service routine. The HP__ENTRY interrupt vector (06FH) contains the CS:IP of the HP__ENTRY__CODE. This routine uses the value contained in the BP register (an offset into the HP__VECTOR__TABLE, vector address) to branch to the appropriate EX-BIOS driver. The HP__VECTOR__TABLE resides at the base of the EX-BIOS data area. The HP__VECTOR__TABLE consists of an array of 3-word (six bytes) entries, one for each EX-BIOS driver. Each entry consists of the IP, CS, and Data Segment (DS) of a driver.

Figure 2.2 illustrates the relationship between the 80286 interrupt vectors, the HP__VECTOR__TABLE, HP__ENTRY__CODE, and the EX-BIOS drivers.

HP__ENTRY__CODE

The CS:IP in the HP__ENTRY interrupt vector points to a piece of code which branches to the desired EX-BIOS driver. The vector address passed in BP must be a multiple of six. The code is as follows:

```
HP__ENTRY__CODE:
                MOV     DS,CS:[BP+4]
                JMP     FAR PTR CS:[BP]
```

This code resides directly after the last entry in the HP__VECTOR__TABLE. Therefore, the CS:IP entry in the HP__ENTRY interrupt vector provides two further pieces of information. CS:0 is the starting address of the HP__VECTOR__TABLE and IP is the length of the HP__VECTOR__TABLE.

Interrupt Vectors and HP__VECTOR__TABLE

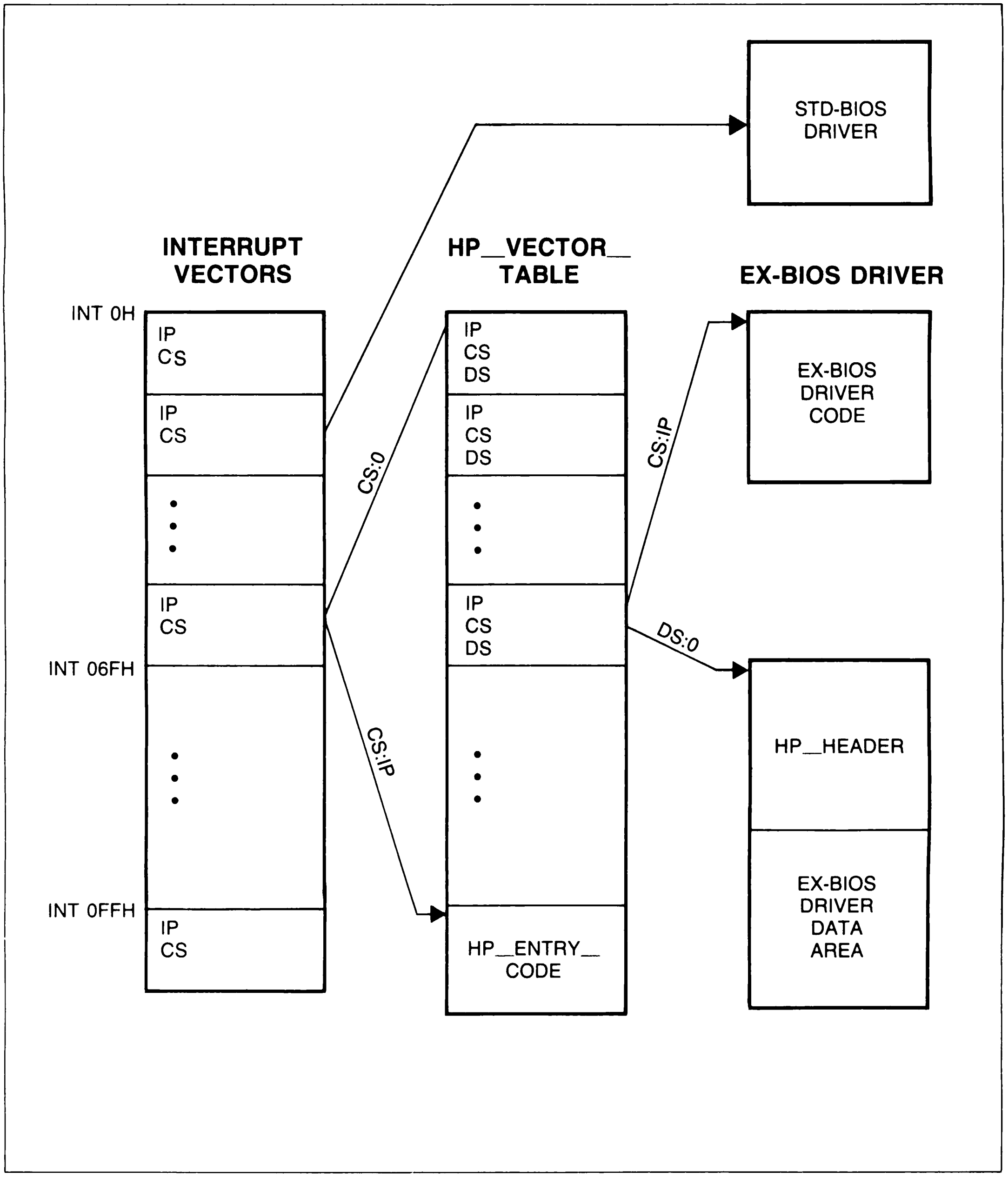


Figure 2.2

Driver Data Areas

Each driver has an independently specified data area. Some EX-BIOS drivers share the same data areas. The data areas for the EX-BIOS drivers are above the HP__VECTOR__TABLE and the HP__ENTRY__CODE shown in figure 2.2. Although each driver has its own data area, the DS for each driver is stored in the HP__VECTOR__TABLE, and its data area must start at DS:0. Each data area must reside on a paragraph boundary.

The data area for each driver consists of a driver header, followed by an optional variable storage area. The variable storage area is unique to each driver. Table 2.5 provides a general description of the contents of an EX-BIOS driver header.

Each driver's header and/or variable storage area is described in a following section.

Table 2.5

HP__DRIVER__HEADER

Variable	Offset	Type	Offset Definition
DH__ATR	0	Word	Driver Attribute Field
DH__NAME__INDEX	2	Word	Driver String Index Field
DH__V__DEFAULT	4	Word	Driver's Default Logical Device Vector
DH__P__CLASS	6	Word	Driver's Parent Class
DH__C__CLASS	8	Word	Driver's Child Class
DH__V__PARENT	0AH	Word	Driver's Parent Vector
DH__V__CHILD	0CH	Word	Driver's Child Vector
DH__MAJOR	0EH	Byte	Sub Address Field
DH__MINOR	0FH	Byte	Sub Address Field

EX-BIOS Driver Headers

The definition of each of these fields is listed in the following. Additional information on these fields can be found in Appendix G.

DH__ATR: Each bit in the DH__ATR field indicates a property of the driver for device mapping purposes. These bits are defined in Appendix G.

DH_NAME_INDEX: The DH_NAME_INDEX is used to derive the localization string index of the driver. This is given by the function F_STR_GET_STRING in the V_SYSTEM driver. See Section 9 for additional information.

DH_V_DEFAULT: The DH_V_DEFAULT field contains the driver's default vector address.

DH_P_CLASS and DH_C_CLASS: In conjunction, these fields indicate which drivers may be mapped together. DH_P_CLASS and DH_C_CLASS are bit masks. Each bit position represents a set of drivers. If a bit is set then the driver is in that set of drivers. The DH_P_CLASS field indicates a driver is in from 0 to 16 different driver sets. A driver can only map to another driver if its DH_P_CLASS field matches at least one bit position of the other driver's DH_C_CLASS field. Furthermore, the DH_ATR field is another condition of mapping. The bits are defined in Appendix G.

DH_V_PARENT: The DH_V_PARENT field contains a vector to the driver that is called when the current driver receives an F_ISR function code that it cannot or doesn't know how to process.

DH_V_CHILD: The DH_V_CHILD field contains a vector to the driver that is called if this driver decides it cannot handle the request function (as long as that function is not F_ISR).

DH_MAJOR and DH_MINOR: Device bus address information.

EX-BIOS Global Data Area

The method for locating the EX-BIOS global data area is found in the ''EX-BIOS Data Area Map'' of Appendix B. The EX-BIOS global data area is shared between several EX-BIOS drivers. It contains temporary and permanent variables that are required by the BIOS to function properly. Some of these variables can be modified by application programs. As with any modification to the STD-BIOS data area, care should be taken with the EX-BIOS global data area. Table 2.6 defines the contents of this area.

Table 2.6

Global Data Area

Byte	Name	Type	Definition
00-013H	Reserved		
14	T_SND_FLAG	Byte	Sound Driver Status
	Bit **Definition** 7 '1' Click enabled 6 '1' Beep enabled 5-0 Reserved		
15	T_SND_CLICK_COUNT	Byte	Contains number of pending key clicks. Maximum of four.
16	T_SND_CLICK_DURA	Byte	Contains current tick duration scaler.
17	T_SND_CLICK_VOLUME	Byte	Contains current key click volume.
18	T_SND_BEEP_CYCLE	Word	Contains current beep period in ten microsecond increments.
1A	T_SND_BEEP_DURA	Word	Contains current duration of the beep in 10 microsecond increments.
1C	T_SND_BEEP_COUNT	Byte	Contains number of pending beep functions. Maximum of four.
1D	Reserved	Byte	
1E	T_STR_NEXT_INDEX	Word	Next unused string index number.
20 and up	Reserved		

SECTION 3

TABLE OF CONTENTS

SECTION 3. VIDEO

The HP MultiMode Video Display Adapter provides a wide variety of display modes, resolution, character attributes, and other features. The purpose of the video driver is to allow programs to access these features and control the video display.

3.1 Overview

In the text mode, the MultiMode Video Display Adapter uses an 8 × 16 character cell which generates high quality characters. Access to the display memory is fully synchronized to eliminate the ''snow'' problem present in many color display adapters. (Snow occurs when writing a character to display memory while the video memory is being accessed by the display refresh circuitry.) This full synchronization makes the INT 10H video driver faster, since there is no need to wait for a vertical retrace to place characters on the screen.

The MultiMode Video Display Adapter provides seven more display modes than the industry standard color graphics adapter. Four of the modes allow 27 lines of text on the screen. The other three modes allow graphics modes that double the graphics resolution of the display (320 × 400 and 640 × 400 pixels). The standard INT 10H video driver has been extended to allow the programmer to set these modes. No other support is provided to make use of these modes. Refer to *HP Vectra Technical Reference Manual Volume I: Hardware* for more information on the MultiMode Video Display Adapter.

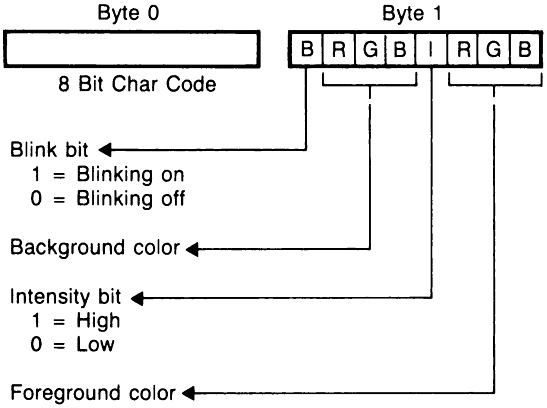
3.2 Data Structures

The MultiMode Video Display Adapter has 32KB of video memory starting at address 0B8000H. This allows graphics resolutions of 320 × 400 in medium resolution modes and 640 × 400 in high resolution modes. The following is a discussion of how this memory is organized depending on the video mode selected.

In either of the text modes (80 × 25 or 40 × 25) memory is organized as sequential pages. Each page contains character cells that are made up of an 8 bit character code and an 8 bit attribute (see Figure 3.1).

Text Display Memory Organization

Character Cell Organization



Color Values

I	R	G	B	Color
0	0	0	0	Black
0	0	0	1	Blue
0	0	1	0	Green
0	0	1	1	Cyan
0	1	0	0	Red
0	1	0	1	Magenta
0	1	1	0	Brown
0	1	1	1	Light Grey
1	0	0	0	Dark Grey
1	0	0	1	Light Blue
1	0	1	0	Light Green
1	0	1	1	Light Cyan
1	1	0	0	Light Red
1	1	0	1	Light Magenta
1	1	1	0	Yellow
1	1	1	1	White

80 x 25 Text Memory Page

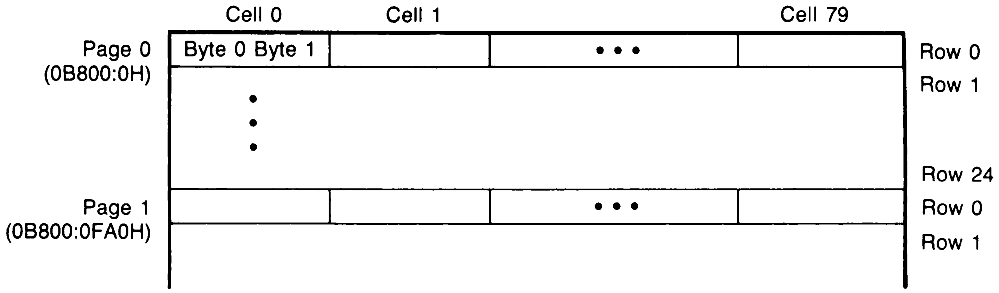
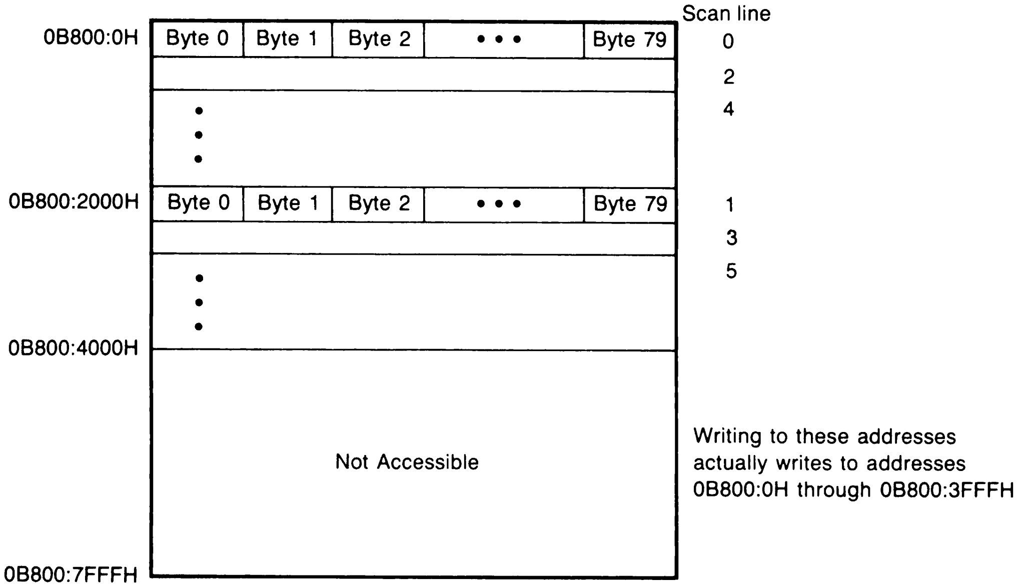


Figure 3.1

Graphics modes can be of two types: medium resolution (320 × 200 or 320 × 400) and high resolution (640 × 200 or 640 × 400). In the medium resolution mode each pixel corresponds to two bits of memory so four colors can be displayed. In the high resolution modes each pixel corresponds to one bit of memory and only one color can be displayed (the background color is always black). See Figures 3.2 and 3.3 for more details.

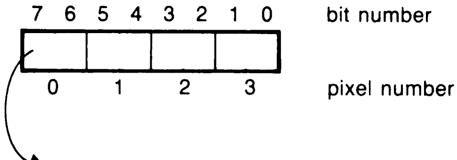
Graphics Display Memory Organization

320 x 200 Graphics Display Memory



Writing to these addresses actually writes to addresses 0B800:0H through 0B800:3FFFH

Byte / Pixel Organization



0 0 - 1 of 16 Background Colors
0 1 - Green/Cyan
1 0 - Red/Magenta
1 1 - Brown/Light Grey

Figure 3.2

Graphics Display Memory Organization

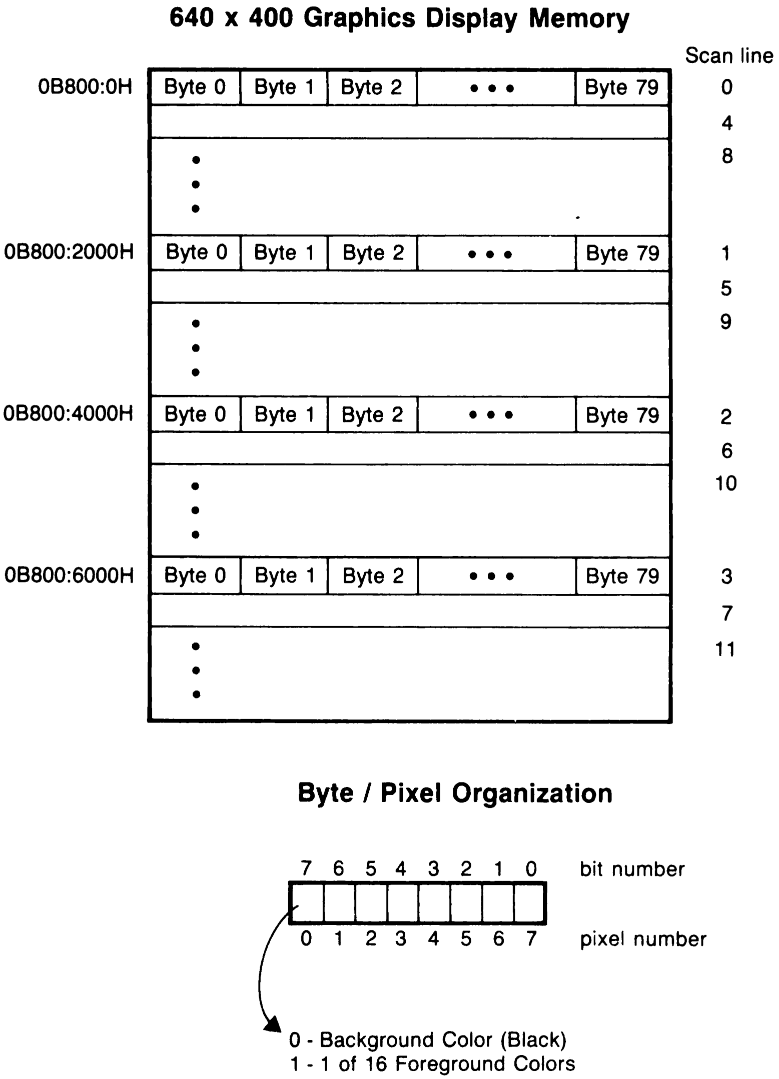


Figure 3.3

In all the graphics modes, the memory used for scan lines is not sequential but it is interleaved at fixed intervals of 8K. In the modes that are 200 scan lines, even scan lines start at offset 0 and odd scan lines start at offset 2000H. In the modes that are 400 scan lines, the following table can be used to determine the appropriate offset:

Scan line is multiple of 4 (0,4,8,12 ...) use offset 0
Scan line is multiple of 4 plus 1 (1,5,9,13 ...) use offset 2000H
Scan line is multiple of 4 plus 2 (2,6,10,14...) use offset 4000H
Scan line is multiple of 4 plus 3 (3,7,11,15...) use offset 6000H

All the scan lines of a particular group are organized sequentially within a particular offset. See Figures 3.2 and 3.3.

Other video driver data structures are located in the STD-BIOS data area. They are stored in memory addresses 449H (40H:49H) through 466H (40H:66H). Table 3.1 lists the memory locations and their definitions.

Table 3.1

STD-BIOS Video Driver Data Area

Address	Type	Definition
00449H	Byte	Current Video Display Mode
0044AH	Word	Number of columns
0044CH	Word	Regen buffer length
0044EH	Word	Starting address of regen buffer
00450H	Word	Cursor position for Display Page 0
00452H	Word	Cursor position for Display Page 1
00454H	Word	Cursor position for Display Page 2
00456H	Word	Cursor position for Display Page 3
00458H	Word	Cursor position for Display Page 4
0045AH	Word	Cursor position for Display Page 5
0045CH	Word	Cursor position for Display Page 6
0045EH	Word	Cursor position for Display Page 7
00460H	Word	Current cursor mode
00462H	Byte	Active page number
00463H	Word	Address of current display adapter
00465H	Byte	Mode (current setting of status register)
00466H	Byte	Pallet setting

Video data structures are also maintained in the EX-BIOS data area. These structures are accessible through the data segment of the EX-BIOS video service routine. The following code sets the ES register to the EX-BIOS video driver's (V__SVIDEO'S) data segment:

```
MOV AX,0                    ;segment at 0
MOV ES,AX                   ;
MOV AX,ES: [6FH*4 + 2]      ;read the base address
                            ;of the HP__VECTOR__TABLE
MOV ES,AX
MOV AX,ES: [V__SVIDEO + 4]  ;read base address of
MOV ES,AX                   ;video parameters
```

The addresses listed are offsets into this data segment. The following table gives the data maintained in V__SVIDEO's (0054H) data segment:

Table 3.2

Video EX-BIOS Data Structures

Variable Name	Offset	Type	Definition
Driver Header	0-5	Byte	Device Header Attributes, Name, Index, and Default Vector
VID__PRIMARY	6	Byte	The current primary display: 00 Card at I/O Address 3B0H 01 Card at I/O Address 3C0H 02 Card at I/O Address 3D0H 03 Card containing ROM Code.
VID__SECONDARY	7	Byte	If two cards are in the system, same number as VID__PRIMARY for the second card.
VID__FOUND__ROM	8	Byte	Flag set to true if ROM code was found in any video adapter card.
VID__IDS	9-0CH	Byte	List of IDs of all cards found.
VID__STATUS	0D-010H	Byte	RAM copies of the status register.
VID__EXT__STATUS	11-014H	Byte	RAM copies of the extended status register for each possible card in the system.
VID__PARM__BLOCK	15-03BH	Byte	Reserved for saving the video parameters stored in the standard BIOS data area when switching between primary and secondary video boards.
VID__LAST__IBM__MODE	03CH	Byte	Used to detect if a 'rogue' program changed the modes without telling the HP system.
VID__EXT__MODE	03DH	Byte	Specifies the current video mode (0 . . . 15).
	3E-03FH	Byte	Reserved

3.3 Video Driver (INT 10H)

The video driver functions can be broken down into the following categories.

- Display Control—These functions control the display appearance, cursor and light pen position, active text memory page, and scrolling through text memory.
- Character Handling Functions—These functions manipulate characters on the screen.
- String Functions—These functions allow placement of strings of text on the screen.
- Graphics Functions—These functions provide a minimal interface to the graphics capabilities of the machine.
- Extended Video Functions—These functions support extra video capabilities of the MultiMode Video Display Adapter hardware.

Table 3.3 summarizes the functions performed by the video driver. A detailed description of the functions is given following the table.

Table 3.3

Video Driver Function Code Summary

INT Hex	Function/ Equate	Function Value	Definition
10H	INT_VIDEO		Video
	F10_SET_MODE	00H	Set video mode
	F10_SET_CURSIZE	01H	Set cursor size
	F10_SET_CURPOS	02H	Set cursor position
	F10_RD_CURPOS	03H	Read cursor position
	F10_RD_PENPOS	04H	Read light-pen position
	F10_SET_PAGE	05H	Set active display page
	F10_SCROLL_UP	06H	Scroll rectangle up
	F10_SCROLL_DN	07H	Scroll rectangle down
	F10_RD_CHARATR	08H	Read character and attribute at cursor position
	F10_WR_CHARATR	09H	Write character and attribute at cursor position
	F10_WR_CHARCUR	0AH	Write character at cursor position
	F10_SET_PALLET	0BH	Set color pallet
	F10_WR_PIXEL	0CH	Write pixel
	F10_RD_PIXEL	0DH	Read pixel
	F10_WR_CHARTEL	0EH	Write teletype character
	F10_GET_STMODE	0FH	Get video state and mode
		10H-12H	Reserved
	Write string functions:		
	F10_WRS_00	1300H	global attribute
	F10_WRS_01	1301H	global attribute, move cursor
	F10_WRS_02	1302H	individual attributes
	F10_WRS_03	1303H	individual attributes, move cursor
	F10_INQUIRE	6F00H	EX-BIOS present
	F10_GET_INFO	6F01H	Get video parameters
	F10_SET_INFO	6F02H	Sets video parameter
	F10_MOD_INFO	6F03H	Modifies video parameters
	F10_GET_RES	6F04H	Reports video resolution
	F10_XSET_MODE	6F05H	Sets video resolution

Video Driver Function Definitions

The following function definitions provide a detailed description of each of the functions in the video driver.

F10__SET__MODE (AH = 00H)

This function sets the display mode of the video adapter. The new mode is determined by the value passed in the AL register. Refer to the *Vectra Technical Reference Manual, Volume I* for additional information on the various video display modes available on the MultiMode Video Display Adapter.

On Entry: AH = F10__SET__MODE (00H)
AL = Mode

Data	Definition
00	40 × 25 Black and White Alphanumeric
01	40 × 25 Color Alphanumeric
02	80 × 25 Black and White Alphanumeric
03	80 × 25 Color Alphanumeric
04	320 × 200 Color Graphics
05	320 × 200 Black and White Graphics
06	640 × 200 Black and White Graphics
07	Only valid if a monochrome display adapter is present.

On Exit: No values returned

Registers Altered: AX

F10__SET__CURSIZE (AH = 01H)

This function sets the size of the cursor displayed in the alphanumeric display modes. Each character cell in the alphanumeric display modes is eight scan lines high. The cursor size is defined by specifying the starting and ending scan lines within the character cell. The scan lines are numbered from 0 (top of cell) to 7 (bottom). The starting and ending scan lines are passed in registers CH and CL. This function performs no operation if the MultiMode Video Display Adapter is in one of the graphics modes.

On Entry: AH = F10__SET__CURSIZE (01H)
CH = Starting scan line
CL = Ending scan line

On Exit: No values returned.

Registers Altered: AH

F10__SET__CURPOS (AH = 02H)

This function sets the row and column address of the cursor to the specified page, and moves the cursor to that address. When the MultiMode Video Display Adapter is in one of the graphics modes, a page number of 0 must be specified.

On Entry: AH = F10__SET__CURPOS (02H)
BH = Display page number
DH = Row address of cursor. (0. . .24)
DL = Column address of cursor. (0. . .79)

On Exit: No values returned.

Registers Altered: None

F10__RD__CURPOS (AH = 03H)

This function returns the current address and size of the cursor on the specified page. If the MultiMode Video Display Adapter is in one of the graphics modes, a page number of 0 must be specified. The values returned for the cursor size in the graphics mode will be invalid.

On Entry: AH = F10__RD__CURPOS (03H)
BH = Display page number

On Exit: CH = Starting scan line
CL = Ending scan line
DH = Row address of cursor. (0. . .24)
DL = Column address of cursor. (0. . .79)

Registers Altered: CX, DX

F10__RD__PENPOS (AH = 04H)

This function returns the current state and position of the light pen if it is activated. The position is reported in both character row/column and graphic pixel formats.

On Entry: AH = F10__RD__PENPOS (04H)

On Exit: AH = Light Pen state

Data	Definition
0	Not activated
1	Activated

BX = Horizontal pixel position of light pen
CH = Vertical pixel position of light pen (200 line mode)
DH = Row position of light pen
DL = Column position of light pen

Registers Altered: AH, BX, CH, DX

F10__SET__PAGE (AH = 05H)

This function sets the active display page in the alphanumeric mode. Valid page numbers are 0 through 7 for 80 × 25 modes, and 0 through 7 for 40 × 25 modes. This function is not valid for graphics modes.

On Entry: AH = F10__SET__PAGE (05H)
AL = Page number (0 through 7)

On Exit: No values returned.

Registers Altered: AX

F10__SCROLL__UP (AH = 06H)

This function scrolls the contents of a window up a specified number of lines. The window is defined by the row and column addresses stored in the CX and DX registers. The number of lines to be scrolled is passed in register AL. If AL is set to 0, the function interprets this as a command to scroll all lines.

On Entry: AH = F10__SCROLL__UP (06H)
AL = Number of lines to scroll (0 = scroll all)
BH = Attribute to place in blanked lines
CH = Row address of upper left corner of window (0. . .24)
CL = Column address of upper left corner of window (0. . .79)
DH = Row address of lower right corner of window (0. . .24)
DL = Column address of lower right corner of window (0. . .79)

On Exit: No values returned.

Registers Altered: None

F10__SCROLL__DN (AH = 07H)

This function scrolls the contents of a window down a specified number of lines. The window is defined by the row and column addresses stored in the CX and DX registers. The number of lines to be scrolled is passed in register AL. If AL is set to 0, the function interprets this as a command to scroll all lines. This function is only valid when the MultiMode Video Display Adapter is in one of the alphanumeric modes.

On Entry: AH = F10__SCROLL__DN (07H)
AL = Number of lines to scroll (0 = scroll all)
BH = Attribute to place in blanked lines
CH = Row address of upper left corner of window (0. . .24)
CL = Column address of upper left corner of window (0. . .79)
DH = Row address of lower right corner of window (0. . .24)
DL = Column address of lower right corner of window (0. . .79)

On Exit: No values returned.

Registers Altered: None

F10__RD__CHARATR (AH = 08H)

This function returns the character byte and attribute byte at the current cursor location. If the MultiMode Video Display Adapter is in one of the alphanumeric modes, a page number must be specified. If the video display adapter is in one of the graphics modes, only the character is returned, since characters do not have attribute bytes in the graphics modes.

On Entry: AH = F10__RD__CHARATR (08H)
BH = Page number (alphanumeric modes only)

On Exit: AH = Attribute byte (valid only in alphanumeric modes)
AL = Character

Registers Altered: AX

F10__WR__CHARATR (AH = 09H)

This function writes character and attribute bytes at the current cursor location. If the MultiMode Video Display Adapter is in one of the alphanumeric modes, a page number may be specified. If the MultiMode Video Display Adapter is in one of the graphics modes, only the character is written. More than one character and attribute can be stored by placing the number of copies desired in CX. This function will wrap around both line and screen if too many characters are specified. Note that this function makes copies of a single character/attribute combination, it does not print a string. Refer to the Write String function for that operation.

On Entry: AH = F10__WR__CHARATR (09H)
AL = Character
BH = Page number (alphanumeric modes only)
BL = Attribute byte (valid only in alphanumeric modes)
CX = Number of characters to write

On Exit: No values returned.

Registers Altered: None

F10__WR__CHARCUR (AH = 0AH)

This function writes a character to the current cursor location, retaining the existing attribute byte. The function is identical to the F10__WR__CHARATR function, except that no attribute byte is written.

On Entry: AH = F10__WR__CHARCUR (0AH)
AL = Character
BH = Page number (alphanumeric modes only)
CX = Number of characters to write

On Exit: No values returned.

Registers Altered: None

F10__SET__PALLET (AH = 0BH)

This function allows setting the background color (if BH = 0) or the foreground color pallet (if BH = 1).

On Entry: AH = F10__SET__PALLET (0BH)
BH = Color Select ID

Data	Definition
0	Set the background color (in medium resolution modes) or the foreground color (in high resolution modes) based on the low bits of BL (bits 0. . .3) to one of 16 colors.
1	Select color pallet (for medium resolution modes) based on the least significant bit of BL. If bit 0 of BL = '0' then select the green, red, brown pallet. If bit of BL = '1' then select the cyan, magenta, light grey pallet.

BL = Color select value

On Exit: No values returned

Registers Altered: None

F10__WR__PIXEL (AH = 0CH)

This function writes a pixel on the screen. If the MultiMode Video Display Adapter is in one of the "Four color" modes (320 × 200) the color of the pixel may be passed in register AL. Bits 0 and 1 of AL are interpreted as the color bits. If bit 7 of AL is set, bits 0 and 1 are 'XOR'ed with the current pixel color bits, otherwise they replace the current pixel color bits. If the MultiMode Video Display Adapter is in the "Two color" mode (640 × 200), the bit corresponding to the desired pixel is set.

On Entry: AH = F10__WR__PIXEL (0CH)
AL = Color

In "Four color" mode (320x200):

Bit	Data	Definition
7	1	Bits 0 and 1 XORed with current pixel.
	0	Bits 0 and 1 replace current pixel.
0,1		Color bits.

In "Two color" mode (640 x 200):

Bit	Data	Definition
7	1	Bit 0 XORed with current pixel.
	0	Bit 0 replaces current pixel.
0		Color bit.

CX = Horizontal pixel address
DX = Vertical pixel address

On Exit: No values returned.

Registers Altered: AX

F10__RD__PIXEL (AH = 0DH)

This function returns the color code of the specified pixel.

On Entry: AH = F10__RD__PIXEL (0DH)
CX = Horizontal pixel address
DX = Vertical pixel address

On Exit: AL = Color value of pixel

Registers Altered: AX, CX, DX

F10__WR__CHARTEL (AH = 0EH)

This function writes a character to the active page, then advances the cursor one location. At the end of a line, the cursor will wrap to the next line; at the end of the screen, the cursor will scroll. In the alphanumeric modes, this function maintains the current video display attributes. In the graphics modes, the foreground color is passed in register BL. The ASCII characters Line Feed (0AH), Carriage Return (0DH), Backspace (08H), and Bell (07H) are interpreted by this function as ASCII commands and are executed as such.

On Entry: AH = F10__WR__CHARTEL (0EH)
AL = Character
BL = Foreground color (in graphics modes only)

On Exit: No values returned.

Registers Altered: AX

F10__GET__STMODE (AH = 0FH)

This function returns the current MultiMode Video Display Adapter state. The mode, number of characters per line, and current display page are returned.

On Entry: AH = F10__GET__STMODE (0FH)

On Exit: AH = Number of characters per line
AL = Current mode
BH = Current display page

Registers Altered: AX, BH

Write String (AH = 13H)

This function writes a string of characters to the screen. This function consists of four separate subfunctions which control whether each character has its own attribute byte or not, and whether the cursor is moved or not. Each of the subfunctions is detailed in the following. The ASCII characters Line Feed (0AH), Carriage Return (0DH), Backspace (08H), and Bell (07H) are interpreted by this function as ASCII commands and are executed as such.

F10__WRS__00 (AX = 1300H)

Write string attribute without moving cursor.

On Entry: AX = F10__WRS__00 (1300H)
BH = Display page number
BL = String attribute byte
CX = Length of string
DH = Row address of first character
DL = Column address of first character
ES:BP = Pointer to start of string
Format of string is:
Char, Char, . . ., Char

On Exit: No values returned.

Registers Altered: None

F10__WRS__01 (AX = 1301H)

Write string attribute and move cursor.

On Entry: AX = F10__WRS__01 (1301H)
BH = Display page number
BL = String attribute byte
CX = Length of string
DH = Row address of first character
DL = Column address of first character
ES:BP = Pointer to start of string
Format of string is:
Char, Char, . . ., Char

On Exit: No values returned.

Registers Altered: None

F10__WRS__02 (AX = 1302H)

Write character attribute without moving cursor.

On Entry: AX = F10__WRS__02 (1302H)
BH = Display page number
BL = String attribute byte
CX = Length of string
DH = Row address of first character
DL = Column address of first character
ES:BP = Pointer to start of string
Format of string is:
Char, Attr, Char, Attr, . . ., Char, Attr

On Exit: No values returned.

Registers Altered: None

F10__WRS__03 (AX = 1303H)

Write character attribute and move cursor.

On Entry: AX = F10__WRS__03 (1303H)
BH = Display page number
CX = Length of string
DH = Row address of first character
DL = Column address of first character
ES:BP = Pointer to start of string
Format of string is:
Char, Attr, Char, Attr, . . ., Char, Attr

On Exit: No values returned.

Registers Altered: None

3.4 HP Video Extension Functions

This set of functions support the features of the MultiMode Video Display Adapter which are not covered using the standard video functions. This function consists of separate subfunctions which support the various extended capabilities of the MultiMode Video Display Adapter. Each of these subfunctions is defined in the following subsections.

F10__INQUIRE (AX = 6F00H)

This subfunction determines whether or not the extended HP functions are available. If the extended video functions are available, the BX register will be set to 4850H (which is the ASCII characters 'HP').

On Entry: AX = F10__INQUIRE (6F00H)
BX = Any value except 4850H ('HP')

On Exit: BX = 'HP' (4850H)

Registers Altered: AX, BX

F10__GET__INFO (AX = 6F01H)

This function returns information about the primary display adapter.

On Entry: AX = F10__GET__INFO (6F01H)

On Exit: AH = Status register information

Bit	Data	Definition
0	1	Display Enabled.
1	1	Light Pen Trigger Set.
2	1	Light Pen Switch Made.
3	1	Vertical Synchronization
4		Monitor Resolution
	0	350 or 400 line monitor
	1	200 line monitor
5		Display type
	0	Color
	1	Monochrome
6-7		Diagnostic Bits

AL = Card Identifier

Data	Definition
00H	Non HP card with ROM and possibly its own INT 10H driver.
41H	MultiMode Video Display Adapter
42H	Reserved
43H	Reserved
44H	Reserved
45H	Industry Standard Monochrome Display Adapter
46H	Industry Standard Color Display Adapter
51H	Reserved

CL = Current value of Extended Control register. This register is only valid when the Card Identifier is 41H.

This description applies to data returned when a MultiMode Video Display Adapter is in the system.

Bit	Data	Definition
0		Current screen resolution
	0	200 line
	1	400 line
1		Underline enable.
	0	'Blue' bit of foreground attribute interpreted as color blue.
	1	'Blue' bit of foreground attribute interpreted as underline.
2		Font Selected.
	0	Standard-8
	1	HP-ROMAN-8
3		Memory disable.
	0	Memory enabled for CPU access.
	1	Memory disabled for CPU access.
4		16/32K Memory select.
	0	Wrap second 16K of RAM into first 16K.
	1	Allow access to full 32K of memory.
5		Page select.
	0	Select first 16K of memory.
	1	Select second 16K of memory.
6-7		Unused

Registers Altered: AX, CL

F10_SET_INFO (AX = 6F02H)

This function modifies the value of the Extended Control register port 3DDH on the MultiMode Video Display Adapter. (Refer to the *Vectra Technical Reference Manual, Volume I* for more information about this port.)

On Entry: AX = F10_SET_INFO (6F02H)
BL = Byte of data to be written to the Extended Control Register.

Bit	Data	Definition
0		Current screen resolution
	0	200 line
	1	400 line
1		Underline enable.
	0	'Blue' bit of foreground attribute interpreted as color blue.
	1	'Blue' bit of foreground attribute interpreted as underline.
2		Font Selected.
	0	Standard-8
	1	HP-ROMAN-8
3		Memory disable.
	0	Memory enabled for CPU access.
	1	Memory disabled for CPU access.
4		16/32K Memory select.
	0	Wrap second 16K of RAM into first 16K.
	1	Allow access to full 32K of memory.
5		Page select.
	0	Select first 16K of memory.
	1	Select second 16K of memory.
6-7		Reserved

On Exit: No values returned.

Registers Altered: AX, BL

F10_MOD_INFO (AX = 6F03H)

This function modifies individual bits in the Extension Control register (port 3DDH) of the MultiMode Video Display Adapter. A mask byte is passed in register BH, which allows individual bits to be modified without changing the state of other mode bits in the register.

On Entry: AX = F10__MOD__INFO (6F03H)
BH = Mask. Bits with a mask value of '1' are not modified; bits with a mask value of '0' are modified.
BL = Bits to change. The bits indicated by the mask (BH) take the value of the BL register.

Bit	Data	Definition
0		Current screen resolution
	0	200 line
	1	400 line
1		Underline enable.
	0	'Blue' bit of foreground attribute interpreted as color blue.
	1	'Blue' bit of foreground attribute interpreted as underline.
2		Font Selected.
	0	Standard-8
	1	HP-ROMAN-8
3		Memory disable.
	0	Memory enabled for CPU access.
	1	Memory disabled for CPU access.
4		16/32K Memory select.
	0	Wrap second 16K of RAM into first 16K.
	1	Allow access to full 32K of memory.
5		Page select.
	0	Select first 16K of memory.
	1	Select second 16K of memory.
6-7		Reserved

On Exit: No values returned.

Registers Altered: AX

Example:

```
MOV  AX,F10__MOD__INFO    ; EX-BIOS Function Modify
                          ; Ex-Reg (6F03H)
MOV  BL,00000100B         ; Select Character Font: HP-ROMAN-8
MOV  BH,11111011B         ; Only Modify Character Font
INT  10H                  ; Call Video Interrupt
```

F10__GET__RES (AX = 6F04H)

This function returns the current video mode and screen resolution.

On Entry: AX = F10__GET__RES (6F04H)

On Exit: AL = Current video mode (See Set Mode.)

Data	Definition
00H	40 × 25 Alphanumeric Black and White
01H	40 × 25 Alphanumeric Color
02H	80 × 25 Alphanumeric Black and White
03H	80 × 25 Alphanumeric Color
04H	320 × 200 Graphics Color
05H	320 × 200 Graphics Black and White
06H	640 × 200 Graphics Black and White
07H	80 × 25 Only Valid for Monochrome Cards
08H	80 × 27 Alphanumeric Black and White
09H	80 × 27 Alphanumeric Color
0AH	40 × 27 Alphanumeric Black and White
0BH	40 × 27 Alphanumeric Color
0CH	Reserved
0DH	640 × 400 Graphics Black and White
0EH	320 × 400 Graphics Color
0FH	320 × 400 Graphics Black and White

If in one of the graphics modes:

BX = Horizontal resolution in pixels
CX = Vertical resolution in pixels

If in one of the text modes:

BX = Number of characters per row
CX = Number of rows

Registers Altered: AX, BX, CX

F10__XSET__MODE (AX = 6F05H)

This function places the MultiMode Video Display Adapter in one of sixteen possible modes of operation. Modes 0 through 7 are identical to the modes available with function F10__SET__MODE of the video driver. Modes 8 through 15 are unique to the HP Vectra and its MultiMode Video Display Adapter, and may only be set using this function.

Programmers must exercise caution when setting video modes with both F10_SET_MODE (0H) and F10_XSET_MODE (6F05H). Whenever F10_XSET_MODE is used to select one of the "HP only" modes (8–15), F10_XSET_MODE (not F10_SET_MODE) must be used to return to one of the industry standard modes (0–7). This "pairing" of function calls is necessary because F10_XSET_MODE modifies an I/O port not normally affected by the industry standard modes. F10_SET_MODE does not deal with this I/O port.

On Entry: AX = F10_XSET_MODE (6F05H)
BL = Video mode

Data	Definition
00H	40 × 25 Alphanumeric Black and White
01H	40 × 25 Alphanumeric Color
02H	80 × 25 Alphanumeric Black and White
03H	80 × 25 Alphanumeric Color
04H	320 × 200 Graphics Color
05H	320 × 200 Graphics Black and White
06H	640 × 200 Graphics Black and White
07H	80 × 25 Only Valid for Monochrome Cards
08H	80 × 27 Alphanumeric Black and White
09H	80 × 27 Alphanumeric Color
0AH	40 × 27 Alphanumeric Black and White
0BH	40 × 27 Alphanumeric Color
0CH	Reserved
0DH	640 × 400 Graphics Black and White
0EH	320 × 400 Graphics Color
0FH	320 × 400 Graphics Black and White

On Exit: No values returned.

Altered Registers: AX, BL

Example:

```
MOV  AX,F10_XSET_MODE       ; Call EX-BIOS function
                            ; Set mode (6F05H)
MOV  BL,0DH                 ; Select 640 × 400 line mode
INT  INT_VIDEO              ; Call video interrupt (10H)
```

SECTION 4

Table of Contents

SECTION 4. INPUT SYSTEM AND HP-HIL

The Input System is a set of drivers which support the HP-HIL input devices. Up to seven HP-HIL input devices may be connected at one time. The Input System can support properly integrated non-HP-HIL devices as well. In its basic configuration, the system has one input device, the keyboard.

4.1 Overview

The standard devices that connect to the system via the HP-HIL link are the keyboard, mouse, touch screen and tablet. The application interface for the keyboard is described in Section 5. The industry standard interface for the mouse (INT 33H functions) is provided in Section 6. The interfaces for simple mouse, touch screen and tablet support are described in this section.

The architecture of the Input System is divided into two levels (see figure 4.1). The application interface level allows the programmer to communicate with the HP-HIL devices with minimum overhead. The second level, the hardware interface level, allows programmers to manipulate the internals of the system. With this interface, support for additional devices can be added or the data path of existing ones re-directed.

The first portion of this section provides an overview of the application interface level, a detailed description of the actual interfaces and how to access them. The second portion of this section describes the hardware interface level.

4.2 Application Interface Level

Application programs interface with the Input System through a set of logical device drivers. The Input System has an application interface for keyboard, tablet, pointer (simple mouse), and touch screen input devices. These drivers are shown in figure 4.1.

Input System Block Diagram

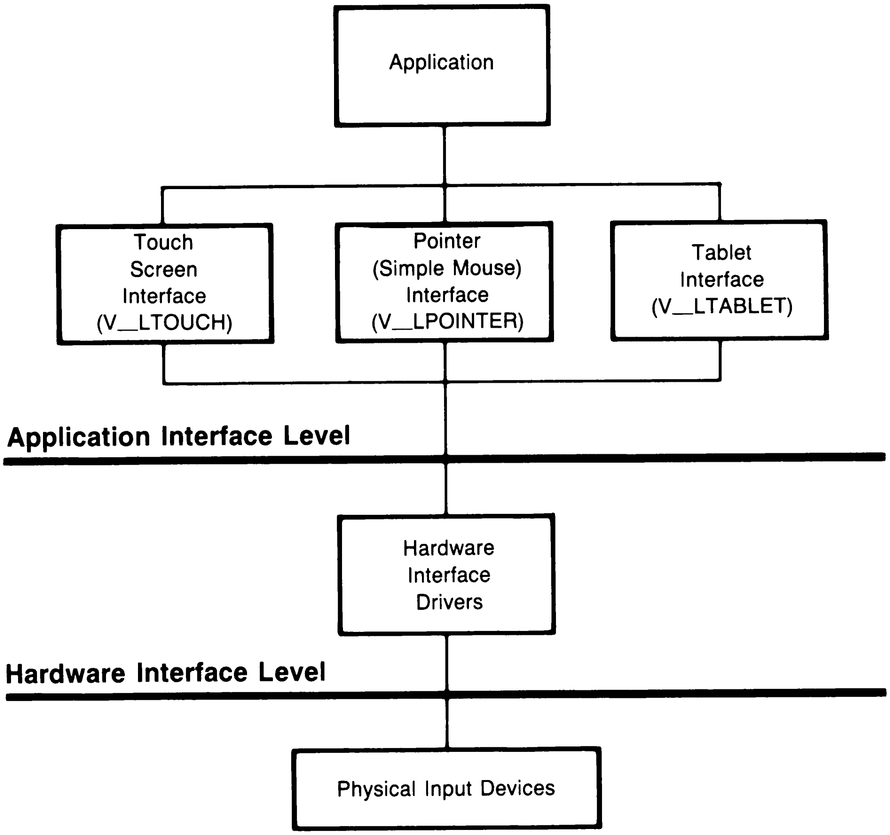


Figure 4.1

The tablet, pointer, and touch screen application program interface drivers are grouped together in figure 4.1 as they are all Graphic Input Device (GID) drivers. GID drivers accept relative graphic motion data, absolute graphics data, and button scancode data from the input devices. Data from these devices is represented in a consistent manner throughout the Input System, making programmatic access to different Graphic Input Devices a simple task (see the Application Event Driver Example later in this section).

4.2.1 Overview

The Input System supports three logical GID drivers; one for each of the standard GID data types. There is a GID driver for each of the touch screen, pointer (simple mouse), and tablet devices called V__LTOUCH, V__LPOINTER, and V__LTABLET respectively. Each of these drivers has a fixed location in the HP__VECTOR__TABLE. They all share a common code module (i.e., they have the same CS:IP in the table), but have different data areas.

The GID drivers perform clipping and scaling under certain conditions. Absolute devices like the touch screen and tablet are always scaled but clipping is user selectable. Relative devices like the mouse can have both scaling and clipping selected by the user.

The logical GID drivers perform two additional tasks. The first is graphics cursor movement (sprite tracking). This is performed by the EX-BIOS driver V__STRACK, which is called by the logical GID driver if tracking is enabled. The second task is to provide interrupt service to the application. The application may install a routine to be called by the logical GID driver every time a GID event occurs, as opposed to the application calling the GID driver repeatedly (polling) to see if an event has occurred.

The following text outlines the actions that occur for touch screen input; from touching the screen to application data retrieval.

1. The user touches the screen. This causes the physical device to generate input data and interrupt the hardware interface level.

2. The hardware interface level processes the interrupt and passes the data (ISR Event Record) to the logical touch screen driver (V__LTOUCH).

3. V__LTOUCH scales the event to fit the current dimensions of the screen. At this point two optional things may happen. First, the data may be clipped. Second, the user defined event driver will be called if it is installed and enabled.

4. If the user event routine was not installed and enabled then the application must call (poll) V__LTOUCH with the F__SAMPLE function (see subsection on V__LTOUCH functions) to get the input data.

There are two methods for applications to receive data from the Input System: polled mode and interrupt mode. In polled mode, the application must continually interrogate the logical GID driver using the F__SAMPLE function to determine if any input has occurred, In interrupt mode, the application must first install an ISR event handling routine (application event driver) using SF__CREATE__EVENT to handle interrupt calls from the logical GID driver. After installation, the application informs the logical GID driver that it is ready to receive interrupts by calling the SF__EVENT__ON subfunction. After event interrupts have been enabled, the application will receive an interrupt every time the logical GID driver receives data from the hardware interface level.

4.2.2 Data Structures

The application interface level uses two major data structures: the Logical Describe Record and the Logical ISR Event Record(s). These data structures help keep track of the numerous events occurring in the Input System.

4.2.2.1 Logical Describe Record

The Logical Describe Record is used by the logical GID drivers to keep track of the current state of their respective devices. Each of the logical GID drivers has a Logical Describe Record associated with it, which is located directly after the driver header starting with memory address DS:0010H. An explanation of the Logical Describe Record fields follows, see table 4.1 for field types and offsets.

Table 4.1

Logical GID Driver Describe Record

Field	Type	Offset	Description
Driver Header		00H	Driver Header (see Section 2)
LD_SOURCE	BYTE	10H	Device GID type
LD_HPHIL_ID	BYTE	11H	Physical device ID
LD_DEVICE_STATE	WORD	12H	Status bits for the logical device
LD_INDEX	BYTE	14H	Physical device vector number
LD_MAX_AXIS	BYTE	15H	Maximum number of axes reported
LD_CLASS	BYLE	16H	Device class
LD_PROMPTS	BYTE	17H	Number of button/prompts
LD_RESERVED	BYTE	18H-1BH	Reserved
LD_TRANSITION	BYTE	1CH	Button transitions
LD_STATE	BYTE	1DH	Current state of the buttons
LD_RESOLUTION	WORD	1EH	Logical device resolution
LD_SIZE_X	WORD	20H	Maximum x-axis count
LD_SIXE_Y	WORD	22H	Maximum y-axis count
LD_ABS_X	WORD	24H	X position data for absolute devices
LD_ABS_Y	WORD	26H	Y position data for absolute devices
LD_REL_X	WORD	28H	X delta for relative devices
LD_REL_Y	WORD	2AH	Y delta for relative devices
LD_ACCUM_X	WORD	2CH	X-axis scaling accumulator
LD_ACCUM_Y	WORD	2EH	Y-axis scaling accumulator

LD_SOURCE
This field is divided into nibbles. Bits 7-4 contain the graphics input device type. This field is loaded with the low order nibble of the appropriate logical GID data type (table 4.5). Bits 3-0 are reserved.

LD_HPHIL_ID
ID byte of the physical device which last reported data. See table 4.2 for a list of HP-HIL ID bytes.

LD_DEVICE_STATE
Status bits for the logical device

Bit	Definition
0FH–05H	Reserved.
04H	Event enabled when set.
03H	Tracking enabled when set.
02H	Clipping enabled when set.
01H	Button error occurred when set.
00H	Interrupt in progress when set.

LD_INDEX
This contains the vector address divided by 6 of the last physical device that reported data.

LD_MAX_AXIS
Maximum number of axes supported by the device. Valid range is 0-2.

LD_CLASS
Device class. Bits 7-4 contain the current class. Bits 3-0 contain the default class. See Appendix G for more information on device classes.

LD_PROMPTS
Number of buttons and prompts supported by the device. Bits 7-4 contain the number of prompts. Bits 3-0 contain the number of buttons.

LD_TRANSITION
Transitions reported per button, i.e., a set bit indicates that the corresponding button was either pushed or released. Bit 7 corresponds to button 7 etc.

LD_STATE
Current state of the buttons. 1 is down, 0 is up. Bit 7 corresponds to button 7 etc. If LD_STATE is XOR'ed with LD_TRANSITION the result is the previous button state.

LD_RESOLUTION
This is the resolution of the logical device. For logical devices this is typically one.

Table 4.2

HP-HIL Device ID Bytes

Device Type	ID Range	Device Description
Keyboard	00H-02H	Reserved
	03H	Swiss-French Keyboard
	04H-06H	Reserved
	07H	Canadian-English Keyboard
	08H-0AH	Reserved
	0BH	Italian Keyboard
	0CH	Reserved
	0DH	Dutch Keyboard
	0EH	Swedish Keyboard
	0FH	German Keyboard
	10H-12H	Reserved
	13H	Spanish
	14H	Reserved
	15H	Belgian (Flemish) Keyboard
	16H	Finnish Keyboard
	17H	United Kingdom Keyboard
	18H	French-Canadian Keyboard
	19H	Swiss-German Keyboard
	1AH	Norwegian Keyboard
	1BH	French Keyboard
	1CH	Danish Keyboard
	1DH	Katakana Keyboard
	1EH	Latin American-Spanish Keyboard
	1FH	United States-American Keyboard
Other	20H-2BH	Reserved
	2CH-2FH	Tone Generator
	30H-3FH	Reserved
Character Entry	40H-4FH	Reserved
	50H-5BH	Reserved
	5CH-5FH	Barcode Reader
Relative Positioners	60H-67H	Reserved
	68H-6BH	Mouse
	6CH-6FH	Trackball
	70H-7FH	Reserved

Device Type	ID Range	Device Description
Absolute Positioners	80H-87H	Reserved
	88H-8BH	Touchpad
	8CH-8FH	Touch Screen
	90H-97H	Graphics Tablet
	98H-9FH	Reserved
Keyboard	0A0H-0BFH	Compressed Keyboard (91–93 keys)
	0C0H-0DFH	Extended Keyboard (107–109 keys)
	0E0H-0FFH	Standard Keyboard (85–87 keys)

LD__SIZE__X	Maximum count (in units of resolution) for the x-axis.

LD__SIZE__Y	Maximum count (in units of resolution) for the y-axis.

LD__ABS__X	X position data for devices which report absolute coordinates (absolute devices).

LD__ABS__Y	Y position data for devices which report absolute coordinates.

LD__REL__X	Latest change in x position for devices which return coordinates relative to the previous position (relative devices).

LD__REL__Y	Latest change in y position for devices which return coordinates relative to the previous position.

LD__ACCUM__X	Accumulator used to sum partial movements when scaling from the physical device space to the logical device space. The value stored here represents a fraction of one logical unit for the x-axis.

LD__ACCUM__Y	Accumulator used to sum partial movements when scaling from the physical device space to the logical device space. The value stored here represents a fraction of one logical unit for the y-axis.

4.2.2.2 Logical ISR Event Records

A Logical ISR Event Record is not a data structure in the truest sense, but is a set of register definitions for inter-driver communication of input events. These definitions apply not only to Input System drivers but to application event drivers as well. Tables 4.3 and 4.4 define the Logical ISR Event Records.

Table 4.3

GID Button ISR Event Record

AH = F_ISR (00H)
DL = Physical device driver's vector address / 6
BX = Button information.

Bit	Value	Definition
0FH-08H	—	Reserved
07H	1	Button up
	0	Button down
06H-00H	—	Button number (0-7)

DH = Data Type
ES:0 = Pointer to Physical device driver header and Physical Describe Record.

Table 4.4

GID Motion ISR Event Record

AH = F_ISR (00H)
DL = Physical device driver's vector address / 6
BX = X axis motion in raw data form.
CX = Y axis motion in raw data form.
DH = Data Type
ES:0 = Pointer to physical device driver header and Physical Describe Record.

The button number in the Button information field (BX) denotes which button on the device is reporting data. Of special interest is button seven (proximity indicator) which is currently used by absolute devices to indicate that the device measurement field is active. For example, someone is touching the touch screen or the stylus is in contact with the tablet surface.

The Data Type field (DH) contains a code representing the current type of logical GID data stored in the event record. For button events this value will be T_KC_BUTTON. For logical GID motion events permissible types are: T_TS, T_POINTER and T_TABLET, which correspond to data originating from V_LTOUCH, V_LPOINTER, and V_LTABLET respectively. For a complete list of logical GID event data types see table 4.5.

Table 4.5

Logical GID Event Data Types

Type	Value	Definition
T__KC__BUTTON	09H	Button data
T__TS	45H	Specially formed data (80 x 25—default) generated by V__LTOUCH
T__TABLET	46H	Specially formed data (640 x 200 range—default) generated by V__LTABLET
T__POINTER	47H	Specially formed data (640 x 200 range—default) generated by V__LPOINTER

4.2.2.3 Application Event Drivers

As previously mentioned, applications may install a routine to handle interrupts from the logical GID drivers. Three predefined vectors in the HP__VECTOR__TABLE are initialized to the null driver (V__PNULL). The three vectors are V__EVENT__TOUCH, V__EVENT__POINTER, and V__EVENT__TABLET which are called by the logical GID drivers V__LTOUCH, V__LPOINTER, and V__LTABLET respectively when event interrupts are enabled by a call to SF__EVENT__ON. A call to SF__CREATE__EVENT sets the corresponding event vector to point to the user application event driver instead of the null driver.

The application event driver is required to support only one function, F__ISR. The driver should return RS__UNSUPPORTED for all unimplemented functions.

4.2.3 Logical GID Drivers

The drivers V__LTOUCH, V__LPOINTER and V__LTABLET represent the application interface to the Input System. These drivers provide functions to poll for data, enable/disable application event interrupts, enable/disable tracking and enable/disable clipping and/or scaling.

4.2.3.1 V__LTOUCH Driver (BP = 00C6H)

This section contains a detailed description of the touch screen driver. Table 4.6 contains a function code summary.

Table 4.6

Touch Screen Driver Function Code Summary

Vector Address	Func. Value	Function Equate	Definition
00C6H		V__LTOUCH	Application interface to Touch Screen
00C6H	00	F__ISR	Logical Interrupt
00C6H	02	F__SYSTEM	System functions
00C6H	02/00	SF__INIT	Initialize the driver data area
00C6H	02/02	SF__START	Start driver
00C6H	02/04	SF__REPORT__STATE	Report state of device
00C6H	02/06	SF__VERSION__DESC	Report driver version number
00C6H	02/08	SF__DEF__ATTR	Set default logical scaling attributes
00C6H	02/0A	SF__GET__ATTR	Get scaling attributes
00C6H	02/0C	SF__SET__ATTR	Set scaling attributes
00C6H	04	F__IO__CONTROL	I/O Control functions
00C6H	04/00	SF__LOCK	Unsupported
00C6H	04/02	SF__UNLOCK	Unsupported
00C6H	04/04	SF__TRACK__ON	Turn cursor track on
00C6H	04/06	SF__TRACK__OFF	Turn cursor track off
00C6H	04/08	SF__CREATE__EVENT	Establish a new routine to be called on logical device events
00C6H	04/0A	SF__EVENT__ON	Enable event call to parent driver
00C6H	04/0C	SF__EVENT__OFF	Disable event call to parent driver
00C6H	04/0E	SF__CLIPPING__ON	Enable logical device clipping
00C6H	04/10	SF__CLIPPING__OFF	Disable logical device clipping
00C6H	06	F__SAMPLE	Report absolute position of GID

Touch Screen Driver Functions Definitions

F__ISR (AH = 00H)

This function receives an ISR Event record from one of the physical GID drivers. The calling driver has handled the physical interrupt and updated the Physical Describe Record to reflect the event. This function translates the physical event into the logical coordinate system and calls its parent, V__EVENT__TOUCH, (if EVENT is enabled). In addition, this function passes the event to V__STRACK so that the sprite can be updated (if TRACK is enabled). This function is a response to a logical hardware interrupt and not user callable.

On Entry: AH = F__ISR (00H)
DH = Data Type
DL = Physical device driver's vector index.
ES:0 = Pointer to Physical device driver header and Physical Describe Record.
BP = V__LTOUCH (00C6H)

For Button Event:
BX = Button information.

Bit	Value	Definition
0FH-08H	—	Reserved
07H	1	Button up
	0	Button down
06H-00H	—	Button number (0-7)

For Motion Event:
BX = X axis motion in raw data form.
CX = Y axis motion in raw data form.

On Exit: AH = Return Status Code

Registers Altered: AX, BP, DS

Related Functions: SF__CREATE__EVENT, SF__EVENT__ON, SF__TRACK__ON

SF__INIT (AX = 0200H)

This subfunction is called to initialize the driver. Refer to Section 9 for a complete discussion of the protocol used in data space allocation.

On Entry: AH = F__SYSTEM (02H)
AL = SF__INIT (00H)
BX = "Last used DS" in HP Data Area
BP = V__LTOUCH (00C6H)

On Exit: AH = Return Status Code
BX = New "last used DS" in HP Data Area

Registers Altered: AX, BX, BP, DS

SF__START (AX = 0202H)

This subfunction starts the logical touch screen driver.

On Entry: AH = F__SYSTEM (02H)
AL = SF__START (02H)
BP = V__LTOUCH (00C6H)

On Exit: AH = Return Status Code

Registers Altered: AX, BP, DS

SF__REPORT__STATE (AX = 0204H)

This subfunction returns the LD__DEVICE__STATE field from the Logical Describe Record.

On Entry: AH = F__SYSTEM (02H)
AL = SF__REPORT__STATE (04H)
BP = V__LTOUCH (00C6H)

On Exit: AH = Return Status Code
DX = LD__DEVICE__STATE from Logical Describe Record

Registers Altered: AX, DX, BP, DS

SF__VERSION__DESC (AX = 0206H)

This subfunction returns the release date code and a double word pointer to the current version number. The date code consists of two BCD coded bytes containing the year and week of release. The BL register contains the number of years since 1960 and the BH register contains the week of the year.

On Entry: AH = F_SYSTEM (02H)
AL = SF_VERSION_DESC (06H)
BP = V_LTOUCH (00C6H)

On Exit: AH = Return Status Code
BX = Release date code
CX = Number of bytes in current version number
ES:DI = Pointer to the current version number

Registers Altered: AX, BX, CX, DI, ES, BP, DS

SF_DEF_ATTR (AX = 0208H)

This subfunction sets the attributes of the logical touch screen driver to their default values. The default attributes for the touch screen driver are: LD_SIZE_X = 79 and LD_SIZE_Y = 24.

On Entry: AH = F_SYSTEM (02H)
AL = SF_DEF_ATTR (08H)
BP = V_LTOUCH (00C6H)

On Exit: AH = Return Status Code

Registers Altered: AX, BP, DS

SF_GET_ATTR (AX = 020AH)

This subfunction returns the current scaling attributes, LD_SIZE_X and LD_SIZE_Y.

On Entry: AH = F_SYSTEM (02H)
AL = SF_GET_ATTR (0AH)
BP = V_LTOUCH (00C6H)

On Exit: AH = Return Status Code
BX = LD_SIZE_X (logical size along X axis)
CX = LD_SIZE_Y (logical size along Y axis)

Registers Altered: AX, BX, CX, BP, DS

SF_SET_ATTR (AX = 020CH)

This subfunction sets the scaling attributes, LD_SIZE_X and LD_SIZE_Y in the Logical Describe Record.

On Entry: AH = F__SYSTEM (02H)
AL = SF__SET__ATTR (0CH)
BX = LD__SIZE__X (logical size along X axis)
CX = LD__SIZE__Y (logical size along Y axis)
BP = V__LTOUCH (00C6H)

On Exit: AH = Return Status Code

Registers Altered: AX, BP, DS

SF__TRACK__ON (AX = 0404H)

This subfunction turns tracking on. For each movement of the logical device, V__STRACK will be called to update the graphics cursor (sprite) position.

On Entry: AH = F__IO__CONTROL (04H)
AL = SF__TRACK__ON (04H)
BP = V__LTOUCH (00C6H)

On Exit: AH = Return Status Code

Registers Altered: AX, BP, DS

SF__TRACK__OFF (AX = 0406H)

This subfunction turns tracking off.

On Entry: AH = F__IO__CONTROL (04H)
AL = SF__TRACK__OFF (06H)
BP = V__LTOUCH (00C6H)

On Exit: AH = Return Status Code

Registers Altered: AX, BP, DS

SF__CREATE__EVENT (AX = 0408H)

This subfunction establishes the routine to be called on logical device events. The IP, CS, and DS of the routine are passed to this subfunction. These values are exchanged with the vector entry of the V__EVENT__TOUCH driver in the HP__VECTOR__TABLE, V__EVENT__TOUCH being the parent of the logical touch screen driver. The IP, CS, and DS of the previous routine are returned to the caller. Note that this subfunction does not enable the event call to the parent routine; this must be done explicitly using SF__EVENT__ON.

The ISR event records passed to the V_EVENT_TOUCH driver will have one of the following two formats depending on the Data Type stored in DL.

V_EVENT_TOUCH Button ISR Event Record:

AH = F_ISR (00H)
DL = Physical device driver's vector address / 6
BX = Button information.

Bit	Value	Definition
0FH-08H	—	Reserved
07H	1	Button up
	0	Button down
06H-00H	—	Button number (0-7)

DH = Data Type
ES:0 = Pointer to V_LTOUCH device driver header and Logical Describe Record.

V_EVENT_TOUCH Motion ISR Event Record:

AH = F_ISR (00H)
DL = Physical device driver's vector address / 6
BX = A number between 0 and LD_SIZE_X
CX = A number between 0 and LD_SIZE_Y
DH = Data Type
ES:0 = Pointer to V_LTOUCH device driver header and Logical Describe Record.

On Entry: AH = F_IO_CONTROL (04H)
AL = SF_CREATE_EVENT (08H)
BP = V_LTOUCH (00C6H)
DX = DS of new V_EVENT_TOUCH routine
SI = IP of new V_EVENT_TOUCH routine
ES = CS of new V_EVENT_TOUCH routine

On Exit: AH = Return Status Code
DX = DS of previous V_EVENT_TOUCH routine
SI = IP of previous V_EVENT_TOUCH routine
ES = CS of previous V_EVENT_TOUCH routine

Registers Altered: AX, DX, SI, BP, ES, DS

Related Functions: SF_EVENT_ON

This example shows how to use the SF_CREATE_EVENT function. The routine EVENT will be the event procedure that is called when events are enabled.

```
EVENT       PROC FAR
        CMP  AH, F_ISR                  ; only support function F_ISR
        JE   PROCESS_EVENT
        MOV  AH, RS_UNSUPPORTED
        IRET
PROCESS_EVENT:
        .                               ; code to process data
        .                               ; (see touch screen
        .                               ; event record)
        MOV  AH, RS_SUCCESSFUL          ; return successful completion
        IRET
EVENT       ENDP

        MOV  AH, F_IO_CONTROL
        MOV  AL, SF_CREATE_EVENT
        MOV  BP, V_LTOUCH
        MOV  DX, DS                     ; want to use the current data
                                        ; segment for event DS
        PUSH CS
        POP  ES                         ; current CS is also segment
                                        ; of event routine
        LEA  SI, CS:EVENT               ; get the IP of the event
                                        ; routine
        PUSH DS                         ; save current DS
        INT  HP_ENTRY                   ; call extended BIOS driver
        POP  DS
```

SF_EVENT_ON (AX = 040AH)

This subfunction enables the event (parent) call to the touch screen event routine (V_EVENT_TOUCH). The link to the touch screen event routine must have already been established using SF_CREATE_EVENT.

On Entry: AH = F_IO_CONTROL (04H)
AL = SF_EVENT_ON (0AH)
BP = V_LTOUCH (00C6H)

On Exit: AH = Return Status Code

Registers Altered: AX, BP, DS

Related Functions: SF__CREATE__EVENT, SF__EVENT__OFF

SF__EVENT__OFF (AX = 040CH)

This subfunction disables the call to the touch screen event routine

On Entry: AH = F__IO__CONTROL (04H)
AL = SF__EVENT__OFF (0CH)
BP = V__LTOUCH (00C6H)

On Exit: AH = Return Status Code

Registers Altered: AX, BP, DS

SF__CLIPPING__ON (AX = 040EH)

This subfunction enables logical device clipping. Physical device motion will be scaled to logical space and will be clipped to avoid overflow or underflow. Clipping is activated for both absolute and relative motion.

When there is a relative device mapped to this device driver, clipping works the best. It will make sure that the new position always falls within the logical space.

On Entry: AH = F__IO__CONTROL (04H)
AL = SF__CLIPPING__ON (0EH)
BP = V__LTOUCH (00C6H)

On Exit: AH = Return Status Code

Registers Altered: AX, BP, DS

SF__CLIPPING OFF (AX = 0410H)

This subfunction disables logical device clipping. Physical device motion will be scaled to logical space, but overflow or underflow may occur.

```
On Entry:  AH  = F__IO__CONTROL (04H)
           AL  = SF__CLIPPING__OFF (10H)
           BP  = V__LTOUCH (00C6H)

On Exit:   AH  = Return Status Code
```

Registers Altered: AX, BP, DS

F__SAMPLE (AH = 06H)

This function allows an application to poll the touch screen device. This function reports the current absolute position of the logical device in a form similar to a Logical ISR Event Record.

```
On Entry:  AH  = F__SAMPLE (06H)
           BP  = V__LTOUCH (00C6H)

On Exit:   AH  = Return Status Code
           BX  = Current logical position along X axis
           CX  = Current logical position along Y axis
           DL  = LD__TRANSITION field of Logical Describe Record
           DH  = LD__STATE field of Logical Describe Record
         ES:0  = Pointer to logical device header and Describe Record
```

Registers Altered: AX, BX, CX, DX, BP, DS, ES

The following is an example of how to call the F__SAMPLE function.

```
MOV  AH, F__SAMPLE     ; load function code
MOV  BP, V__LTOUCH     ; load vector address
PUSH DS                ; save the current DS
INT  HP__ENTRY         ; call extended BIOS driver
POP  DS                ; restore DS
```

4.2.3.2 V__LPOINTER Driver (BP = 00C0H)

This section contains a detailed description of the pointer driver. Table 4.7 summarizes the functions supported by this driver.

Table 4.7

Pointer Driver Function Code Summary

Vector Address	Func. Value	Function Equate	Definition
00C0H		V__LPOINTER	Application interface to Pointer/Mouse
00C0H	00	F__ISR	Logical Interrupt
00C0H	02	F__SYSTEM	System functions
00C0H	02/00	SF__INIT	Initialize the driver data area
00C0H	02/02	SF__START	Start driver
00C0H	02/04	SF__REPORT__STATE	Report state of device
00C0H	02/06	SF__VERSION__DESC	Report driver version number
00C0H	02/08	SF__DEF__ATTR	Set default logical scaling attributes
00C0H	02/0A	SF__GET__ATTR	Get scaling attributes
00C0H	02/0C	SF__SET__ATTR	Set scaling attributes
00C0H	04	F__IO__CONTROL	I/O Control Functions
00C0H	04/00	SF__LOCK	Unsupported
00C0H	04/02	SF__UNLOCK	Unsupported
00C0H	04/04	SF__TRACK__ON	Turn cursor track on
00C0H	04/06	SF__TRACK__OFF	Turn cursor track off
00C0H	04/08	SF__CREATE__EVENT	Establish a new routine to be called on logical device events
00C0H	04/0A	SF__EVENT__ON	Enable event call to parent driver
00C0H	04/0C	SF__EVENT__OFF	Disable event call to parent driver
00C0H	04/0E	SF__CLIPPING__ON	Enable logical device clipping
00C0H	04/10	SF__CLIPPING__OFF	Disable logical device clipping
00C0H	06	F__SAMPLE	Report absolute position of GID

Pointer Driver Function Definitions

F__ISR (AH = 00H)

This function receives an ISR Event record from one of the physical GID drivers. The calling driver has handled the physical interrupt and updated the Physical Describe Record to reflect the event. This function translates the physical event into the logical coordinate system and calls its parent, V__EVENT__POINTER, (if EVENT is enabled). In addition, this function passes the event to V__STRACK so that the sprite can be updated (if TRACK is enabled). This function is a response to a logical hardware interrupt and not user callable.

On Entry: AH = F__ISR (00H)
DH = Data Type
DL = Physical device driver's vector index.
ES:0 = Pointer to physical device driver header and Physical Describe Record.
BP = V__LPOINTER (00C0H)

For Button Event:
BX = Button information.

Bit	Value	Definition
0FH-08H	—	Reserved
07H	1	Button up
	0	Button down
06H-00H	—	Button number (0-7)

For Motion Event:
BX = X axis motion in raw data form.
CX = Y axis motion in raw data form.

On Exit: AH = Return Status Code

Registers Altered: AX, BP, DS

Related Functions: SF__CREATE__EVENT, SF__EVENT__ON, SF__TRACK__ON

SF__INIT (AX = 0200H)

This subfunction is called to initialize the driver. Refer to Section 9 for a complete discussion of the protocol used in data space allocation.

On Entry: AH = F__SYSTEM (02H)
AL = SF__INIT (00H)
BX = "Last used DS" in HP Data Area
BP = V__LPOINTER (00C0H)

On Exit: AH = Return Status Code
BX = New "last used DS" in HP Data Area

Registers Altered: AX, BX, BP, DS

SF_START (AX = 0202H)

This subfunction starts the logical pointer driver.

On Entry: AH = F_SYSTEM (02H)
AL = SF_START (02H)
BP = V_LPOINTER (00C0H)

On Exit: AH = Return Status Code

Registers Altered: AX, BP, DS

SF_REPORT_STATE (AX = 0204H)

This subfunction returns the LD_DEVICE_STATE field from the Logical Describe Record.

On Entry: AH = F_SYSTEM (02H)
AL = SF_REPORT_STATE (04H)
BP = V_LPOINTER (00C0H)

On Exit: AH = Return Status Code
DX = LD_DEVICE_STATE from Logical Describe Record

Registers Altered: AX, DX, BP, DS

SF_VERSION_DESC (AX = 0206H)

This subfunction returns the release date code and a double word pointer to the current version number. The date code consists of two BCD coded bytes containing the year and week of release. The BL register contains the number of years since 1960 and the BH register contains the week of the year.

On Entry: AH = F_SYSTEM (02H)
AL = SF_VERSION_DESC (06H)
BP = V_LPOINTER (00C0H)

On Exit: AH = Return Status Code
BX = Release date code
CX = Number of bytes in current version number
ES:DI = Pointer to the current version number

Registers Altered: AX, BX, CX, DI, ES, BP, DS

SF_DEF_ATTR (AX = 0208H)

This subfunction sets the attributes of the logical pointer driver to their default values. The default attributes for the pointer driver are: LD_SIZE_X = 639 and LD_SIZE_Y = 199.

On Entry: AH = F_SYSTEM (02H)
AL = SF_DEF_ATTR (08H)
BP = V_LPOINTER (00C0H)

On Exit: AH = Return Status Code

Registers Altered: AX, BP, DS

SF_GET_ATTR (AX = 020AH)

This subfunction returns the current scaling attributes, LD_SIZE_X and LD_SIZE_Y.

On Entry: AH = F_SYSTEM (02H)
AL = SF_GET_ATTR (0AH)
BP = V_LPOINTER (00C0H)

On Exit: AH = Return Status Code
BX = LD_SIZE_X (logical size along X axis)
CX = LD_SIZE_Y (logical size along Y axis)

Registers Altered: AX, BX, CX, BP, DS

SF_SET_ATTR (AX = 020CH)

This subfunction sets the scaling attributes, LD_SIZE_X and LD_SIZE_Y in the Logical Describe Record.

On Entry: AH = F_SYSTEM (02H)
AL = SF_SET_ATTR (0CH)
BX = LD_SIZE_X (logical size along X axis)
CX = LD_SIZE_Y (logical size along Y axis)
BP = V_LPOINTER (00C0H)

On Exit: AH = Return Status Code

Registers Altered: AX, BP, DS

SF_TRACK_ON (AX = 0404H)

This subfunction turns tracking on. For each movement of the logical device, V_STRACK will be called to update the graphics cursor (sprite) position.

On Entry: AH = F_IO_CONTROL (04H)
AL = SF_TRACK_ON (04H)
BP = V_LPOINTER (00C0H)

On Exit: AH = Return Status Code

Registers Altered: AX, BP, DS

SF_TRACK_OFF (AX = 0406H)

This subfunction turns tracking off.

On Entry: AH = F_IO_CONTROL (04H)
AL = SF_TRACK_OFF (06H)
BP = V_LPOINTER (00C0H)

On Exit: AH = Return Status Code

Registers Altered: AX, BP, DS

SF_CREATE_EVENT (AX = 0408H)

This subfunction establishes the routine to be called on logical device events. The IP, CS, and DS of the routine are passed to this subfunction. These values are exchanged with the vector entry of the V_EVENT_POINTER driver in the HP_VECTOR_TABLE, V_EVENT_POINTER being the parent of the logical pointer driver. The IP, CS, and DS of the previous routine are returned to the caller. Note that this subfunction does not enable the event call to the parent routine; this must be done explicitly using SF_EVENT_ON.

The ISR event records passed to the V_EVENT_POINTER driver will have one of the following two formats depending on the Data Type stored in DL.

V_EVENT_POINTER Button ISR Event Record:

AH = F_ISR (00H)
DL = Physical device driver's vector address / 6
BX = Button information.

Bit	Value	Definition
0FH-08H	—	Reserved
07H	1	Button up
	0	Button down
06H-00H	—	Button number (0-7)

DH = Data Type
ES:0 = Pointer to V_LPOINTER device driver header and Logical Describe Record.

V_EVENT_POINTER Motion ISR Event Record:

AH = F_ISR (00H)
DL = Physical device driver's vector address / 6
BX = Relative movement in the X direction
(Positive number indicates movement to the right)
CX = Relative movement in the Y direction
(Positive number indicates movement down)
DH = Data Type
ES:0 = Pointer to V_LPOINTER device driver header and Logical Describe Record.

On Entry: AH = F_IO_CONTROL (04H)
AL = SF_CREATE_EVENT (08H)
BP = V_LPOINTER (00C0H)
DX = DS of new V_EVENT_POINTER routine
SI = IP of new V_EVENT_POINTER routine
ES = CS of new V_EVENT_POINTER routine

On Exit: AH = Return Status Code
DX = DS of previous V_EVENT_POINTER routine
SI = IP of previous V_EVENT_POINTER routine
ES = CS of previous V_EVENT_POINTER routine

Registers Altered: AX, DX, SI, BP, ES, DS

Related Functions: SF_EVENT_ON

This example shows how to use the SF_CREATE_EVENT function. The routine EVENT will be the event procedure that is called when events are enabled.

```
EVENT      PROC  FAR
      CMP  AH, F_JSR                    ; only support function F_JSR
      JE   PROCESS_EVENT
      MOV  AH, RS_UNSUPPORTED
      IRET
PROCESS_EVENT:
      .                                 ; code to process data (see
      .                                 ; pointer event record)
      .
      MOV  AH, RS_SUCCESSFUL            ; return successful completion
      IRET
EVENT      ENDP

      MOV  AH, F_IO_CONTROL
      MOV  AL, SF_CREATE_EVENT
      MOV  BP, V_LPOINTER
      MOV  DX, DS                       ; want to use the current data
                                        ; segment for event DS
      PUSH CS
      POP  ES                           ; current CS is also segment
                                        ; of event routine
      LEA  SI, CS:EVENT                 ; get the IP of the event
                                        ; routine
      PUSH DS                           ; save current DS
      INT  HP_ENTRY                     ; call extended BIOS driver
      POP  DS
```

SF_EVENT_ON (AX = 040AH)

This subfunction enables the event (parent) call to the pointer event routine (V_EVENT_POINTER). The link to the pointer event routine must have already been established using SF_CREATE_EVENT.

On Entry: AH = F_IO_CONTROL (04H)
AL = SF_EVENT_ON (0AH)
BP = V_LPOINTER (00C0H)

On Exit: AH = Return Status Code

Registers Altered: AX, BP, DS

Related Functions: SF_CREATE_EVENT, SF_EVENT_OFF

SF__EVENT__OFF (AX = 040CH)

This subfunction disables the call to the pointer event routine.

On Entry: AH = F__IO__CONTROL (04H)
AL = SF__EVENT__OFF (0CH)
BP = V__LPOINTER (00C0H)

On Exit: AH = Return Status Code

Registers Altered: AX, BP, DS

SF__CLIPPING__ON (AX = 040EH)

This subfunction enables logical device clipping. Physical device motion will be scaled to logical space and will be clipped to avoid overflow or underflow. Clipping is activated for both absolute and relative motion.

When there is a relative device mapped to this device driver, clipping works the best. It will make sure that the new position always falls within the logical space.

On Entry: AH = F__IO__CONTROL (04H)
AL = SF__CLIPPING__ON (0EH)
BP = V__LPOINTER (00C0H)

On Exit: AH = Return Status Code

Registers Altered: AX, BP, DS

SF__CLIPPING__OFF (AX = 0410H)

This subfunction disables logical device clipping. Physical device motion will be scaled to logical space, but overflow or underflow may occur.

On Entry: AH = F__IO__CONTROL (04H)
AL = SF__CLIPPING__OFF (10H)
BP = V__LPOINTER (0C0H)

On Exit: AH = Return Status Code

Registers Altered: AX, BP, DS

F_SAMPLE (AH = 06H)

This function allows an application to poll the pointer device. This function reports the current absolute position of the logical device in a form similar to a Logical ISR Event Record.

On Entry:	AH	= F_SAMPLE (06H)
	BP	= V_LPOINTER (00C0H)

On Exit:	AH	= Return Status Code
	BX	= Current logical position along X axis
	CX	= Current logical position along Y axis
	DL	= LD_TRANSITION field of Logical Describe Record
	DH	= LD_STATE field of Logical Describe Record
	ES:0	= Pointer to logical device header and Describe Record

Registers Altered: AX, BX, CX, DX, BP, DS, ES

The following is an example of how to call the F_SAMPLE function.

```
MOV  AH, F_SAMPLE        ; load function code
MOV  BP, V_LPOINTER      ; load vector address
PUSH DS                  ; save the current DS
INT  HP_ENTRY            ; call extended BIOS driver
POP  DS                  ; restore DS
```

4.2.3.3 V_LTABLET Driver (BP = 00BAH)

This section contains a detailed description of the tablet driver. See table 4.8 for a summary of functions supported by this driver.

Table 4.8

Tablet Driver Function Code Summary

Vector Address	Func. Value	Function Equate	Definition
00BAH		V_LTABLET	Application interface to Tablet
00BAH	00	F_ISR	Logical Interrupt
00BAH	02	F_SYSTEM	System functions
00BAH	02/00	SF_INIT	Initialize the driver data area
00BAH	02/02	SF_START	Start driver
00BAH	02/04	SF_REPORT_STATE	Report state of device
00BAH	02/06	SF_VERSION_DESC	Report driver version number
00BAH	02/08	SF_DEF_ATTR	Set default logical scaling attributes
00BAH	02/0A	SF_GET_ATTR	Get scaling attributes
00BAH	02/0C	SF_SET_ATTR	Set scaling attributes
00BAH	04	F_IO_CONTROL	I/O Control Functions
00BAH	04/00	SF_LOCK	Unsupported
00BAH	04/02	SF_UNLOCK	Unsupported
00BAH	04/04	SF_TRACK_ON	Turns cursor track on
00BAH	04/06	SF_TRACK_OFF	Turns cursor track off
00BAH	04/08	SF_CREATE_EVENT	Establish a new routine to be called on logical device events
00BAH	04/0A	SF_EVENT_ON	Enable event call to parent driver
00BAH	04/0C	SF_EVENT_OFF	Disable event call to parent driver
00BAH	04/0E	SF_CLIPPING_ON	Enable logical device clipping
00BAH	04/10	SF_CLIPPING_OFF	Disable logical device clipping
00BAH	06	F_SAMPLE	Report absolute position of GID

Tablet Driver Functions Definition

F_ISR (AH = 00H)

This function receives an ISR Event record from one of the physical GID drivers. The calling driver has handled the physical interrupt and updated the Physical Describe Record to reflect the event. This function translates the physical event into the logical coordinate system and calls its parent, V_EVENT_TABLET, (if EVENT is enabled). In addition, this function passes the event to V_STRACK so that the sprite can be updated (if TRACK is enabled). This function is a response to a logical hardware interrupt and not user callable.

On Entry: AH = F__ISR (00H)
DH = Data Type
DL = Physical device driver's vector index.
ES:0 = Pointer to physical device driver header and Physical Describe Record.
BP = V__LTABLET (00BAH)

For Button Event:
BX = Button information.

Bit	Value	Definition
0FH-08H	—	Reserved
07H	1	Button up
	0	Button down
06H-00H	—	Button number (0-7)

For Motion Event:
BX = X axis motion in raw data form.
CX = Y axis motion in raw data form.

On Exit: AH = Return Status Code

Registers Altered: AX, BP, DS

Related Functions: SF__CREATE__EVENT, SF__EVENT__ON, SF__TRACK__ON

SF__INIT (AX = 0200H)

This subfunction is called to initialize the driver. Refer to Section 9 for a complete discussion of the protocol used in data space allocation.

On Entry: AH = F__SYSTEM (02H)
AL = SF__INIT (00H)
BX = "Last used DS" in HP Data Area
BP = V__LTABLET (00BAH)

On Exit: AH = Return Status Code
BX = New "last used DS" in HP Data Area

Registers Altered: AX, BX, BP, DS

SF_START (AX = 0202H)

This subfunction starts the logical tablet driver.

On Entry:	AH	=	F_SYSTEM (02H)
	AL	=	SF_START (02H)
	BP	=	V_LTABLET (00BAH)

On Exit:	AH	=	Return Status Code

Registers Altered: AX, BP, DS

SF_REPORT_STATE (AX = 0204H)

This subfunction returns the LD_DEVICE_STATE field from the Logical Describe Record.

On Entry:	AH	=	F_SYSTEM (02H)
	AL	=	SF_REPORT_STATE (04H)
	BP	=	V_LTABLET (00BAH)

On Exit:	AH	=	Return Status Code
	DX	=	LD_DEVICE_STATE from Logical Describe Record

Registers Altered: AX, DX, BP, DS

SF_VERSION_DESC (AX = 0206H)

This subfunction returns the release date code and a double word pointer to the current version number. The date code consists of two BCD coded bytes containing the year and week of release. The BL register contains the number of years since 1960 and the BH register contains the week of the year.

On Entry:	AH	=	F_SYSTEM (02H)
	AL	=	SF_VERSION_DESC (06H)
	BP	=	V_LTABLET (00BAH)

On Exit:	AH	=	Return Status Code
	BX	=	Release date code
	CX	=	Number of bytes in current version number
	ES:DI	=	Pointer to the current version number

Registers Altered: AX, BX, CX, DI, ES, BP, DS

SF_DEF_ATTR (AX = 0208H)

This subfunction sets the attributes of the logical tablet driver to their default values. The default attributes for the tablet driver are: LD_SIZE_X = 639 and LD_SIZE_Y = 199.

On Entry: AH = F_SYSTEM (02H)
AL = SF_DEF_ATTR (08H)
BP = V_LTABLET (00BAH)

On Exit: AH = Return Status Code

Registers Altered: AX, BP, DS

SF_GET_ATTR (AX = 020AH)

This subfunction returns the current scaling attributes, LD_SIZE_X and LD_SIZE_Y.

On Entry: AH = F_SYSTEM (02H)
AL = SF_GET_ATTR (0AH)
BP = V_LTABLET (00BAH)

On Exit: AH = Return Status Code
BX = LD_SIZE_X (logical size along X axis)
CX = LD_SIZE_Y (logical size along Y axis)

Registers Altered: AX, BX, CX, BP, DS

SF_SET_ATTR (AX = 020CH)

This subfunction sets the scaling attributes, LD_SIZE_X and LD_SIZE_Y in the Logical Describe Record.

On Entry: AH = F_SYSTEM (02H)
AL = SF_SET_ATTR (0CH)
BX = LD_SIZE_X (logical size along X axis)
CX = LD_SIZE_Y (logical size along Y axis)
BP = V_LTABLET (00BAH)

On Exit: AH = Return Status Code

Registers Altered: AX, BP, DS

SF__TRACK__ON (AX = 0404H)

This subfunction turns tracking on. For each movement of the logical device, V__STRACK will be called to update the graphics cursor (sprite) location.

```
On Entry:  AH = F__IO__CONTROL (04H)
           AL = SF__TRACK__ON (04H)
           BP = V__LTABLET (00BAH)

On Exit:   AH = Return Status Code
```

Registers Altered: AX, BP, DS

SF__TRACK__OFF (AX = 0406H)

This subfunction turns tracking off.

```
On Entry:  AH = F__IO__CONTROL (04H)
           AL = SF__TRACK__OFF (06H)
           BP = V__LTABLET (00BAH)

On Exit:   AH = Return Status Code
```

Registers Altered: AX, BP, DS

SF__CREATE__EVENT (AX = 0408H)

This subfunction establishes the routine to be called on logical device events. The IP, CS, and DS of the routine are passed to this subfunction. These values are exchanged with the vector entry of the V__EVENT__TABLET driver in the HP__VECTOR__TABLE, V__EVENT__TABLET being the parent of the logical tablet driver. The IP, CS, and DS of the previous routine are returned to the caller. Note that this subfunction does not enable the event call to the parent routine; this must be done explicitly using SF__EVENT__ON.

The ISR event records passed to the V__EVENT__TABLET driver will have one of the following two formats depending on the data type stored in DL.

V__EVENT__TABLET Button ISR Event Record:

AH = F__ISR (00H)
DL = Physical device driver's vector address / 6
BX = Button information.

Bit	Value	Definition
0FH-08H	—	Reserved
07H	1	Button up
	0	Button down
06H-00H	—	Button number (0-7)

DH = Data Type
ES:0 = Pointer to V__LTABLET device driver header and Logical Describe Record.

V__EVENT__TABLET Motion ISR Event Record:

AH = F__ISR (00H)
DL = Physical device driver's vector address / 6
BX = A number between 0 and LD__SIZE__X
CX = A number between 0 and LD__SIZE__Y
DH = Data Type
ES:0 = Pointer to V__TABLET device driver header and Logical Describe Record.

On Entry: AH = F__IO__CONTROL (04H)
AL = SF__CREATE__EVENT (08H)
BP = V__LTABLET (00BAH)
DX = DS of new V__EVENT__TABLET routine
SI = IP of new V__EVENT__TABLET routine
ES = CS of new V__EVENT__TABLET routine

On Exit: AH = Return Status Code
DX = DS of previous V__EVENT__TABLET routine
SI = IP of previous V__EVENT__TABLET routine
ES = CS of previous V__EVENT__TABLET routine

Registers Altered: AX, DX, SI, BP, ES, DS

Related Functions: SF__EVENT__ON

This example shows how to use the SF__CREATE__EVENT function. The routine EVENT will be the event procedure that is called when events are enabled.

```
EVENT     PROC  FAR
      CMP       AH, F_ISR                ; only support function F_ISR
      JE        PROCESS_EVENT
      MOV       AH, RS_UNSUPPORTED
      IRET
PROCESS_EVENT:
      .                                  ; code to process data (see
      .                                  ; tablet event record)
      .
      MOV       AH, RS_SUCCESSFUL        ; return successful completion
      IRET
EVENT     ENDP

      MOV       AH, F_IO_CONTROL
      MOV       AL, SF_CREATE_EVENT
      MOV       BP, V_LTABLET
      MOV       DX, DS                   ; want to use the current data
                                         ; segment for event DS
      PUSH      CS
      POP       ES                       ; current CS is also segment
                                         ; of event routine
      LEA       SI, CS:EVENT             ; get the IP of the event
                                         ; routine
      PUSH      DS                       ; save current DS
      INT       HP_ENTRY                 ; call extended BIOS driver
      POP       DS
```

SF_EVENT_ON (AX = 040AH)

This subfunction enables the event (parent) call to the tablet event routine (V_EVENT_TABLET). The link to the tablet event routine must have already been established using SF_CREATE_EVENT.

On Entry: AH = F_IO_CONTROL (04H)
AL = SF_EVENT_ON (0AH)
BP = V_LTABLET (00BAH)

On Exit: AH = Return Status Code

Registers Altered: AX, BP, DS

Related Functions: SF_CREATE_EVENT, SF_EVENT_OFF

SF__EVENT__OFF (AX = 040CH)

This subfunction disables the call to the tablet event routine.

On Entry: AH = F__IO__CONTROL (04H)
AL = SF__EVENT__OFF (0CH)
BP = V__LTABLET (00BAH)

On Exit: AH = Return Status Code

Registers Altered: AX, BP, DS

SF__CLIPPING__ON (AX = 040EH)

This subfunction enables logical device clipping. Physical device motion will be scaled to logical space and will be clipped to avoid overflow or underflow. Clipping is activated for both absolute and relative motion.

When there is a relative device mapped to this device driver, clipping works the best. It will make sure that the new position always falls within the logical space.

On Entry: AH = F__IO__CONTROL (04H)
AL = SF__CLIPPING__ON (0EH)
BP = V__LTABLET (00BAH)

On Exit: AH = Return Status Code

Registers Altered: AX, BP, DS

SF__CLIPPING__OFF (AX = 0410H)

This subfunction disables logical device clipping. Physical device motion will be scaled to logical space, but overflow or underflow may occur.

On Entry: AH = F__IO__CONTROL (04H)
AL = SF__CLIPPING__OFF (10H)
BP = V__LTABLET (00BAH)

On Exit: AH = Return Status Code

Registers Altered: AX, BP, DS

F_SAMPLE (AH = 06H)

This function allows an application to poll the tablet device. This function reports the current absolute position of the logical device in a form similar to a Logical ISR Event Record.

On Entry: AH = F_SAMPLE (06H)
BP = V_LTABLET (00BAH)

On Exit: AH = Return Status Code
BX = Current logical position along X axis
CX = Current logical position along Y axis
DL = LD_TRANSITION field of Logical Describe Record
DH = LD_STATE field of Logical Describe Record
ES:0 = Pointer to logical device header and Describe Record

Registers Altered: AX, BX, CX, DX, BP, DS, ES

The following is an example of how to call the F_SAMPLE function.

```
MOV  AH, F_SAMPLE     ; load function code
MOV  BP, V_LTABLET    ; load vector address
PUSH DS               ; save the current DS
INT  HP_ENTRY         ; call extended BIOS driver
POP  DS               ; restore DS
```

4.2.4 Application Event Driver Example

The following program is an example of how to input touch screen data using application event interrupts. The program installs an application event driver using the SF_CREATE_EVENT function and enables event interrupts using the SF_EVENT_ON function. The event handler supports only the F_ISR function which processes both button and motion Logical ISR Event Records.

Touch Example

```
.286c
page    59,132
title   TOUCH Example
;***DRIVER HEADER*****************************************************
;
;  NAME:          TOUCH Example
;
;  DESCRIPTION: This program demonstrates how touch works.
;
;  LIST OF SECTIONS:
;
;
;*********************************************************************
```

```
                                         page

                                        HP_SHEADER              struc
0000  0000                              DH_ATR                  dw      0
0002  0000                              DH_NAME_INDEX           dw      0
0004  0000                              DH_V_DEFAULT            dw      0
0006  0000                              DH_P_CLASS              dw      0
0008  0000                              DH_C_CLASS              dw      0
000A  0000                              DH_V_PARENT             dw      0
000C  0000                              DH_V_CHILD              dw      0
000E  00                                DH_MAJOR                db      0
000F  00                                DH_MINOR                db      0
0010                                    HP_SHEADER              ends
= 006F                                  HP_ENTRY                equ     06FH
                                        SYSCALL                 macro   vector
                                         ifnb                   <vector>
                                                                mov     bp,vector
                                         endif
                                                                int     HP_ENTRY
                                                                endm
= 8000                                  ATR_HP                  equ     8000H
= 0000                                  CL_NULL                 equ     0000H
= 0000                                  F_ISR                   equ     0000H
= 0004                                  F_IO_CONTROL            equ     0004H
= 0008                                  SF_CREATE_EVENT         equ     0008H
= 000C                                  SF_EVENT_OFF            equ     000CH
= 000A                                  SF_EVENT_ON             equ     000AH
= 0000                                  RS_SUCCESSFUL           equ     0000H
= 0002                                  RS_UNSUPPORTED          equ     0002H
= 0009                                  T_KC_BUTTON             equ     09H      ; reported by the physical driver to the logical drive
                                                                                 ; PGID translates T_KC_ITF to T_KC_BUTTON and filters
                                                                                 ; any other scancode out of the data stream
= 0045                                  T_TS                    equ     45H      ;  Specially formed data  ( 0..80 x 0..25 range - defa
= 0006                                  V_DOLITTLE              equ     0006H
= 00C6                                  V_LTOUCH                equ     00C6H
= 0060                                  V_EVENT_TOUCH           equ     0060H
= 0001                                  READ_CHAR_ECHO          equ     01H
= 0080                                  MAKE_BREAK_BIT          equ     10000000B
= 004C                                  TERMINATE_PROC          equ     4CH

0000                                    TS_EVENT_HEADR          segment
=                                       EXAM_HP_ATTR            equ     ATR_HP
0000  8000                                          HP_SHEADER          <EXAM_HP_ATTR,V_EVENT_TOUCH/6,V_EVENT_TOUCH,CL_NULL,CL_NULL,V_
                                        TLE,V_DOLITTLE>
0002  0010
0004  0060
0006  0000
0008  0000
000A  0006
000C  0006
000E  00
000F  00

0010                                    TS_EVENT_HEADR  ends
0000                                    DATA_SEG        segment
```

Touch Example (cont.)

```
????                     SAVE_CS           dw      ?
????                     SAVE_IP           dw      ?
????                     SAVE_DS           dw      ?
   50 [                  STACK             dw      80 dup (?)
         ????
                ]

????                     STK_TOP           dw      ?
                         DATA_SEG          ends
                         CODE_SEG          segment
                                           assume  cs:CODE_SEG,ds:DATA_SEG,ss:DATA_SEG
B8 ---- R                BEGIN:            mov     ax,DATA_SEG             ;Load up the ds register with the data segment
8E D8                                      mov     ds,ax
8E D0                                      mov     ss,ax                   ;The stack segment is also in the code segment
8B 26 00A6 R                               mov     sp,STK_TOP              ;Point to the top of the stack
E8 001D R                                  call    TOUCH_ENABLE
B4 01                    INPUT_LOOP:       mov     ah,READ_CHAR_ECHO       ;Read a character w/echo until "^"
CD 21                                      int     21H
3C 5E                                      cmp     al,"^"                  ;Is this the exit character?
75 F8                                      jne     INPUT_LOOP
E8 0084 R                EXIT_PROG:        call    TOUCH_RESTORE
B4 4C                                      mov     ah,TERMINATE_PROC       ;Exit
CD 21                                      int     21H
                         TOUCH_ENABLE      proc
B4 04                                      mov     ah,F_IO_CONTROL         ;Move my touch event handler into the HP vector tab
                         le
B0 08                                      mov     al,SF_CREATE_EVENT
8C CB                                      mov     bx,cs
8E C3                                      mov     es,bx
8D 36 0048 R                               lea     si,TOUCH_HANDLER
BA ---- R                                  mov     dx,TS_EVENT_HEADR
                                           syscall V_LTOUCH
BD 00C6                 +                          mov     bp,V_LTOUCH
CD 6F                   +                          int     HP_ENTRY
8C C0                                      mov     ax,es                   ;Save the old event values
A3 0000 R                                  mov     word ptr SAVE_CS,ax
89 36 0002 R                               mov     word ptr SAVE_IP,si
89 16 0004 R                               mov     word ptr SAVE_DS,dx
B4 04                                      mov     ah,F_IO_CONTROL         ;Start accepting calls
B0 0A                                      mov     al,SF_EVENT_ON
                                           syscall V_LTOUCH
BD 00C6                 +                          mov     bp,V_LTOUCH
CD 6F                   +                          int     HP_ENTRY
C3                                         ret
                         TOUCH_ENABLE      endp

                         TOUCH_HANDLER     proc
80 FC 00                                   cmp     ah,F_ISR                ;Logical interrupt?
74 03                                      je      PROCESS_ISR             ; yes, continue
B4 02                                      mov     ah,RS_UNSUPPORTED       ;set return code
CF                                         iret
60                       PROCESS_ISR:      pusha                           ;Save all the registers
80 FE 45                                   cmp     dh,T_TS                 ;Is this a position report or a make/break report
74 07                                      je      short POS_REPORT
80 FE 09                                   cmp     dh,T_KC_BUTTON
```

Touch Example (cont.)

```
0059  74 0E                                               je       short BUTTON_REPORT
005B  EB 23                                               jmp      short EXIT_TOUCH
005D  B4 02                             POS_REPORT:       mov      ah,02H                 ;Move the cursor to the recieved position
005F  8A F1                                               mov      dh,cl                  ;using the standard IBM BIOS int 10.
0061  8A D3                                               mov      dl,bl
0063  B7 00                                               mov      bh,0
0065  CD 10                                               int      10H
0067  EB 17                                               jmp      short EXIT_TOUCH       ;That finishes that ISR.
0069  F6 C3 80                          BUTTON_REPORT:    test     bl,MAKE_BREAK_BIT      ;See if this is a touch or a release.
006C  74 0A                                               jz       short BUTTON_PUSH
006E  B5 0E                                               mov      ch,0EH                 ;On a release make the cursor back into
0070  B1 0F                                               mov      cl,0FH                 ;a line.
0072  B4 01                                               mov      ah,1
0074  CD 10                                               int      10H
0076  EB 08                                               jmp      short EXIT_TOUCH       ;That finishes a release ISR.
0078  B5 00                             BUTTON_PUSH:      mov      ch,0                   ;Make the cursor into a box on touch.
007A  B1 0F                                               mov      cl,0fh
007C  B4 01                                               mov      ah,1
007E  CD 10                                               int      10H
0080  61                                EXIT_TOUCH:       popa                            ;Restore all the registers.
0081  B4 00                                               mov      ah,RS_SUCCESSFUL       ;Set the return status.
0083  CF                                                  iret                            ;Return from the ISR
0084                                    TOUCH_HANDLER     endp
0084                                    TOUCH_RESTORE     proc
0084  B4 04                                               mov      ah,F_IO_CONTROL        ;Stop accepting calls
0086  B0 0C                                               mov      al,SF_EVENT_OFF
                                                          syscall V_LTOUCH
0088  BD 00C6                     +                                mov      bp,V_LTOUCH
008B  CD 6F                       +                                int      HP_ENTRY
008D  B4 04                                               mov      ah,F_IO_CONTROL        ;Restore the old event handler
008F  B0 08                                               mov      al,SF_CREATE_EVENT
0091  8B 1E 0000 R                                        mov      bx,word ptr SAVE_CS
0095  8E C3                                               mov      es,bx
0097  8D 36 0002 R                                        lea      si,word ptr SAVE_IP
009B  8B 16 0004 R                                        mov      dx,word ptr SAVE_DS
                                                          syscall V_LTOUCH
009F  BD 00C6                     +                                mov      bp,V_LTOUCH
00A2  CD 6F                       +                                int      HP_ENTRY
00A4  C3                                                  ret
00A5                                    TOUCH_RESTORE     endp
00A5                                    CODE_SEG          ends
                                                          end      BEGIN
```

Touch Example (cont.)

```
Macros:

                N a m e                         Length

SYSCALL. . . . . . . . . . . . . .              0002

Structures and records:

                N a m e                         Width    # fields
                                                Shift    Width    Mask     Initial

HP_SHEADER . . . . . . . . . . . . .            0010     0009
  DH_ATR . . . . . . . . . . . . . . .          0000
  DH_NAME_INDEX. . . . . . . . . . . .          0002
  DH_V_DEFAULT . . . . . . . . . . . .          0004
  DH_P_CLASS . . . . . . . . . . . . .          0006
  DH_C_CLASS . . . . . . . . . . . . .          0008
  DH_V_PARENT. . . . . . . . . . . . .          000A
  DH_V_CHILD . . . . . . . . . . . . .          000C
  DH_MAJOR . . . . . . . . . . . . . .          000E
  DH_MINOR . . . . . . . . . . . . . .          000F

Segments and Groups:

                N a m e                         Size     Align    Combine Class

CODE_SEG . . . . . . . . . . . . . .            00A5     PARA     NONE
DATA_SEG . . . . . . . . . . . . . .            00A8     PARA     NONE
TS_EVENT_HEADR . . . . . . . . . . .            0010     PARA     NONE

Symbols:

                N a m e                         Type     Value    Attr

ATR_HP . . . . . . . . . . . . . . .            Number   8000
BEGIN. . . . . . . . . . . . . . . .            L NEAR   0000     CODE_SEG
BUTTON_PUSH. . . . . . . . . . . . .            L NEAR   0078     CODE_SEG
BUTTON_REPORT. . . . . . . . . . . .            L NEAR   0069     CODE_SEG
CL_NULL. . . . . . . . . . . . . . .            Number   0000
EXAM_HP_ATTR . . . . . . . . . . . .            Alias    ATR_HP
EXIT_PROG. . . . . . . . . . . . . .            L NEAR   0018     CODE_SEG
EXIT_TOUCH . . . . . . . . . . . . .            L NEAR   0080     CODE_SEG
F_IO_CONTROL . . . . . . . . . . . .            Number   0004
F_ISR. . . . . . . . . . . . . . . .            Number   0000
HP_ENTRY . . . . . . . . . . . . . .            Number   006F
INPUT_LOOP . . . . . . . . . . . . .            L NEAR   000E     CODE_SEG
MAKE_BREAK_BIT . . . . . . . . . . .            Number   0080
POS_REPORT . . . . . . . . . . . . .            L NEAR   005D     CODE_SEG
PROCESS_ISR. . . . . . . . . . . . .            L NEAR   0050     CODE_SEG
READ_CHAR_ECHO . . . . . . . . . . .            Number   0001
RS_SUCCESSFUL. . . . . . . . . . . .            Number   0000
RS_UNSUPPORTED . . . . . . . . . . .            Number   0002
SAVE_CS. . . . . . . . . . . . . . .            L WORD   0000     DATA_SEG
SAVE_DS. . . . . . . . . . . . . . .            L WORD   0004     DATA_SEG
SAVE_IP. . . . . . . . . . . . . . .            L WORD   0002     DATA_SEG
SF_CREATE_EVENT. . . . . . . . . . .            Number   0008

SF_EVENT_OFF . . . . . . . . . . . .            Number   000C
SF_EVENT_ON. . . . . . . . . . . . .            Number   000A
STACK. . . . . . . . . . . . . . . .            L WORD   0006     DATA_SEG         Length =0050
STK_TOP. . . . . . . . . . . . . . .            L WORD   00A6     DATA_SEG
TERMINATE_PROC . . . . . . . . . . .            Number   004C
TOUCH_ENABLE . . . . . . . . . . . .            N PROC   001D     CODE_SEG         Length =002B
TOUCH_HANDLER. . . . . . . . . . . .            N PROC   0048     CODE_SEG         Length =003C
TOUCH_RESTORE. . . . . . . . . . . .            N PROC   0084     CODE_SEG         Length =0021
T_KC_BUTTON. . . . . . . . . . . . .            Number   0009
T_TS . . . . . . . . . . . . . . . .            Number   0045
V_DOLITTLE . . . . . . . . . . . . .            Number   0006
V_EVENT_TOUCH. . . . . . . . . . . .            Number   0060
V_LTOUCH . . . . . . . . . . . . . .            Number   00C6

48576 Bytes free

Warning Severe
Errors  Errors
0       0
```

4.3 Hardware Interface Level

The hardware interface of the Input System is composed of a set of drivers to respond to hardware interrupts and process physical data from the input devices into a form usable by the application interface drivers. These drivers are shown in Figure 4.2.

4.3.1 Overview

This section describes the drivers, data structures, and interrupt service routine (ISR) event processing that takes place below the application interface level. The following data flow expands on step 2 of the data flow presented in Section 4.2.1. A detailed explanation of each step is presented after the data flow.

1. The user touches the screen. This causes a hardware interrupt which is managed by the 8259A interrupt controller service (V__S8259). V__S8259 responds to the interrupt controller chip and transfers control to the HP-HIL driver.

2. The HP-HIL driver (V__HPHIL) services the HP-HIL controller chip, retrieving the input device data. V__HPHIL processes the input data and transfers control to the Input System dispatch service.

3. The dispatch service (V__SINPUT) transfers control to the appropriate physical device driver based on the source of the input data (in this case the physical touch screen driver).

4. The physical touch screen driver builds the Physical Describe Record and transfers control to the application interface driver V__LTOUCH.

V__S8259 provides a funnel point for managing HP specific hardware. The Input System hardware communicates with the hardware interface drivers via three interrupts: the 8041 service request (SVC), the 8041 Output Buffer Full (OBF), and the HP-HIL controller interrupt. The 8041 SVC and OBF interrupts are discussed in the keyboard section (Section 5). The HP-HIL controller interrupt is chained to the HP-HIL driver (V__HPHIL), i.e., when V__S8259 receives an HP-HIL controller interrupt it generates an HP__ENTRY software interrupt to transfer control to V__HPHIL.

The HP-HIL driver services the HP-HIL controller and generates the appropriate Physical ISR Event Record(s). After processing the input data V__HPHIL chains to V__SINPUT.

Hardware Interface Level Drivers

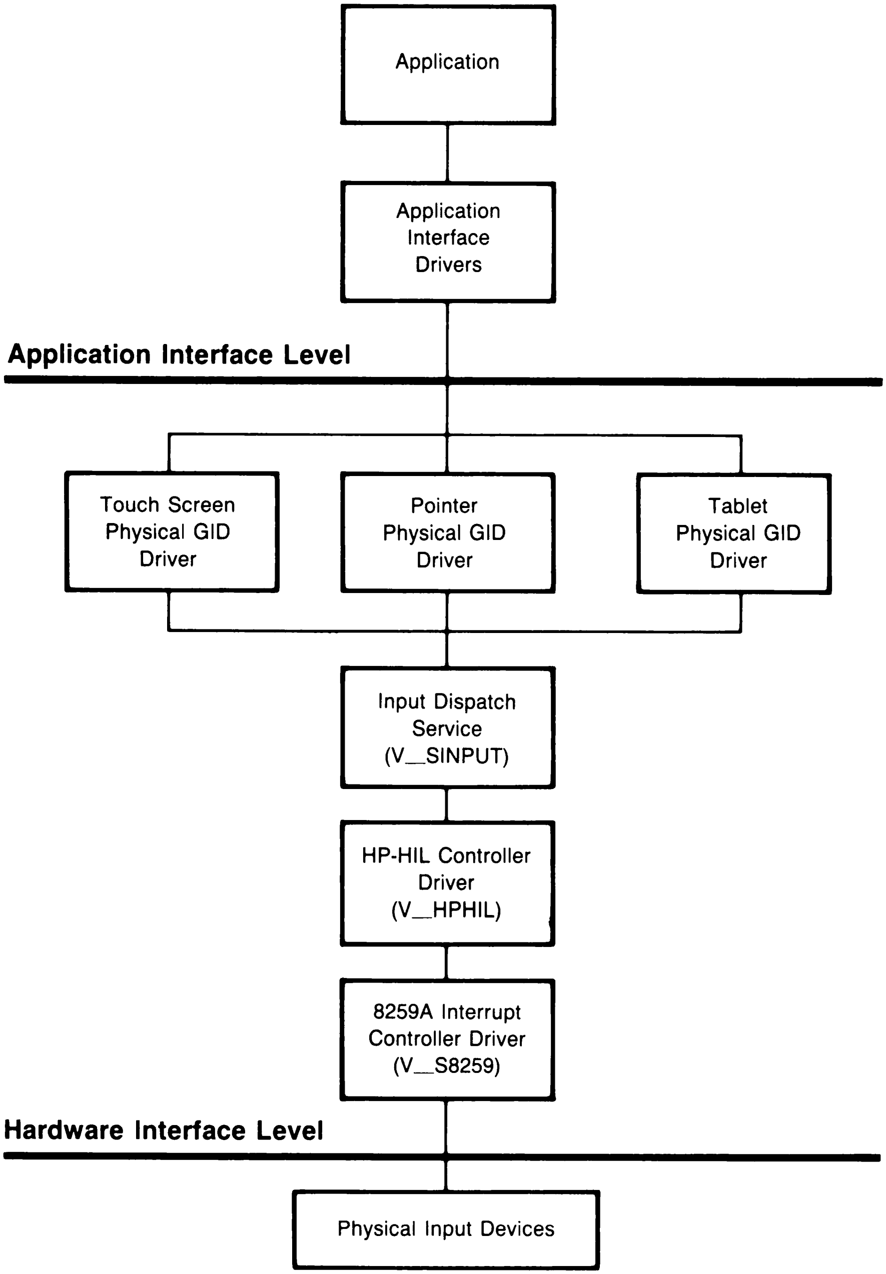

Application
Application Interface Drivers
Application Interface Level
Touch Screen Physical GID Driver
Pointer Physical GID Driver
Tablet Physical GID Driver
Input Dispatch Service (V__SINPUT)
HP-HIL Controller Driver (V__HPHIL)
8259A Interrupt Controller Driver (V__S8259)
Hardware Interface Level
Physical Input Devices

Figure 4.2

V__SINPUT chains to the appropriate physical device driver based on the vector index (vector address divided by six) stored in the Physical ISR Event Record (DL register). It provides an entry point into the Input System for non-HP-HIL devices. V__SINPUT also provides driver mapping functions that will be discussed later in this section.

Two physical drivers will be discussed later in this section. The first is the physical GID driver (PGID) which handles both absolute and relative data. Because PGID can handle both types of GID data, it can chain to any logical GID driver; this forms the basis for Input System device driver mapping. The second physical driver is the null device driver (V__PNULL), which serves as a handler for unsupported devices. The keyboard driver is discussed in Section 5.

4.3.1.1 Device Driver Mapping

Each driver in the Input System has a vector in the HP__VECTOR__TABLE, and a driver header. Each driver header has two fields which determine the mapping of the driver. One field contains the vector of the driver's parent driver and the other contains the vector of the driver's child driver. Refer to Section 2 and Appendix G for a detailed description of driver headers.

Calls are made to the vector address contained in the parent field to pass the interrupt on to the next driver in the device driver chain, moving the data from the hardware toward the application via the desired logical GID driver. Hardware commands from the application are passed down the device driver chain to the device via the vector address contained in the child vector field. By changing the value of the parent or child vector field, the sequence of drivers called to handle an interrupt or function request is changed. In general an application may re-map a driver by changing the driver header directly. Functions are provided by the V__SINPUT service to map the physical GID drivers to the logical GID drivers.

4.3.1.2 Device Emulation

Device emulation occurs when one or more physical devices are mapped to a logical device that does not represent the original source of the data. For example, mapping a physical mouse driver to a logical touch screen driver allows the mouse to look like a touch screen to the application. The key requirement for a logical device driver to emulate other devices is that it accept both absolute and relative data. Referencing the above example, the logical touch screen driver which reports absolute data must accept both absolute (touch) data and relative (mouse) data.

An example of device mapping and emulation occurring in the system is the translation of mouse input to Cursor Control Pad (CCP) input. Since standard DOS processes keyboard input only, (not mouse input), the physical GID driver which processes mouse input is mapped, in its default state, to a driver called V__PGID__CCP. This driver causes mouse input to emulate input from the CCP. For an application which processes industry standard mouse input (INT 33H) to use the HP Mouse, the mouse physical GID driver should be mapped to the V__LHPMOUSE driver using the F33__INSTALL function (see Section 6 for more details).

4.3.2 Data Structures

The hardware interface level uses two major data structures: the Physical Describe Record and the Physical ISR Event Record(s). These data structures help keep track of the numerous events occurring in the Input System.

4.3.2.1 Physical Describe Record

The Physical Describe Record is used by the physical GID drivers to keep track of the current state of their respective devices. Each of the physical GID drivers has a Physical Describe Record associated with it, which is located directly after the driver header starting with memory address DS:0010H. An explanation of the Physical Describe Record fields follows, table 4.9 contains the field types and offsets.

Table 4.9

Physical GID Device Describe Record

Field	Type	Offset	Description
Driver Header		00H	Driver header (see Section 2)
D__SOURCE	BYTE	10H	Input type and device address
D__HPHIL__ID	BYTE	11H	Device ID
D__DESC__MASK	BYTE	12H	Describe header byte
D__IO__MASK	BYTE	13H	Device I/O descriptor byte
D__XDESC__MASK	BYTE	14H	Extended describe header byte
D__MAX__AXIS	BYTE	15H	Maximum number of axes
D__CLASS	BYTE	16H	Device class
D__PROMPTS	BYTE	17H	Number of button/prompts
D__RESERVED	BYTE	18H	Reserved
D__BURST__LEN	BYTE	19H	Maximum output burst length
D__WR__REG	BYTE	1AH	Number of write registers
D__RD__REG	BYTE	1BH	Number of read registers
D__TRANSITION	BYTE	1CH	Button transitions
D__STATE	BYTE	1DH	Current state of the buttons
D__RESOLUTION	WORD	1EH	Physical device resolution
D__SIZE__X	WORD	20H	Maximum x-axis count
D__SIZE__Y	WORD	22H	Maximum y-axis count
D__ABS__X	WORD	24H	X position data for absolute devices
D__ABS__Y	WORD	26H	Y position data for absolute devices
D__REL__X	WORD	28H	X delta for relative devices
D__REL__Y	WORD	2AH	Y delta for relative devices
D__ACCUM__X	WORD	2CH	Reserved
D__ACCUM__Y	WORD	2EH	Reserved

D__SOURCE — This field is divided into nibbles. Bits 7-4 contain the graphics input device type. This field is loaded with the low order nibble of the appropriate physical GID data type. See table 4.12. Bits 3-0 are the link address of the physical device.

D__HPHIL__ID — ID byte of the physical device which last reported data. See table 4.2 for a list of HP-HIL ID bytes.

D__DESC__MASK — Physical device describe byte. This byte contains information about the physical device characteristics, see *HP-HIL Technical Reference Manual* for more information.

D_IO_MASK — Physical device I/O descriptor byte. This byte contains information on the number of prompts and acknowledges the device supports. See *HP-HIL Technical Reference Manual* for more information.

D_XDESC_MASK — Physical device extended describe byte. This byte contains additional device characteristics. See *HP-HIL Technical Reference Manual* for more information.

D_MAX_AXIS — Maximum number of axes supported by the device. Valid range is 0-2.

D_CLASS — Device class. Bits 7-4 contain the current class. Bits 3-0 contain the default class. See Appendix G for more information on device classes.

D_PROMPTS — Number of buttons and prompts supported by the device. Bits 7-4 is the number of prompts. Bits 3-0 is the number of buttons.

D_BURST_LEN — Maximum number of bytes that can be output to the device using a single write command.

D_WR_REG — Number of write registers supported by the device.

D_RD_REG — Number of read registers supported by the device.

D_TRANSITION — Transitions reported per button, i.e. a set bit indicates that the corresponding button was either pushed or released. Bit 7 corresponds to button 7 etc.

D_STATE — Current state of the buttons. 1 is down, 0 is up. Bit 7 corresponds to button 7 etc. If D_STATE is XOR'ed with D_TRANSITION the result is the previous button state.

D_RESOLUTION — This is the resolution of the physical device. The resolution is in counts per meter for devices that report 8 bits of data. For devices that report 16 bits of data the resolution is in counts per centimeter.

D_SIZE_X — Maximum count (in units of resolution) for the x-axis.

D_SIZE_Y — Maximum count (in units of resolution) for the y-axis.

D_ABS_X — X position data for devices which report absolute coordinates (absolute devices).

D_ABS_Y — Y position data for devices which report absolute coordinates.

D_REL_X Latest change in x position for devices which return coordinates relative to the previous position (relative devices).

D_REL_Y Latest change in y position for devices which return coordinates relative to the previous position.

4.3.2.2 Physical ISR Event Records

A Physical ISR Event Record is not a data structure in the truest sense, but is a set of register definitions for inter-driver communication of input events. Tables 4.10 and 4.11 define the Physical ISR Event Records.

Table 4.10

GID Button ISR Event Record

AH = F_ISR (00H)
DL = Physical device driver's vector address / 6
BX = Button information.

Bit	Value	Definition
0FH-08H	—	Reserved
07H	1	Button up
	0	Button down
06H-00H	—	Button number (0-7)

DH = Data Type

ES:0 = Pointer to physical device driver header and Physical Describe Record.

Table 4.11

GID Motion ISR Event Record

AH	=	F__ISR (00H)
DL	=	Physical device driver's vector address / 6
BX	=	X axis motion in raw data form.
CX	=	Y axis motion in raw data form.
DH	=	Data Type
ES:0	=	Pointer to physical device driver header and Physical Describe Record.

The button number in the Button Transition Information field (BX) denotes which button on the device is reporting data. Of special interest is button seven (proximity indicator) which is currently used by absolute devices to indicate that the device measurement field is active, ie. someone is touching the touch screen or the stylus is in contact with the tablet surface.

The Data Type field (DH) contains a code representing the current type of physical GID data stored in the event record. For button events this value will be T__KC__BUTTON. For a complete list of physical GID event data types see table 4.12.

Table 4.12

Physical GID Event Data Types

Type	Value	Definition
T__KC__BUTTON	09H	Button data.
T__REL08	40H	Signed 8 bit relative data
T__REL16	41H	Signed 16 bit relative data
T__ABS08	42H	Unsigned 8 bit absolute data
T__ABS16	43H	Unsigned 16 bit absolute data

4.3.3 Hardware Interface Level Drivers

This section describes the hardware interface level drivers in detail.

4.3.3.1 V__S8259 Driver (BP = 001EH)

The V__S8259 driver services the HP 8259A slave interrupt controller. Three interrupt request lines are connected to this controller; the 8041 SVC (Service port) service request, the HP-HIL controller, and the 8041 OBF (Output Buffer Full) service request.

When this driver is initialized, the interrupt vectors for the three interrupts listed above are set for their respective entry points into the V__S8259 driver. When an interrupt occurs, control is transferred to one of the three entry points. The V__S8259 driver will perform an F__ISR call to one of three drivers; the V__8041 driver for the 8041 SVC interrupt, the V__HPHIL driver for the HP-HIL controller interrupt, and the INT 09H driver for the 8041 OBF interrupt.

In the case of the 8041 SVC interrupt and the HP-HIL controller interrupt the corresponding interrupt is masked off on the HP slave controller and an End-of-Interrupt command is sent to the master interrupt controller before passing the interrupt on (via F__ISR). This allows other interrupts even of lower priority to be serviced on the HP slave 8259A but does not require interrupt handlers to be interrupt reentrant since the same interrupt is not allowed to fire until the entire driver chain has completed processing. When these two driver chains finish processing the V__S8259 issues a specific End-of-Interrupt command to the HP 8259A slave controller and then unmasks the corresponding interrupt so it can fire again.

In the case of the 8041 OBF interrupt a specific End-of-Interrupt is sent to the HP slave controller before passing on the interrupt, allowing the industry standard INT 09H driver to manage the master 8259A controller as if the HP slave controller were not present.

In addition to initiating response to the hardware interrupts, the 8259A driver contains other functions which initialize the interrupt vectors, and program the proper parameters into the HP 8259A slave interrupt controller.

V__S8259 Driver Function Definitions

A summary of the V__S8259 function codes is provided in table 4.13.

Table 4.13

V__S8259 Function Code Summary

Vector Address	Func. Value	Function Equate	Definition
001EH		V__S8259	8259 interrupt controller support
001EH	02	F__SYSTEM	System functions
001EH	02/00	SF__INIT	Initialize HP slave 8259A
001EH	02/02	SF__START	Enable HP slave 8259A interrupts
001EH	02/06	SF__VERSION__DESC	Report HP version number
001EH	04	F__IO__CONTROL	Entry point to I/O control functions
001EH	04/00	SF__ENABLE__SVC	Unmask svc/8041 interrupt
001EH	04/02	SF-DISABLE__SVC	Mask svc/8041 interrupt
001EH	04/04	SF__ENABLE__KBD	Unmask keyboard INT 9 interrupt
001EH	04/06	SF__DISABLE__KBD	Mask keyboard INT 9 interrupt
001EH	04/08	SF__ENABLE__HPHIL	Unmask HP-HIL interrupt
001EH	04/0A	SF__DISABLE__HPHIL	Mask HP-HIL interrupt

F__ISR (AH = 00H)

Because this driver directly services hardware interrupts from an 8259A interrupt controller, this function is *not applicable.* If called, this function will return a Return Status Code of RS__UNSUPPORTED.

SF__INIT (AX = 0200H)

This subfunction sets the interrupt vectors for the three HP 8259A slave interrupt sources to the appropriate entry points in the driver. In addition, the necessary 8259A parameters are programmed into the HP 8259A slave interrupt controller. This subfunction leaves interrupts disabled. They must be enabled with the SF__START subfunction.

On Entry: AH = F__SYSTEM (02H)
AL = SF__INIT (00H)
BP = V__S8259 (001EH)

On Exit: AH = Return Status Code

Registers Altered: AX, BP, DS

SF__START (AX = 0202H)

This subfunction enables the interrupts on the HP 8259A slave interrupt controller.

```
On Entry:  AH  = F__SYSTEM (02H)
           AL  = SF__START (02H)
           BP  = V__S8259 (001EH)

On Exit:   AH  = Return Status Code
```

Registers Altered: AX, BP, DS

SF__VERSION__DESC (AX = 0206H)

This subfunction returns the release date code and a double word pointer to the current version number. The date code consists of two BCD coded bytes containing the year and week of release. The BL register contains the number of years since 1960 and the BH register contains the week of the year.

```
On Entry:  AH  = F__SYSTEM (02H)
           AL  = SF__VERSION__DESC (06H)
           BP  = V__S8259 (001EH)

On Exit:   AH  = Return Status Code
           BX  = Release date code
           CX  = Number of bytes in current version number
        ES:DI  = Pointer to the current version number
```

Registers Altered: AX, BX, CX, DI, ES, BP, DS

SF__ENABLE__SVC (AX = 00400H)

This function unmasks (enables) the 8041 SVC interrupt on the HP 8259A slave controller.

```
On Entry:  AH  = F__IO__CONTROL (04H)
           AL  = SF__ENABLE__SVC (00H)
           BP  = V__S8259 (001EH)

On Exit:   AH  = Return Status Code
```

Registers Altered: AX, BP, DS

SF__DISABLE__SVC (AX = 0402H)

This function masks off (disables) the 8041 SVC interrupt on the HP 8259A slave controller.

On Entry: AH = F__IO__CONTROL (04H)
AL = SF__DISABLE__SVC (02H)
BP = V__S8259 (001EH)

On Exit: AH = Return Status Code

Registers Altered: AX, BP, DS

SF__ENABLE__KBD (AX = 0404H)

This function unmasks (enables) the 8041 OBF interrupt on the HP 8259A slave controller.

On Entry: AH = F__IO__CONTROL (04H)
AL = SF__ENABLE__KBD (04H)
BP = V__S8259 (001EH)

On Exit: AH = Return Status Code

Registers Altered: AX, BP, DS

SF__DISABLE__KBD (AX = 0406H)

This routine masks off (disables) the 8041 OBF interrupt on the HP 8259A slave controller.

On Entry: AH = F__IO__CONTROL (04H)
AL = SF__DISABLE__KBD (06H)
BP = V__S8259 (001EH)

On Exit: AH = Return Status Code

Registers Altered: AX, BP, DS

SF__ENABLE__HPHIL (AX = 0408H)

This routine unmasks (enables) the HP-HIL controller interrupt on the HP 8259A slave controller.

On Entry: AH = F__IO__CONTROL (04H)
AL = SF__ENABLE__HPHIL (08H)
BP = V__S8259 (001EH)

On Exit: AH = Return Status Code

Registers Altered: AX, BP, DS

SF__DISABLE__HPHIL (AX = 040AH)

This routine masks off (disables) the HP-HIL controller interrupt on the HP 8259A slave controller.

On Entry: AH = F__IO__CONTROL (04H)
AL = SF__DISABLE__HPHIL (0AH)
BP = V__S8259 (001EH)

On Exit: AH = Return Status Code

Registers Altered: AX, BP, DS

4.3.3.2 V__HPHIL Driver (BP = 0114H)

The HP-HIL driver retrieves input data from the HP-HIL controller and builds an ISR Event Record to pass to V__SINPUT.

A summary of the V__HPHIL function codes is provided in table 4.14.

Table 4.14

V__HPHIL Driver Function Code Summary

Vector	Func.	Function	
Address	Value	Equate	Definition
0114H		V__HPHIL	Setup HP-HIL to INPUT driver linkage
0114H	00	F__ISR	Logical Interrupt
0114H	02	F__SYSTEM	System Functions
0114H	02/00	SF__INIT	Initializes the driver data area.
0114H	02/04	SF__REPORT__STATE	Reports state of device
0114H	02/06	SF__VERSION__DESC	Reports driver version number.
0114H	02/0E	SF__OPEN	Put driver in open state.
0114H	02/10	SF__CLOSE	Put driver in open state.
0114H	04	F__IO__CONTROL	I/O control to driver
0114H	04/04	SF__CRV__CRV__MAJ__MIN	Reserved
0114H	04/06	SF__CRV__RECONFIGURE	Forces HP-HIL to reconfigure all devices.
0114H	04/08	SF__CRV__WR__PROMPTS	Write a prompt to a device
0114H	04/0A	SF__CRV__WR__ACK	Write an acknowledge to a device
0114H	04/0C	SF__CRV__REPEAT	Sets either 30Hz or 60Hz repeat rate
0114H	04/0E	SF__CRV__DISABLE__REPEAT	Cancels keyboard repeat rate
0114H	04/10	SF__CRV__SELF__TEST	Issue self-test command to physical device.
0114H	04/12	SF__CRV__REPORT__STATUS	Get status from any HP-HIL device that needs to report
0114H	04/14	SF__CRV__REPORT__NAME	Returns the ASCII name for a device
0114H	04/16	SF__KEYBOARD__REPEAT	Set typematic values
0114H	04/18	SF__KEYBOARD__LED	Sets keyboard LED states
0114H	06	F__PUT__BYTE	Write one byte to specified HP-HIL device.
0114H	08	F__GET__BYTE	Read one byte from specified HP-HIL device.
0114H	0A	F__PUT__BUFFER	Write a string of bytes to HP-HIL device.

V_HPHIL Driver Function Definitions

F_ISR (AH = 00H)

This function is called by the V_S8259 driver to initiate processing of an interrupt from the HP-HIL controller. This function reads input device data from the HP-HIL controller, generates one or more ISR Event Records, and chains to V_SINPUT. THIS FUNCTION SHOULD ONLY BE CALLED BY THE V_S8259 DRIVER.

On Entry: AH = F_ISR (00H)
BP = V_HPHIL (0114H)

On Exit: AH = Return Status Code

Registers Altered: AX, BP, DS

SF_INIT (AX = 0200H)

This subfunction initializes the driver and HP-HIL controller. Refer to Section 9 for a complete discussion of the protocol utilized in data space allocation (''last used DS'' passed in register BX).

On Entry: AH = F_SYSTEM (02H)
AL = SF_INIT (00H)
BX = ''Last used DS'' in HP Data Area
BP = V_HPHIL (0114H)

On Exit: AH = Return Status Code
BX = New ''last used DS'' in HP Data Area

Registers Altered: AX, BX, BP, DS

SF_REPORT_STATE (AX = 0204H)

This subfunction returns the current status of V_HPHIL.

On Entry: AH = F_SYSTEM (02H)
AL = SF_REPORT_STATE (04H)
BP = V_HPHIL (0114H)

On Exit: AH = Return Status Code
BX = Status word

Bit	Value	Definition
0FH-0DH	—	Reserved
0CH	1	Timeout has occurred
0BH	1	Output request has completed
0AH	—	Reserved
09H	1	Error during output request
08H	1	HP-HIL link has been reconfigured
07H	—	Reserved
06H	1	HP-HIL driver is open
	0	HP-HIL driver is closed
05H-04H	—	Reserved
03H	1	General failure
02H	1	No devices attached.
01H	—	Reserved
00H	1	Link configuration in progress

Registers Altered: AX, BX, BP, DS

SF__VERSION__DESC (AX = 0206H)

This subfunction returns the release date code and a double word pointer to the current version number. The date code consists of two BCD coded bytes containing the year and week of release. The BL register contains the number of years since 1960 and the BH register contains the week of the year.

On Entry: AH = F__SYSTEM (02H)
AL = SF__VERSION__DESC (06H)
BP = V__HPHIL (0114H)

On Exit: AH = Return Status Code
BX = Release date code
CX = Number of bytes in current version number
ES:DI = Pointer to the current version number

Registers Altered: AX, BX, CX, DI, ES, BP, DS

SF__OPEN (AX = 020EH)

This subfunction puts the HP-HIL driver in the open state. When the driver has been placed in the open state, output to the HP-HIL devices is allowed.

On Entry: AH = F__SYSTEM (02H)
AL = SF__OPEN (0EH)
BP = V__HPHIL (0114H)

On Exit: AH = Return Status Code

Registers Altered: AX, BP, DS

SF__CLOSE (AX = 0210H)

This subfunction puts the HP-HIL driver in the closed state. When the driver has been placed in the closed state, output to the HP-HIL devices is not allowed.

On Entry: AH = F__SYSTEM (02H)
AL = SF__CLOSE (10H)
BP = V__HPHIL (0114H)

On Exit: AH = Return Status Code

Registers Altered: AX, BP, DS

SF__CRV__RECONFIGURE (AX = 0406H)

This subfunction instructs the HP-HIL controller to reconfigure the link.

On Entry: AH = F__IO__CONTROL (04H)
AL = SF__CRV__RECONFIGURE (06H)
BP = V__HPHIL (0114H)

On Exit: AH = Return Status Code

Registers Altered: AX, BP, DS

SF__CRV__WR__PROMPTS (AX = 0408H)

This subfunction issues a prompt command to a device on the HP-HIL link. The prompt command is either specific (prompt number 1 - 7) or generic (a prompt number other than 1 - 7).

On Entry: AH = F__IO__CONTROL (04H)
AL = SF__CRV__WR__PROMPTS (08H)
BX = Device address indicator

Bit	Value	Definition
0FH-0EH	—	Reserved
0DH	1	Valid address is present in DH
	0	Reserved for future enhancement, currently returns RS__FAIL
0CH	1	Valid register is present in DL
0BH-00H	—	Reserved

DH = HP-HIL device address
DL = Prompt number
BP = V__HPHIL (0114H)

On Exit: AH = Return Status Code

Registers Altered: AX, BP, DS

SF__CRV__WR__ACK (AX = 040AH)

This subfunction issues an acknowledge command to a device on the HP-HIL link. The acknowledge command is either specific (acknowledge number 1 - 7) or generic (an acknowledge number other than 1 - 7).

On Entry: AH = F__IO__CONTROL (04H)
AL = SF__CRV__WR__ACK (0AH)
BX = Device address indicator

Bit	Value	Definition
0FH-0EH	—	Reserved
0DH	1	Valid address is present in DH
	0	Reserved for future enhancement, currently returns RS__FAIL
0CH	1	Valid register is present in DL
0BH-00H	—	Reserved

DH = HP-HIL device address (major address)
DL = Acknowledge number
BP = V__HPHIL (0114H)

On Exit: AH = Return Status Code

Registers Altered: AX, BP, DS

SF__CRV__REPEAT (AX = 040CH)

This subfunction sets the key repeat rate of a specific HP-HIL device. A repeat rate of 30 or 60 times a second may be specified. This subfunction will only operate if the HP-HIL driver is in the open state.

On Entry: AH = F__IO__CONTROL (04H)
AL = SF__CRV__REPEAT (0CH)
BX = Device address indicator

Bit	Value	Definition
0FH-0EH	—	Reserved.
0DH	1	Valid address is present in DH.
	0	Reserved for future enhancement, currently returns RS__FAIL.
0CH	1	Valid register is present in DL.
0BH-00H	—	Reserved.

CL = 0 for a repeat rate of 30 Hz, 1 for 60 Hz
DH = HP-HIL device address (major address)
BP = V__HPHIL (0114H)

On Exit: AH = Return Status Code

Registers Altered: AX, BP, DS

SF__CRV__DISABLE__REPEAT (AX = 040EH)

This subfunction disables the key repeat of a specified HP-HIL device. This subfunction will only operate if the HP-HIL driver is in the open state.

On Entry: AH = F__IO__CONTROL (04H)
AL = SF__CRV__DISABLE__REPEAT (0EH)
BX = Device address indicator

Bit	Value	Definition
0FH-0EH	—	Reserved
0DH	1	Valid address is present in DH.
	0	Reserved for future enhancement, currently returns RS__FAIL.
0CH	1	Valid register is present in DL.
0BH-00H	—	Reserved

DH = HP-HIL device address (major address)
BP = V__HPHIL (0114H)

On Exit: AH = Return Status Code

Registers Altered: AX, BP, DS

SF__CRV__SELF__TEST (AX = 0410H)

This subfunction initiates device self-test on the specified HP-HIL device. The HP-HIL device will respond with a one byte status code indicating the result of the test. This subfunction should not be called with an HP-HIL device address of zero (all devices), as the test could then take up to 1.5 seconds to execute. Also, if one of the devices fails, there would be no way to determine which device reported a failure.

On exit the buffer has the return status of the self-test done on the physical device.

On Entry: AH = F__IO__CONTROL (04H)
AL = SF__CRV__SELF__TEST (10H)
BX = Device address indicator

Bit	Value	Definition
0FH-0EH	—	Reserved
0DH	1	Valid address is present in DH
	0	Reserved for future enhancement, currently returns RS__FAIL
0CH	1	Valid register is present in DL
0BH-00H	—	Reserved

DH = HP-HIL device address (major address)
BP = V__HPHIL (0114H)
ES:SI = Pointer to a buffer area

On Exit: AH = Return Status Code
ES:SI = Pointer to buffer area
CX = Number of bytes in buffer

Registers Altered: AX, CX, BP, DS

SF__CRV__REPORT__STATUS (AX = 0412H)

This subfunction issues a send status command to a specified HP-HIL device. The returned status information ranges from 1 to 15 bytes in length. A pointer to a 15 byte buffer must be passed to the subfunction. This subfunction will only operate if the HP-HIL driver is in the open state.

On Entry: AH = F__IO__CONTROL (04H)
AL = SF__CRV__REPORT__STATUS (12H)
BX = Device address indicator

Bit	Value	Definition
0FH-0EH	—	Reserved
0DH	1	Valid address is present in DH.
	0	Reserved for future enhancement, currently returns RS__FAIL.
0CH	1	Valid register is present in DL.
0BH-00H	—	Reserved

DH = HP-HIL device address (major address)
BP = V__HPHIL (0114H)
ES:SI = Pointer to a buffer area

On Exit: AH = Return Status Code
ES:SI = Pointer to buffer area
CX = Number of bytes in buffer

Registers Altered: AX, CX, BP, DS

SF__CRV__REPORT__NAME (AX = 0414H)

This subfunction issues a report name command to a specified HP-HIL device. The returned name information ranges from 1 to 15 bytes in length. A pointer to a 15 byte buffer must be passed to the subfunction. This subfunction will only operate if the HP-HIL driver is in the open state.

On Entry: AH = F__IO__CONTROL (04H)
AL = SF__CRV__REPORT__NAME (14H)
BX = Device address indicator

Bit	Value	Definition
0FH-0EH	—	Reserved
0DH	1	Valid address is present in DH.
	0	Reserved for future enhancement, currently returns RS__FAIL.
0CH	1	Valid register is present in DL.
0BH-00H	—	Reserved

DH = HP-HIL device address (major address)
BP = V__HPHIL (0114H)
ES:SI = Pointer to a buffer area

On Exit: AH = Return Status Code
ES:SI = Pointer to buffer area
CX = Number of bytes in buffer

Registers Altered: AX, CX, BP, DS

SF__KEYBOARD__REPEAT (AX = 0416H)

This subfunction sets the typematic rate and delay values for the keyboard. The Cursor Control keypad (CCP) may be set independent of the rest of the keyboard, i.e. the CCP may start repeating and repeat at different rates from the rest of the keyboard. See Section 5 for more information.

On Entry: AH = F__IO__CONTROL (04H)
AL = SF__KEYBOARD__REPEAT (16H)
BH = If BH = 0 set the typematic rate only, if BH = 1 set the delay only, if BH = 2 set both values.
BL = If BL = 0 the typematic rate and delay values are for the non-CCP keypads, if BL = 1 the values are for the Cursor Control keypad only.
DL = Bits 0-3 contain the typematic rate, Bits 4-7 contain the delay value. See Section 5, function F16__DEF__ATTR for permissable values.
BP = V__HPHIL (0114H)

On Exit: AH = Return Status Code

Registers Altered: AX, BP, DS

SF__KEYBOARD__LED (AX = 0418H)

This subfunction controls the state of three keyboard LED indicators. See Section 5 for more information.

If back to back calls to this function are made, only the most current value will be written to the keyboard device.

On Entry: AH = F__IO__CONTROL (04H)
AL = SF__KEYBOARD__LED (18H)
BL = Bit mask

Bit	Value	Definition
07H-03H	—	Reserved
02H	1	Turn on Caps lock LED
	0	Turn off Caps lock LED
01H	1	Turn on Num lock LED
	0	Turn off Num lock LED
00H	1	Turn on Scroll lock LED
	0	Turn off Scroll lock LED

BP = V__HPHIL (0114H)

On Exit: AH = Return Status Code

Registers Altered: AX, BP, DS

F__PUT__BYTE (AH = 06H)

This function outputs a byte of data to a specific HP-HIL device register. This function will only operate if the HP-HIL driver is in the open state.

On Entry: AH = F_PUT_BYTE (06H)
AL = Byte to output
BX = Device address indicator

Bit	Value	Definition
0FH-0EH	—	Reserved
0DH	1	Valid address is present in DH
	0	Reserved for future enhancement, currently returns RS_FAIL
0CH	1	Valid register is present in DL
0BH-00H	—	Reserved

DH = HP-HIL device address
DL = HP-HIL device register (0-127)
BP = V_HPHIL (0114H)

On Exit: AH = Return Status Code

Registers Altered: AX, BP, DS

F_GET_BYTE (AH = 08H)

This function returns the contents of a specific HP-HIL device register. This function will only operate if the HP-HIL driver is in the open state.

On Entry: AH = F_GET_BYTE (08H)
BX = Device address indicator

Bit	Value	Definition
0FH-0EH	—	Reserve Value Definition
0FH-0EH	—	Reserved
0DH	1	Valid address is present in DH
	0	Reserved for future enhancement, currently returns RS_FAIL
0CH	1	Valid register is present in DL
0BH-00H	—	Reserved

DH = HP-HIL device address
DL = HP-HIL device register (0-127)
BP = V_HPHIL (0114H)

On Exit: AH = Return Status Code
AL = Contents of specified register

Registers Altered: AX, BP, DS

F__PUT__BUFFER (AH = 0AH)

This function outputs a buffer to a specific HP-HIL device register. The HP-HIL controller and devices are capable of data transfer at rates up to 6500 bytes per second. If the number of bytes in the buffer is greater than the number the HP-HIL device can handle, this function will transfer as many bytes as possible to the device, and adjust the value in CX to reflect the number of bytes left in the buffer (not sent to the device).

On Entry: AH = F__PUT__BUFFER (0AH)
BX = Device address indicator

Bit	Value	Definition
0FH-0EH	—	Reserved
0DH	1	Valid address is present in DH
	0	Reserved for future enhancement, currently returns RS__FAIL
0CH	1	Valid register is present in DL
0BH-00H	—	Reserved

CX = Number of bytes in buffer
DH = HP-HIL device address
DL = HP-HIL device register (0-127)
BP = V__HPHIL (0114H)
ES:SI = Pointer to buffer containing data to output

On Exit: AH = Return Status Code
CX = 0 means all the data in buffer is transferred, otherwise the number of bytes left in buffer.

Registers Altered: AX, CX, BP, DS

4.3.3.3 V__SINPUT (BP = 002AH)

The V__SINPUT driver dispatches ISR events generated by the HP-HIL controller to the appropriate physical driver, thus providing an entry point into the Input System for non-HP-HIL devices (i.e., RS-232 mice, tablets, etc.). It also provides a number of functions which support device mapping.

A summary of the V_SINPUT function codes is provided in table 4.15.

Table 4.15

V_SINPUT Driver Function Code Summary

Vector Address	Func. Value	Function Equate	Definition
002AH		V_SINPUT	Inquire Commands
002AH	00	F_ISR	Pass ISR event record to physical driver
002AH	02/	F_SYSTEM	System Functions
002AH	02/00	SF_INIT	Initialize driver
002AH	04	F_IO_CONTROL	Entry point to IO control functions
002AH	04/00	SF_DEF_LINKS	Set header link fields to system defaults
002AH	04/02	SF_GET_LINKS	Return device header link field entries
002AH	04/04	SF_SET_LINKS	Set device header link field entries
002AH	06	F_INQUIRE	Return describe record for an HP-HIL device.
002AH	08	F_INQUIRE_ALL	Return device IDs for all HP-HIL devices present
002AH	0A	F_INQUIRE_FIRST	Return vector address of first HP-HIL device driver.
002AH	0C	F_REPORT_ENTRY	Report entry point of PGID

V_SINPUT Driver Function Definitions

F_ISR (AH = 00H)

This function passes an ISR Event Record to the appropriate physical device driver based on the value in DL. Non-HP-HIL devices which call V_SINPUT must provide the physical device driver that will handle the ISR event record, and must place its vector index (vector address divided by six) in DL. (See Section 9, V_SYSTEM functions, to obtain a valid vector address).

On Entry: AH = F_ISR (00H)
BP = V_SINPUT
(See tables 4.10 and 4.11 for other register values)

On Exit: AH = Return Status Code

Registers Altered: AX, BP, DS

SF_INIT (AX = 0200H)

This subfunction initializes the driver.

On Entry: AH = F_SYSTEM (02H)
AL = SF_INIT (00H)
BP = V_SINPUT (002AH)

On Exit: AH = Return Status Code

Registers Altered: AX, BP, DS

SF_DEF_LINKS (AX = 0400H)

This subfunction sets the parent vectors in the HP-HIL physical device driver headers to their system defaults. The defaults are shown in table 4.16. The child vector entries are set to the null device driver (V_PNULL) by default (see Appendix F).

Table 4.16

Default Physical Device Driver Parents

Device	Parent
Keyboard	V_8041
Mouse	V_PGID_CCP
Tablet	V_LTABLET
Touch Screen	V_LTOUCH
Barcode Reader	V_PNULL
Rotary Knob	V_PGID_CCP

On Entry: AH = F_IO_CONTROL (04H)
AL = SF_DEF_LINKS (00H)
BP = V_SINPUT (002AH)

On Exit: AH = Return Status Code

Registers Altered: AX, BP, DS

SF__GET__LINKS (AX = 0402H)

This subfunction returns the current parent and child vectors in the HP-HIL physical device driver headers. The address of a seven word (14 byte) table is passed to the subfunction. When the subfunction returns, the buffer will contain the current vectors. See table 4.17 for the buffer format.

Table 4.17

Mapping Buffer Format

Word	Parent Vector	Child Vector	HP-HIL Device
0	High byte	Low byte	Device # 1
1	" "	" "	" " 2
2	" "	" "	" " 3
3	" "	" "	" " 4
4	" "	" "	" " 5
5	" "	" "	" " 6
6	" "	" "	" " 7

On Entry: AH = F__IO__CONTROL (04H)
AL = SF__GET__LINKS (02H)
BP = V__SINPUT (002AH)
ES:SI = Pointer to table

On Exit: AH = Return Status Code
ES:SI = Pointer to table

Registers Altered: AX, BP, DS

SF__SET__LINKS (AX = 0404H)

This subfunction sets the parent and child vectors in the HP-HIL physical device driver headers. The address of a seven word (14 byte) table is passed to the subfunction. The table contains the new parent and child vectors for the drivers. The format of the buffer is shown in table 4.17.

```
On Entry:  AH  = F_IO_CONTROL (04H)
           AL  = SF_SET_LINKS (04H)
           BP  = V_SINPUT (002AH)
        ES:SI  = Pointer to table

On Exit:   AH  = Return Status Code
```

Registers Altered: AX, BP, DS

The following example is how to use the SF_SET_LINKS function. It is presumed that a call to F_INQUIRE_ALL has been made, and that the device at address #3, is a tablet. The tablet is going to be mapped to the V_LHPMOUSE driver. The BX register already has the offset into the buffer of tablet mappings.

```
BUFFER     DW   7 DUP (?)

       MOV      CX, BUFFER[BX]          ; get the current mapping of
                                        ; the tablet
       MOV      CH, V_LHPMOUSE / 6      ; change tablet to HP Mouse
       MOV      BUFFER[BX], CX          ; save the new mapping
       MOV      AH, F_IO_CONTROL        ; load function code
       MOV      AL, SF_SET_LINKS        ; load subfunction code
       MOV      BP, V_SINPUT            ; load vector address
       LEA      SI, BUFFER              ; get the offset of the buffer
       PUSH     DS
       POP      ES                      ; ES = DS
       PUSH     DS                      ; save current DS
       INT      HP_ENTRY                ; call extended BIOS driver
       POP      DS
```

F_INQUIRE (AH = 06H)

This function returns a pointer to the Physical Describe Record of the specified HP-HIL physical device driver. WARNING: THE PHYSICAL DESCRIBE RECORD SHOULD NOT BE MODIFIED IN ANY WAY.

```
On Entry:  AH  = F_INQUIRE (06H)
           AL  = HP-HIL Device Number (1 – 7)
           BP  = V_SINPUT (002AH)

On Exit:   AH  = Return Status Code
        ES:SI  = Pointer to Physical Describe Record
```

Registers Altered: AX, BP, SI, DS, ES

F__INQUIRE__ALL (AH = 08H)

This subfunction is used to determine which HP-HIL devices are present on the loop. The address of a seven word table is passed to the subfunction. When the subfunction returns, the table will contain the current status of all HP-HIL devices. The format of the buffer is shown in table 4.18.

Table 4.18

Device Inquire Buffer Format

Word	HP-HIL Device ID	Device Status*	HP-HIL Device
0	High byte	Low byte	Device # 1
1	" "	" "	" " 2
2	" "	" "	" " 3
3	" "	" "	" " 4
4	" "	" "	" " 5
5	" "	" "	" " 6
6	" "	" "	" " 7

* Bit 0 = 1 if device present, 0 if no device at this address.
Bits 2 – 7 are reserved.

On Entry: AH = F__INQUIRE__ALL (08H)
BP = V__SINPUT (002AH)
ES:SI = Pointer to table

On Exit: AH = Return Status Code
ES:SI = Pointer to table

Registers Altered: AX, BP, DS

The following example shows how to use the F_INQUIRE_ALL function.

```
BUFFER    DW   7 DUP (?)
     MOV       AH, F_INQUIRE_ALL        ; load function code
     LEA       SI, BUFFER               ; get offset of buffer
     PUSH      DS
     POP       ES                       ; ES = DS
     PUSH      DS                       ; save current DS
     INT  HP_ENTRY                      ; call EX-BIOS driver
     POP  DS                            ; restore DS
```

F_INQUIRE_FIRST (AH = 0AH)

This function returns the vector address of the first HP-HIL physical device driver (HP-HIL address 1). This address allows the vector address of all HP-HIL physical device drivers to be easily calculated since the vectors are contiguous in the HP_VECTOR_TABLE (see table 4.19).

On Entry: AH = F_INQUIRE_FIRST (0AH)
BP = V_SINPUT (002AH)

On Exit: AH = Return Status Code
BX = Vector address of first HP-HIL physical device driver

Registers Altered: AX, BX, BP, DS

F_REPORT_ENTRY (AH = 0CH)

This function is used to get the CS:IP of the physical GID driver.

On Entry: AH = F_REPORT_ENTRY (0CH)
BP = V_SINPUT (002AH)

On Exit: AH = Return Status Code
BX = offset of physical GID driver
ES = segment of physical GID driver

Registers Altered: AX, BX, BP, DS, ES

4.3.3.4 Physical GID Driver

The physical GID driver is responsible for updating the Physical Describe Record. Two types of graphics input devices are defined in the input system, absolute (touch screen and tablet), and relative (mouse). An instance of this driver (same code module, different data area) is installed for each graphic input device present.

A summary of the PGID function codes is provided in table 4.19.

Table 4.19

Physical GID Driver Function Code Summary

Vector Address	Func. Value	Function Equate	Definition
xxxH		HP-HIL driver vector 1 through HP-HIL driver vector 7.	Physical HP-HIL driver vectors (these vectors do not have fixed HP__VECTOR__TABLE addresses)
	00	F__ISR	Logical Interrupt
	02	F__SYSTEM	System functions
	02/00	SF__INIT	Initialize driver
	02/02	SF__START	Start driver
	02/04	SF__REPORT__STATE	Unsupported
	02/06	SF__VERSION__DESC	Report HP version number

Physical GID Driver Function Definitions

F__ISR (AH = 00H)

This function processes ISR Event Records, updates the fields in its Physical Describe Record, then calls its parent driver. HP-HIL devices report upward relative motion with a positive sign and downward relative motion with a negative sign. The industry standard representation is the opposite of this.

On Entry: AH = F__ISR (00H)
DH = Data Type
DL = Physical device driver's vector address / 6
BP = HP-HIL device n vector address

For Button Event:

BX = Button information.

Bit	Value	Definition
0FH-08H	——	Reserved
07H	1	Button up
	0	Button down
06H-00H	——	Button number (0-7)

For Motion Event:

BX = X axis motion in raw data form.
CX = Y axis motion in raw data form.

On Exit: AH = Return Status Code

Registers Altered: AX, BP, DS

SF__INIT (AX = 0200H)

This subfunction is called to initialize the driver.

On Entry: AH = F__SYSTEM (02H)
AL = SF__INIT (00H)
BP = HP-HIL device n vector address

On Exit: AH = Return Status Code

Registers Altered: AX, BP, DS

SF_START (AX = 0202H)

This subfunction starts the driver.

On Entry: AH = F_SYSTEM (02H)
AL = SF_START (02H)
BP = HP-HIL device n vector address

On Exit: AH = Return Status Code

Registers Altered: AX, BP, DS

SF_VERSION_DESC (AX = 0206H)

This subfunction returns the release date code and a double word pointer to the current version number. The date code consists of two BCD coded bytes containing the year and week of release. The BL register contains the number of years since 1960 and the BH register contains the week of the year.

On Entry: AH = F_SYSTEM (02H)
AL = SF_VERSION_DESC (06H)
BP = HP-HIL device n vector address

On Exit: AH = Return status code
BX = Release date code
CX = Number of byte in current version number
ES:DI = Pointer to the current version number

Registers Altered: AX, BX, CX, DI, ES, BP, DS

4.3.3.5 V_PNULL Driver (BP = 000CH)

The null device driver is the default event driver routine. It is used when the physical device is not recognized or the user event handler is not installed. It sets the AH register to RS_SUCCESSFUL and does an IRET.

4.3.4 Hardware Interface Level Services

Service drivers are provided as useful subroutines available to any driver. Currently the hardware interface level has only one service, the tracking sprite, V__STRACK. (For more information on sprites see Section 6).

4.3.4.1 V__STRACK Driver (BP = 0005AH)

V__STRACK is called by the logical GID drivers to move the graphics cursor (sprite) on the display screen. V__STRACK provides functions that allow the parameters of the sprite to be defined, and move the sprite around the display.

A summary of the V__STRACK function codes is provided in table 4.20.

Table 4.20

V__STRACK Driver Function Code Summary

Vector Address	Func. Value	Function Equate	Definition
005AH		V__STRACK	Sprite control
005AH	02	F__SYSTEM	System functions
005AH	02/00	SF__INIT	Initialize driver
005AH	02/02	SF__START	Start driver
005AH	04	F__TRACK__INIT	Sets tracking to default state
005AH	06	F__TRACK__ON	Enables tracking
005AH	08	F__TRACK__OFF	Disables tracking
005AH	0A	F__DEF__MASKS	Define sprite masks
005AH	0C	F__SET__LIMITS__X	Set max/min horizontal values
005AH	0E	F__SET__LIMITS__Y	Set max/min vertical values
005AH	10	F__PUT__SPRITE	Display sprite
005AH	12	F__REMOVE__SPRITE	Remove sprite from display

V__STACK Driver Function Definitions

F__ISR (AH = 00H)

This function is called to move the sprite to a new location. The display under the sprite is restored, and the sprite is redisplayed in its new location. The hot spot of the sprite is placed at the coordinates passed in BX and CX.

On Entry: AH = F__ISR (00H)
BX = X coordinate of sprite
CX = Y coordinate of sprite
DL = Source vector index
BP = V__STRACK (005AH)

On Exit: AH = Return Status Code

Registers Altered: AX, BP, DS

SF__INIT (AX = 0200H)

This subfunction is called to initialize the driver. Refer to Section 9 for a complete discussion of the protocol utilized in data space allocation (''last used DS'' passed in register BX).

On Entry: AH = F__SYSTEM (02H)
AL = SF__INIT (00H)
BX = ''Last used DS'' in HP Data Area
BP = V__STRACK (005AH)

On Exit: AH = Return Status Code
BX = New ''last used DS'' in HP Data Area

Registers Altered: AX, BX, BP, DS

SF__START (AX = 0202H)

This subfunction is called to start the tracking driver.

On Entry: AH = F__SYSTEM (02H)
AL = SF__START (02H)
BP = V__STRACK (005AH)

On Exit: AH = Return Status Code

Registers Altered: AX, BP, DS

F_TRACK_INIT (AH = 04H)

This function sets the tracking driver to its default state. It determines the current video mode, and initializes the tracking parameters.

On Entry: AH = F_TRACK_INIT (04H)
BP = V_STRACK (005AH)

On Exit: AH = Return Status Code

Registers Altered: AX, BP, DS

F_TRACK_ON (AH = 06H)

This function enables tracking. The sprite is displayed on the screen.

On Entry: AH = F_TRACK_ON (06H)
BP = V_STRACK (005AH)

On Exit: AH = Return Status Code

Registers Altered: AX, BP, DS

F_TRACK_OFF (AH = 08H)

This function disables tracking. The sprite is removed from the screen.

On Entry: AH = F_TRACK_OFF (08H)
BP = V_STRACK (005AH)

On Exit: AH = Return Status Code

Registers Altered: AX, BP, DS

F__DEF__MASKS (AH = 0AH)

This function is called to define the sprite and screen masks used by the driver. If tracking is enabled, the sprite is erased and the new sprite is displayed in its place. The size of the sprite (its width in bytes multiplied by its height) is limited to a total of 144 bytes. The width of the save area is one byte greater than the width of the sprite.

```
On Entry:  AH  = F__DEF__MASKS (0AH)
           BH  = Width of the save area (in bytes)
           BL  = Hot Spot X coordinate
           CH  = Height of sprite (in scan lines)
           CL  = Hot Spot Y coordinate
           BP  = V__STRACK (005AH)
        ES:SI  = Pointer to sprite mask

On Exit:   AH  = Return Status Code
```

Registers Altered: AX, BP, DS

The following example shows how to use the F__DEF__MASKS function provided by the tracking driver.

```
SPRITE   DW    0F9FFH    ; 1111100111111111    "*" marks the
         DW    0F0FFH    ; 11110*0011111111        Hot Spot
         DW    0E07FH    ; 1110000001111111
         DW    0E07FH    ; 1110000001111111
         DW    0C03FH    ; 1100000000111111
         DW    0C03FH    ; 1100000000111111
         DW    0801FH    ; 1000000000011111
         DW    0801FH    ; 1000000000011111
         DW    0000FH    ; 0000000000001111
         DW    0000FH    ; 0000000000001111
         DW    0F0FFH    ; 1111000011111111
         DW    0F0FFH    ; 1111000011111111
         DW    0F0FFH    ; 1111000011111111
         DW    0F0FFH    ; 1111000011111111
         DW    0F0FFH    ; 1111000011111111
         DW    0F0FFH    ; 1111000011111111

 Define the XOR mask
```

```
          DW     00000H    ; 0000000000000000    "*" marks the
          DW     00600H    ; 00000*1000000000        Hot Spot
          DW     00F00H    ; 0000111100000000
          DW     00F00H    ; 0000111100000000
          DW     01F80H    ; 0001111110000000
          DW     01F80H    ; 0001111110000000
          DW     03FC0H    ; 0011111111000000
          DW     03FC0H    ; 0011111111000000
          DW     07FE0H    ; 0111111111100000
          DW     00600H    ; 0000011000000000
          DW     00600H    ; 0000011000000000
          DW     00600H    ; 0000011000000000
          DW     00600H    ; 0000011000000000
          DW     00600H    ; 0000011000000000
          DW     00600H    ; 0000011000000000
          DW     00000H    ; 0000000000000000

     MOV  AH, F__DEF__MASKS       ; load function code
     LEA  SI, SPRITE              ; get the offset of the sprite
     PUSH DS
     POP  ES                      ; ES = DS of sprite
     MOV  CH, 10H                 ; height of sprite
     MOV  BH, 3                   ; number of bytes wide the
                                  ; save area is
     MOV  BL, 5                   ; hot spot x
     MOV  CL, 1                   ; hot spot y
     MOV  BP, V__STRACK           ; load vector address
     PUSH DS                      ; save current DS
     INT  HP__ENTRY               ; call EX-BIOS DRIVER
     POP  DS                      ; restore DS
```

F__SET__LIMITS__X (AH = 0CH)

This function sets the minimum and maximum horizontal position of the sprite on the screen. The default minimum and maximum values are the same as the current screen mode.

On Entry: AH = F__SET__LIMITS__X (0CH)
CX = Minimum X coordinate
DX = Maximum X coordinate
BP = V__STRACK (005AH)

On Exit: AH = Return Status Code

Registers Altered: AX, BP, DS

F__SET__LIMITS__Y (AH = 0EH)

This function sets the minimum and maximum vertical position of the sprite on the screen. The default minimum and maximum values are the same as the current screen mode.

On Entry: AH = F__SET__LIMITS__Y (0EH)
CX = Minimum Y coordinate
DX = Maximum Y coordinate
BP = V__STRACK (005AH)

On Exit: AH = Return Status Code

Registers Altered: AX, BP, DS

F__PUT__SPRITE (AH = 10H)

This function is called to put the sprite on the display.

On Entry: AH = F__PUT__SPRITE (10H)
BX = X coordinate of sprite
CX = Y coordinate of sprite
BP = V__STRACK (005AH)

On Exit: AH = Return Status Code

Registers Altered: AX, BP, DS

F__REMOVE__SPRITE (AH = 12H)

This function removes the sprite from the display.

On Entry: AH = F__REMOVE__SPRITE (12H)
BP = V__STRACK (005AH)

On Exit: AH = Return Status Code.

Registers Altered: AX, BP, DS

SECTION 5

TABLE OF CONTENTS

SECTION 5. KEYBOARD

5.1 Overview

The Keyboard Input System consists of four components: The input device drivers, STD-BIOS keyboard drivers, 8041 keyboard controller chip and the EX-BIOS keyboard drivers. The input device drivers are discussed in Section 4. The other three components are discussed in this section (See figure 5.1).

The industry standard INT 16H and INT 09H handlers make up the STD-BIOS keyboard drivers. INT 16H is used by applications to get characters from the keyboard buffer. INT 09H responds to interrupts from the 8041 controller and places characters in the keyboard buffer.

The 8041 controller chip provides an industry standard hardware interface to the INT 09H driver. It also provides timers and other services to the Keyboard Input System.

The EX-BIOS drivers translate HP-HIL keyboard scancodes to industry standard scancodes. They also allow applications to redefine the scancodes generated by certain groups of keys on the keyboard (keypads).

The following data flow describes the actions that occur when a user presses a key until it is read by an application:

1. When a key is pressed on the keyboard, the input device driver V__HPHIL creates an ISR event and chains to the input device driver V__SINPUT. The input device driver V__SINPUT chains to the EX-BIOS logical keyboard driver.

2. The EX-BIOS logical keyboard driver determines which keypad the scancode is from and calls the appropriate translator service. After translation, the logical keyboard driver chains to the 8041 interface driver.

3. The 8041 interface driver (V__8041) sends the scancode to the 8041 controller chip. The 8041 controller generates an Output Buffer Full (OBF) interrupt to notify the STD-BIOS INT 09H driver that a scancode is available.

4. The STD-BIOS INT 09H driver reads the scancode from the 8041 chip. The scancode is placed in the STD-BIOS keyboard buffer along with its associated ASCII character (keycode).

Keyboard Block Diagram

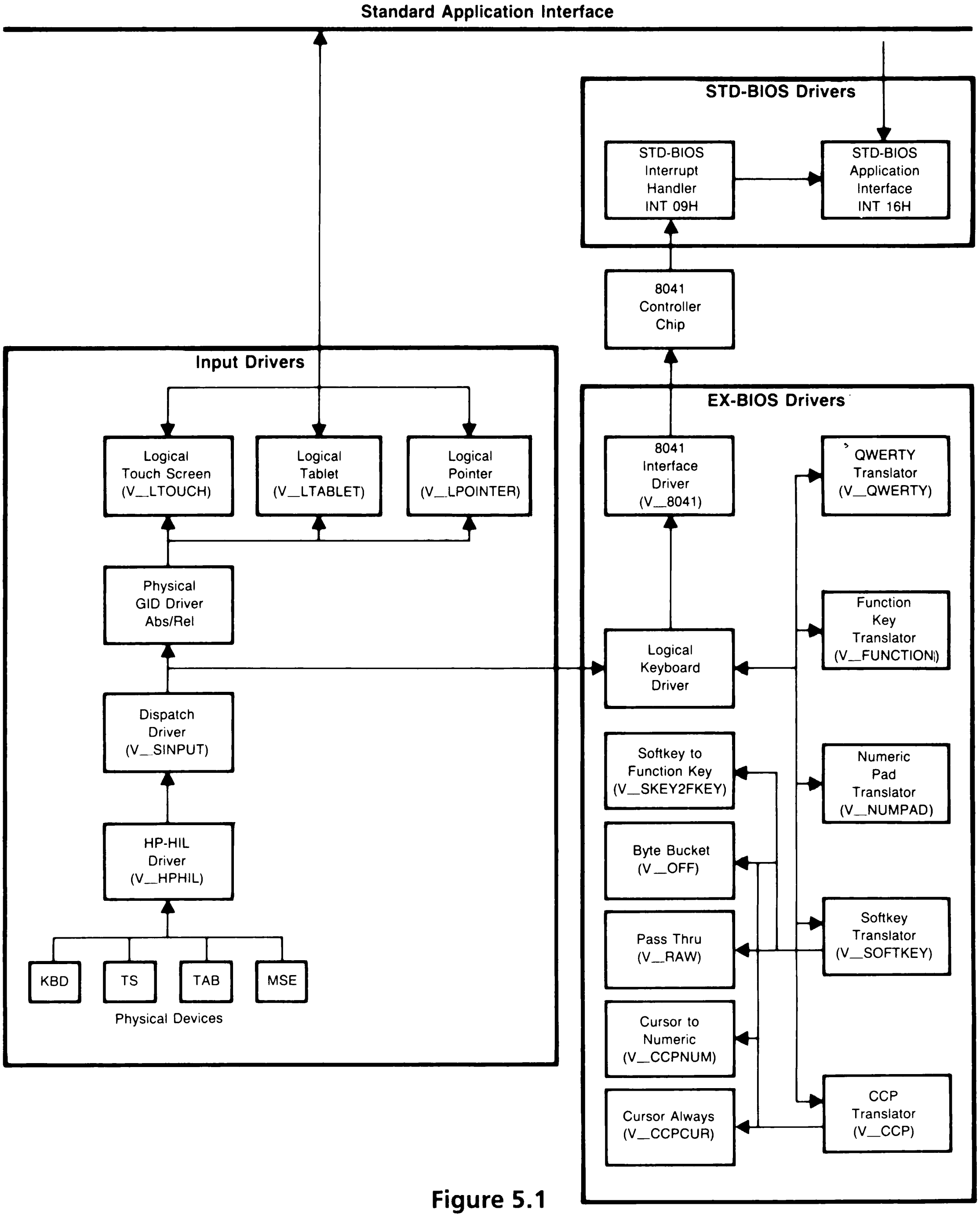

Standard Application Interface
STD-BIOS Drivers
STD-BIOS Interrupt Handler INT 09H
STD-BIOS Application Interface INT 16H
8041 Controller Chip
Input Drivers
Logical Touch Screen (V_LTOUCH)
Logical Tablet (V_LTABLET)
Logical Pointer (V_LPOINTER)
Physical GID Driver Abs/Rel
Dispatch Driver (V_SINPUT)
HP-HIL Driver (V_HPHIL)
KBD
TS
TAB
MSE
Physical Devices
EX-BIOS Drivers
8041 Interface Driver (V_8041)
QWERTY Translator (V_QWERTY)
Function Key Translator (V_FUNCTION)
Logical Keyboard Driver
Softkey to Function Key (V_SKEY2FKEY)
Numeric Pad Translator (V_NUMPAD)
Byte Bucket (V_OFF)
Softkey Translator (V_SOFTKEY)
Pass Thru (V_RAW)
Cursor to Numeric (V_CCPNUM)
CCP Translator (V_CCP)
Cursor Always (V_CCPCUR)

Figure 5.1

5. When an application is ready to receive keyboard input it calls the STD-BIOS INT 16H driver to retrieve the keycode and scancode from the STD-BIOS keyboard buffer.

5.2 STD-BIOS Keyboard Drivers

The STD-BIOS component consists of two drivers: the keyboard ISR routine (INT 09H), and the keyboard interface driver (INT 16H). The drivers discussed here cover steps 4 and 5 in the overview of Section 5.

5.2.1 Overview

The INT 09H driver responds to the 8041 OBF interrupt (generated by V__S8259) and reads in a scancode from the 8041 controller. If the scancode is from one of the keyboard modifier keys, the appropriate state bits are updated. The scancode is then placed in the STD-BIOS keyboard buffer along with its corresponding ASCII character (keycode) or a null byte (0H).

The INT 16H driver provides functions to allow the application to interrogate and manipulate the keyboard input system. Applications may check for keycodes in the STD-BIOS keyboard buffer, remove keycodes from it, and retrieve the state of the keyboard modifiers.

Extended functions are provided by the INT 16H driver to give the application additional control over the keyboard and to facilitate keyboard driver mapping. Extended functions allow the application to turn off or change the default translations performed on the HP Softkeys and Cursor Control keypads (see figure 5.2). Applications may inquire about and/or change the typematic rate and delay values for the keyboard. Functions are also provided to aid applications wishing to install keypad translator services of their own.

5.2.2 Data Structures

The INT 16H and INT 09H driver data structures are located in the STD-BIOS data area. They are stored in memory addresses 417H (40:17H) through 43DH (40:3DH). Table 5.1 lists these memory locations and their definitions.

Keyboard Keypad Groups

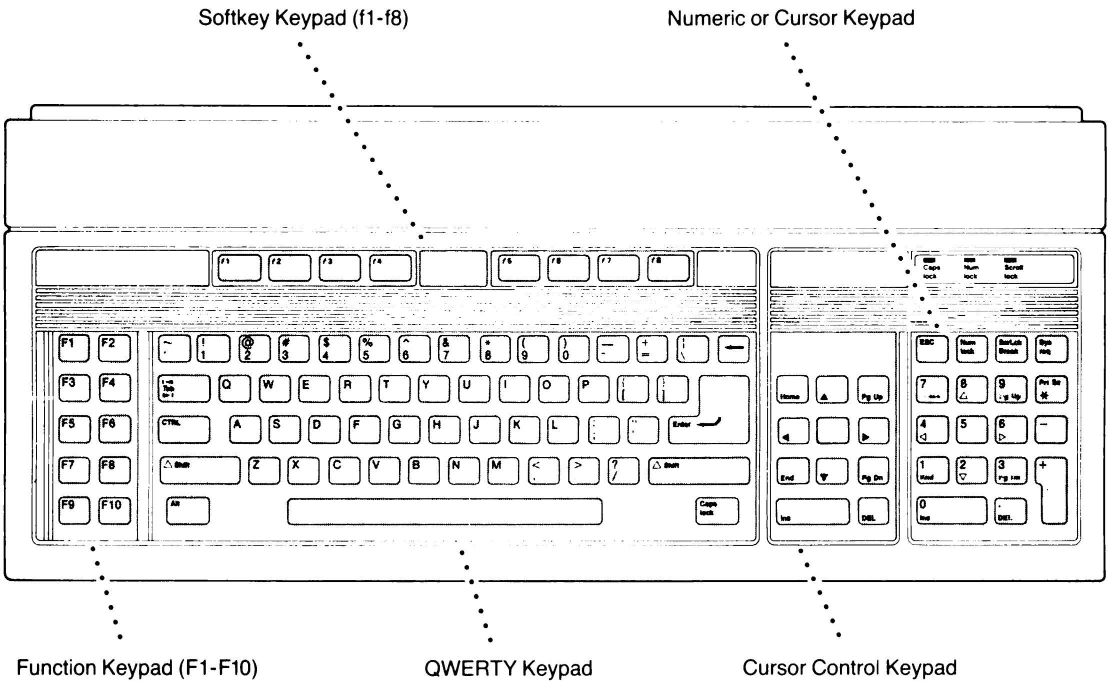


Figure 5.2

Table 5.1

STD-BIOS Keyboard Driver Data Area

Address	Length Bytes	Definition
00417H	2	Keyboard Flags
00419H	1	Alt/Numpad accumulator
0041AH	2	Keyboard buffer head pointer
0041CH	2	Keyboard buffer tail pointer
0041EH	32	Keyboard buffer

The keyboard buffer can store up to 16 entries. Each buffer entry consists of two bytes; an ASCII character (keycode) and a scancode. The keycode and the scancode are placed in the keyboard buffer by the INT 09H driver, and the keyboard head pointer is adjusted accordingly. They are retrieved from the buffer by the INT 16H driver, and the keyboard tail pointer is adjusted.

The keyboard flags are maintained by the INT 09H driver. These flags indicate the state of the keyboard modifier keys and their respective modes. The byte at memory location 417H indicates the mode, while the byte at 418H reflects the actual state of the keys themselves. Tables 5.2 and 5.3 list these flags and their meaning.

Table 5.2

Keyboard Flags (Address 417H)

Address	Bit	Data	Definition
00417H	07H		Insert state
		1	Insert mode is active
	06H		Caps lock state
		1	Caps lock mode is active
	05H		Num lock state
		1	Num lock mode is active
	04H		Scroll lock state
		1	Scroll lock mode is active
	03H		<Alt> key State
		1	<Alt> key is pressed
	02H		<CTRL> key State
		1	<CTRL> key is pressed
	01H		Left <Shift> key state
		1	Left <Shift> key is pressed
	00H		Right <Shift> key state
		1	Right <Shift> key pressed

Table 5.3

Keyboard Flags (Address 418H)

Address	Bit	Data	Definition
00418H	07H		<Ins> key state
		1	<Ins> key is pressed
	06H		<Caps lock> key state
		1	<Caps lock> key is pressed
	05H		<Num lock> key state
		1	<Num lock> key is pressed
	04H		<ScrLck> key state
		1	<ScrLck> key is pressed
	03H		Pause State
		1	Indicates the <CTRL>-<Num lock> pause state is active
	02H		<Sys req> key state
		1	<Sys req> key is pressed
	01H-00H		Reserved

Note:

Applications which modify these two bytes may experience difficulty in maintaining synchronization between the Cursor Control keypad and the Numeric keypad.

5.2.3 STD-BIOS Keyboard ISR (INT 09H)

The keyboard interrupt service routine is responsible for retrieving scancodes from the 8041 controller, generating the associated keycodes, and placing them into the STD-BIOS keyboard buffer. Certain keys and key combinations do not generate a standard ASCII character code. In these cases a keycode equal to 0 indicates that an application program should examine the scancode byte to determine the ''extended'' ASCII code. Table 5.4 contains the scancode to keycode translation assignments.

Table 5.4

Scancode Conversion Table

Key Number	AT Scancode	Hp Scancode	Key Cap	Unshifted ASCII	Hex	Shifted ASCII	Hex	Control	Alt
90	076H	001H	ESC	esc	1BH	esc	1BH	1BH	—
02	016H	002H	1	'1'	31H	'!'	21H	—	00/78H
03	01EH	003H	2	'2'	32H	'@'	40H	00H	00/79H
04	026H	004H	3	'3'	33H	'#'	23H	—	C0/7AH
05	025H	005H	4	'4'	34H	'$'	24H	—	00/7BH
06	02EH	006H	5	'5'	35H	'%'	25H	—	00/7CH
07	036H	007H	6	'6'	36H	'^'	5EH	1EH	00/7DH
08	03DH	008H	7	'7'	37H	'&'	26H	—	00/7EH
09	03EH	009H	8	'8'	38H	'*'	2AH	—	00/7FH
10	046H	00AH	9	'9'	39H	'('	28H	—	00/80H
11	045H	00BH	0	'0'	30H	')'	29H	—	00/81H
12	04EH	00CH	—	'-'	2DH	'_'	5FH	1FH	00/82H
13	055H	00DH	=	'='	3DH	'+'	2BH	—	00/83H
15	066H	00EH	backspace	bs	08H	bs	08H	7FH	—
16	00DH	00FH	Tab	tab	09H	si	0FH	—	—
17	015H	010H	Q	'q'	71H	'Q'	51H	11H	00/10H
18	01DH	011H	W	'w'	77H	'W'	57H	17H	00/11H
19	024H	012H	E	'e'	65H	'E'	45H	05H	00/12H
20	02DH	013H	R	'r'	72H	'R'	52H	12H	00/13H
21	02CH	014H	T	't'	74H	'T'	54H	14H	00/14H
22	035H	015H	Y	'y'	79H	'Y'	59H	19H	00/15H
23	03CH	016H	U	'u'	75H	'U'	55H	15H	00/16H
24	043H	017H	I	'i'	69H	'I'	49H	09H	00/17H
25	044H	018H	O	'o'	6FH	'O'	4FH	0FH	00/18H
26	04DH	019H	P	'p'	70H	'P'	50H	10H	00/19H
27	054H	01AH	[	'['	5BH	'{'	7BH	1BH	—
28	05BH	01BH	]	']'	5DH	'}'	7DH	1DH	—
43	05AH	01CH	Enter	cr	0DH	cr	0DH	0AH	—
30	014H	01DH	CTRL	–	–	–	–	—	—
31	01CH	01EH	A	'a'	61H	'A'	41H	01H	00/1EH
32	01BH	01FH	S	's'	73H	'S'	53H	13H	00/1FH
33	023H	020H	D	'd'	64H	'D'	44H	04H	00/20H
34	02BH	021H	F	'f'	66H	'F'	46H	06H	00/21H
35	034H	022H	G	'g'	67H	'G'	47H	07H	00/22H
36	033H	023H	H	'h'	68H	'H'	48H	08H	00/23H
37	03BH	024H	J	'j'	6AH	'J'	4AH	0AH	00/24H
38	042H	025H	K	'k'	6BH	'K'	4BH	0BH	00/25H
39	04BH	026H	L	'l'	6CH	'L'	4CH	0CH	00/26H
40	04CH	027H	;	';'	3BH	':'	3AH	—	—
41	052H	028H	'	'''	27H	'"'	22H	—	—
01	00EH	029H	'	'''	60H	'~'	7EH	—	—
44	012H	02AH	Left Shift	–	–	–	–	—	—

Key	AT	Hp		Unshifted		Shifted			
Number	Scancode	Scancode	Key Cap	ASCII	Hex	ASCII	Hex	Control	Alt
14	05DH	02BH	\	'\'	5CH	'|'	7CH	1CH	—
46	01AH	02CH	Z	'z'	7AH	'Z'	5AH	1AH	00/2CH
47	022H	02DH	X	'x'	78H	'X'	58H	18H	00/2DH
48	021H	02EH	C	'c'	63H	'C'	43H	03H	00/2EH
49	02AH	02FH	V	'v'	76H	'V'	56H	16H	00/2FH
50	032H	030H	B	'b'	62H	'B'	42H	02H	00/30H
51	031H	031H	N	'n'	6EH	'N'	4EH	0EH	00/31H
52	03AH	032H	M	'm'	6DH	'M'	4DH	0DH	00/32H
53	041H	033H	,	','	2CH	'<'	3CH	—	—
54	049H	034H	.	'.'	2EH	'>'	3EH	—	—
55	04AH	035H	/	'/'	2FH	'?'	3FH	—	—
57	059H	036H	Right Shift	–	–	–	–	—	—
106	07CH	037H	Prt Sc	'*'	2AH	–	–	00/72H	—
58	011H	038H	Alt	–	–	–	–	—	—
61	029H	039H	Space	' '	20H	' '	20H	20H	20H
64	058H	03AH	Caps lock	–	–	–	–	—	—
70	005H	03BH	F1	–	3BH	–	54H	00/5EH	00/68H
65	006H	03CH	F2	–	3CH	–	55H	00/5FH	00/69H
71	004H	03DH	F3	–	3DH	–	56H	00/60H	00/6AH
66	00CH	03EH	F4	–	3EH	–	57H	00/61H	00/6BH
72	003H	03FH	F5	–	3FH	–	58H	00/62H	00/6CH
67	00BH	040H	F6	–	40H	–	59H	00/63H	00/6DH
73	083H	041H	F7	–	41H	–	5AH	00/64H	00/6EH
68	00AH	042H	F8	–	42H	–	5BH	00/65H	00/6FH
74	001H	043H	F9	–	43H	–	5CH	00/66H	00/70H
69	009H	044H	F10	–	44H	–	5DH	00/67H	00/71H

				NumLock		None Or	
Key	AT	Hp		or		NumLock	
Number	Scancode	Scancode	Key Cap	Shift		and Shift	Control
95	077H	045H	Num lock	–	45H	—	—
100	07EH	046H	ScrLck	–	46H	—	—
91	06CH	047H	Home	'7'	37H	00/47H	0077H
96	075H	048H	↑	'8'	38H	00/48H	—
101	07DH	049H	Pg Up	'9'	39H	00/49H	00/84H
107	07BH	04AH	—	'-'	3AH	3AH	—
92	06BH	04BH	←	'4'	34H	00/4BH	00/73H
97	073H	04CH	5	'5'	35H	—	—
102	074H	04DH	→	'6'	36H	00/4DH	00/74H
108	079H	04EH	+	'+'	2BH	2BH	—
93	069H	04FH	End	'1'	31H	00/4FH	00/75H
98	072H	050H	↓	'2'	32H	00/50H	—
108	07AH	051H	Pg Dn	'3'	33H	00/51H	00/76H
99	070H	052H	Ins	'0'	30H	00/52H	—
104	071H	053H	DEL	'.'	2EH	00/53H	—
105	084H	054H	Sysreq	–	–	—	—

Key Number	AT Scancode	Hp Scancode	Key Cap	Unshifted ASCII Hex	Shifted ASCII Hex	Control	Alt
		055H	- undef.				
		056H	- undef.				
		057H	- undef.				
		058H	- undef.				
		059H	- undef.				
		05AH	- undef.				
		05BH	- undef.				
		05CH	- undef.				
		05DH	- undef.				
59		05EH	Unlabled-L	00/D7H	00/BDH	00/A3H	00/89H
62		05FH	Unlabled-R	00/D8H	00/BEH	00/A4H	00/8AH
113		060H	CCP-Up	00/D9H	00/BFH	00/A5H	00/8BH
111		061H	CCP-Lft	00/DAH	00/C0H	00/A6H	00/8CH
115		062H	CCP-Dn	00/DBH	00/C1H	00/A7H	00/8DH
118		063H	CCP-Rht	00/DCH	00/C2H	00/A8H	00/8EH
110		064H	CCP-Home	00/DDH	00/C3H	00/A9H	00/8FH
117		065H	CCP-PgUp	00/DEH	00/C4H	00/AAH	00/90H
112		066H	CCP-End	00/DFH	00/C5H	00/ABH	00/91H
119		067H	CCP-PgDn	00/E0H	00/C6H	00/ACH	00/92H
116		068H	CCP-Ins	00/E1H	00/C7H	00/ADH	00/93H
120		069H	CCP-Del	00/E2H	00/C8H	00/AEH	00/94H
114		06AH	CCP-CNTR	00/E3H	00/C9H	00/AFH	00/95H
		06BH	- undef.	00/E4H	00/CAH	00/B0H	00/96H
		06CH	- undef.	00/E5H	00/CBH	00/B1H	00/97H
		06DH	- undef.	00/E6H	00/CCH	00/B2H	00/98H
		06EH	- undef.	00/E7H	00/CDH	00/B3H	00/99H
		06FH	- undef.	00/E8H	00/CEH	00/B4H	00/9AH
121		070H	f1	00/E9H	00/CFH	00/B5H	00/9BH
122		071H	f2	00/EAH	00/D0H	00/B6H	00/9CH
123		072H	f3	00/EBH	00/D1H	00/B7H	00/9DH
124		073H	f4	00/ECH	00/D2H	00/B8H	00/9EH
125		074H	f5	00/EDH	00/D3H	00/B9H	00/9FH
126		075H	f6	00/EEH	00/D4H	00/BAH	00/A0H
127		076H	f7	00/EFH	00/D5H	00/BBH	00/A1H
128		077H	f8	00/F0H	00/D6H	00/BCH	00/A2H
		078H through 7FH—undef.					

The INT 09H driver tracks the state of the keyboard modifiers presented in tables 5.2 and 5.3 as well as processing the special key combinations in table 5.5.

Table 5.5

INT 09H Special Key Sequences

Key Combinations	Action
<CTRL>-<Num lock>	Stops execution until any non-shift key on the keyboard is struck.
<CTRL>-<Alt>-< + >	This key sequence enables the key click feature. The longer the <CTRL>-<Alt>-< + > keys are pressed, the louder the key click that will result. After maximum volume is achieved the key click volume will wrap around to low volume. Applications which depend upon datacom rates at and above 9600 baud while keyboard input is being entered should disable the keyclick feature.
<CTRL>-<Alt>-<->	This key sequence reduces the key click volume until it is off.
<CTRL>-<Break>	This key combination is interpreted as a program break request. When this key combination is detected, the INT 09H driver will execute an INT 1BH instruction. The vector for this interrupt is initialized during the boot process to point to a routine within MS-DOS which sets a flag then performs an IRET instruction. This vector may be modified to point to an alternate routine to handle a <CTRL>-<Break>.
<CTRL>-<Alt>-<DEL>	This key combination is interpreted as a system reset command. When this key combination is detected, control is transferred to the BIOS Reset routine.
<Shift>-<Prt Sc>	This key combination is interpreted as a print screen command. When this key combination is detected, an INT 05H instruction is executed.
<Sys req>	This key is interpreted as a system request for multi-tasking.

Key Combinations	Action
<CTRL>-<Alt>-<Sys req>	This key sequence provides the user with a method of generating a hard reset sequence. The key sequence is communicated to the ROM-BIOS via a non-maskable interrupt (NMI) to the 80286. This key sequence does not require the HP-HIL firmware interface to be operational. The key sequence is used to recover from exceptional error conditions without power cycling the system. The EX-BIOS code then: 1) Inspects for the source of the NMI (either I/O channel check, Memory Parity Error, or as in this case NMI-RESET). 2) Clears CMOS location 28H, 29H, 2AH, and 2CH to its default setting. 3) Jumps to location F000H:FFFEH.
<ALT>-nnn	Where nnn represents a three digit decimal number entered on the numeric keypad and yields the associated ASCII characters, i.e., <ALT>-122 yields the character ''z''.

5.2.4 STD-BIOS Keyboard Driver (INT 16H)

The INT 16H driver acts as the interface between applications and the keyboard. This driver has two sets of functions. One set provides functions to return keycodes and keyboard status. The other set of functions allows the application to change the translation algorithms of the scancodes and to vary the repeat rates of the keys on the keyboard. Table 5.6 contains a summary of this driver's function codes.

Table 5.6

Keyboard Driver (INT 16H) Function Code Summary

Int Hex	Function Equate	Function Value	Definition
16H	INT__KBD		Keyboard
	F16__GET__KEY	00H	Read keycode from keyboard buffer
	F16__STATUS	01H	Report Status of keyboard buffer
	F16__KEY__STATE	02H	Get Key Modifier Status
	F16__INQUIRE	6F00H	EX-BIOS present
	F16__DEF__ATTR	6F01H	Report default typematic values
	F16__GET__ATTR	6F02H	Report typematic values
	F16__SET__ATTR	6F03H	Set typematic values
	F16__DEF__MAPPING	6F04H	Report default transfer assignments
	F16__GET__MAPPING	6F05H	Report transfer assignments
	F16__SET__MAPPING	6F06H	Set transfer assignments
	F16__SET__XLATORS	6F07H	Set CCP and softkey pads
	F16__KBD	6F08H	Report keyboard information
	F16__KBD__RESET	6F09H	Reset keyboard to defaults

Keyboard Driver (INT 16H) Function Definitions

F16__GET__KEY (AH = 00H)

This function returns the next keycode from the keyboard buffer. If no keycode is ready, this function waits for one.

On Entry: AH = F16__GET__KEY (00H)

On Exit: AH = Scancode
AL = ASCII keycode or extended keycode

Registers Altered: AX

F16__STATUS (AH = 01H)

This function returns the status of the keyboard buffer. The Zero flag is cleared if a keycode is available, or set if there is no keycode in the buffer. If a keycode is ready, the scancode and keycode are returned in the AH and AL registers respectively. Even though the scancode and keycode are returned with this function, they must be read with F16__GET__KEY to remove them from the keyboard buffer.

On Entry: AH = F16__STATUS (01H)

On Exit: Z = 1 if no keycode is ready.
Z = 0 if a keycode is ready.
and
AH = Scancode
AL = Keycode or extended keycode.

Registers Altered: AX

F16__KEY__STATE (AH = 02H)

This function returns the state of the various keyboard modifiers. The status byte returned is a copy of the keyboard modifier status byte stored at memory location 417H.

On Entry: AH = F16__KEY__STATE (02H)

On Exit: AL = Modifier Status Byte

Bit	Data	Definition
07H	1	Insert mode active
	0	Insert mode inactive
06H	1	Caps lock mode active
	0	Caps lock mode inactive
05H	1	Num lock mode active
	0	Num lock mode inactive
04H	1	Scroll lock mode active
	0	Scroll lock mode inactive
03H	1	<Alt> key pressed
	0	<Alt> key released
02H	1	<CTRL> key pressed
	0	<CTRL> key released
01H	1	Left <Shift> key pressed
	0	Left <Shift> key released
00H	1	Right <Shift> key pressed
	0	Right <Shift> key pressed

Registers Altered: AL

F16__INQUIRE (AX = 6F00H)

This subfunction determines whether or not the extended HP functions are available. If the HP functions are available, the BX register will be set to 4850H (which is the ASCII characters 'HP').

On Entry: AX = F16__INQUIRE (6F00H)
BX = Any value except 4850H, 'HP'.

On Exit: BX = 'HP'

Registers Altered: BX

F16__DEF__ATTR (AX = 6F01H)

This subfunction reports the default typematic rate and delay values for the keyboard. A pointer to a four byte buffer is returned. The bytes in the buffer are defined in table 5.7.

Table 5.7

INT 16H Typematic Buffer Format

Byte	Function
0	Delay before repeat action starts for all keys, except the Cursor Control Pad.
1	Typematic Repeat rate for all keys, except the Cursor Control Pad.
2	Delay before repeat action starts for all Cursor Control Pad keys.
3	Typematic Repeat rate for all Cursor Control Pad keys.

Table 5.8 summarizes the typematic rate and delay values defined for each data byte accepted in the typematic buffer by the INT 16H driver. Note that the typematic rates are the same for both the HP cursor control pad and the non-cursor pad keys while two delay values are provided (one for each group).

Table 5.8

INT 16H Typematic Rates and Delays

Data Byte	Byte 1 and 3 Reports per Poll*	Byte 2 Number of Polls Delayed**	Byte 0 Number of Polls Delayed
00H	1 (60.00)	1 [0.017]	1 [0.017]
01H	2 (30.00)	5 [0.083]	9 [0.150]
02H	3 (20.00)	9 [0.150]	17 [0.283]
03H	4 (15.00)	13 [0.217]	25 [0.417]
04H	5 (12.00)	17 [0.283]	33 [0.550]
05H	6 (10.00)	21 [0.350]	41 [0.683]
06H	7 (8.57)	25 [0.417]	49 [0.817]
07H	8 (7.50)	29 [0.483]	57 [0.950]
08H	9 (6.66)	33 [0.550]	65 [1.083]
09H	10 (6.00)	37 [0.617]	73 [1.217]
0AH	11 (5.45)	41 [0.683]	81 [1.350]
0BH	12 (5.00)	45 [0.750]	89 [1.483]
0CH	13 (4.62)	49 [0.817]	97 [1.617]
0DH	14 (4.28)	53 [0.883]	105 [1.750]
0EH	15 (4.00)	57 [0.950]	113 [1.883]
0FH	none (off)	61 [1.017]	121 [2.017]

*Numbers in parentheses ( ) indicate the approximate number of repeated scancodes per second (assuming a poll rate of 60 cycles per second).

**Numbers in brackets [ ] indicate the approximate length of delay prior to the first repeated scancode report (assuming a poll rate of 60 cycles per second).

On Entry: AX = F16__DEF__ATTR (6F01H)

On Exit:
AH = 00H (Successful operation)
ES:SI = Pointer to buffer
CX = 4 (Number of entries in table)

Registers Altered: AX, CX, SI, ES

F16__GET__ATTR (AX = 6F02H)

This subfunction reports the current typematic rate and delay values for the keyboard. A pointer to a four byte buffer is returned. The bytes in the buffer are interpreted as shown in table 5.7 and 5.8.

```
On Entry:   AX  = F16__GET__ATTR (6F02H)

On Exit:    AH  = 00H (Successful operation)
         ES:SI  = Pointer to buffer
            CX  = 4 (Number of entries in table)
```

Registers Altered: AX, CX, SI, ES

F16__SET__ATTR (AX = 6F03H)

This subfunction sets the current typematic rate and delay values for the keyboard. A pointer to a four byte buffer is passed. The bytes in the buffer are interpreted as shown in table 5.7 and 5.8.

```
On Entry:   AX  = F16__SET__ATTR (6F03H)
         ES:SI  = Pointer to buffer

On Exit:    AH  = 00H (Successful operation)
```

Registers Altered: AX

F16__DEF__MAPPING (AX = 6F04H)

This subfunction reports the default keyboard translator mappings. A pointer to a buffer of 1EH bytes is supplied by the caller to be filled in by the ROM-BIOS. The table will contain the default HP__VECTOR__TABLE entries for each of the five translator drivers. Each of five entries in the table will contain the IP, CS, and DS for each translator driver.

Caution

> An application should restore the translator drivers to their original condition upon termination. If an application replaces one of these drivers it should be aware that STD-BIOS keyboard driver functions 6F07H may no longer function properly.

The format of the buffer is given in table 5.9.

Table 5.9

INT 16H Mapping Buffer Format

Byte	Translator
00H	Entry for V__QWERTY driver
06H	Entry for V__SOFTKEY driver
0CH	Entry for V__FUNCTION driver
12H	Entry for V__NUMPAD driver
18H	Entry for V__CCP driver

On Entry: AX = F16__DEF__MAPPING (6F04H)
ES:SI = Pointer to buffer

On Exit: AH = 00H (Successful)
ES:SI = Pointer to buffer of 1EH bytes
CX = 1EH (Size of buffer)

Registers Altered: AX, CX

F16__GET__MAPPING (AX = 6F05H)

This subfunction reports the current keyboard translator mappings. A pointer to a buffer 1EH bytes in length is supplied by the caller to be filled in by the ROM-BIOS. The buffer will contain the current HP__VECTOR__TABLE entries for each of the five translator drivers (IP, CS, and DS for each driver). The format of the buffer is given in table 5.9.

On Entry: AX = F16__GET__MAPPING (6F05H)
ES:SI = Pointer to buffer

On Exit: AH = 00H (Successful)
ES:SI = Pointer to buffer
CX = 1EH (Size of table)

Registers Altered: AX, CX

F16__SET__MAPPING (AX = 6F06H)

This subfunction sets the current keyboard translator mappings. A pointer to a buffer containing the entries to be written into the HP__VECTOR__TABLE is passed in. The format of the buffer is given in table 5.9.

A driver that replaces a scancode translator can expect to handle a Keyboard ISR Event Record (table 5.10). If the translator wishes to remove the passed in scancode from the scancode stream, it returns a status of RS_DONE. Otherwise, a return status of RS_SUCCESSFUL should be set and an appropriate ISR EVENT record returned. The ISR Event Record will then be passed on to the next driver in the chain. The driver can depend on 20H bytes of stack.

On Entry: AX = F16_SET_MAPPING (6F06H)
ES:SI = Pointer to table.
CX = 01EH (size of table in bytes)

On Exit: AH = 00H (Successful)

Registers Altered: AX

F16_SET_XLATORS (AX = 6F07H)

This subfunction sets the current mappings of the HP Softkey (V_SOFTKEY) and HP Cursor Control Pad (V_CCP) translators. Note that only one translator may be set with each call to this subfunction. Figure 5.1 shows the possible mappings for the two HP proprietary keypads.

On Entry: AX = F16_SET_XLATORS (6F07H)
BL = Translation

Data	Definition
00H	Maps V_CCP to V_CCPCUR which forces the HP Cursor Pad to generate Numeric pad cursor key scancodes, regardless of state of <Num lock>. (Default mapping)
01H	Maps V_CCP to V_CCPNUM which forces the HP Cursor Pad to generate numeric pad or cursor key scancodes, depending on state of <Num lock>.
02H	Maps V_CCP to V_OFF which disables the HP Cursor Pad.
03H	Maps V_CCP to V_CCPGID (if installed) which converts HP Cursor Pad data to GID data.
04H	Maps V_CCP to V_RAW which passes HP Cursor Pad scancodes untranslated to the INT 09H driver.
05H	Maps V_SOFTKEY to V_SKEY2FKEY which translates HP Softkey scancodes into equivalent industry standard function key scancodes. (Default mapping)
06H	Maps V_SOFTKEY to V_RAW which passes HP Softkey scancodes untranslated to INT 09H driver.
07H	Maps V_SOFTKEY to V_OFF which disables HP Softkeys.

On Exit: AH = 00 (Successful)

Registers Altered: AX

F16__KBD (AX = 6F08H)

This subfunction returns the HP-HIL ID and address of the keyboard. The HP-HIL address (BH) may be used to locate the logical keyboard driver in the HP__VECTOR__TABLE. The logical keyboard driver's vector address is:

$$\text{vector address} = (BH - 1) \times 6 + n$$

Where n is the vector address of the first HP-HIL physical device driver (see Section 4, V__SINPUT function F__INQUIRE__FIRST.

On Entry: AX = F16__KBD (6F08H)

On Exit: AH = 00H (Successful)
BH = HP-HIL Address
BL = HP-HIL ID

Registers Altered: AX, BX

F16__KBD__RESET (AX = 6F09H)

This subfunction resets all keyboard mappings to their default translators and resets all keyboard typematic rates and delays to their default values.

On Entry: AX = F16__KBD__RESET (6F09H)

On Exit: AH = 00H (Successful)

Registers Altered: AX

5.3 EX-BIOS Keyboard Drivers

The rest of this section discusses keyboard information related to ISR events and ISR Event Records, device driver chains, and HP-HIL device data input; these concepts were introduced in Section 4.

5.3.1 Overview

The EX-BIOS keyboard component consists of the logical keyboard driver, the keyboard translator services, and the V__8041 interface driver. The drivers discussed here cover steps 2 and 3 in the data flow of Section 5.1.

5.3.1.1 Logical Keyboard Driver

The logical keyboard driver is the primary interface for the physical keyboard and controls the process of scancode translation. Based on the keypad, the scancode is passed to one of five translator services: V__QWERTY, V__SOFTKEY, V__FUNCTION, V__CCP and V__NUMPAD. Figure 5.2 shows the layout of the different keypad groups. This driver also maintains the state of the following keyboard modifier keys: <CTRL>, left and right <Shift>, <Alt>, <Caps lock>, and <Num lock>. This state information is passed to the V__CCP, V__NUMPAD and V__QWERTY translator services.

5.3.1.2 Keyboard Translators

The keyboard translators act as subroutines for the logical keyboard driver. There are five translators corresponding to the keyboard keypads (see figure 5.2). The five translators are:

V__QWERTY handles keys from the QWERTY keypad.
V__FUNCTION handles F1 thru F10 function keys.
V__NUMPAD handles numeric or cursor pad keys.
V__SOFTKEY handles HP's f1 thru f8 softkeys.
V__CCP handles HP's cursor control pad.

The translators for the HP softkeys and HP cursor control pad are special cases.

The V__SOFTKEY translator can translate its scancodes in the following ways:

1. Map softkeys f1 thru f8 into function keys F1 thru F8 (V__SKEY2FKEY).
2. Throw away f1 thru f8 softkeys (V__OFF).
3. Pass back f1 thru f8 softkeys untranslated to the logical keyboard driver (V__RAW).

The V__CCP translator can translate its scancodes in the following ways:

1. Map CCP keys to numeric keypad cursor control scancodes (V__CCPCUR).
2. Map CCP keys to numeric keypad scancodes (V__CCPNUM).
3. Pass CCP keys as untranslated scancodes to the logical keyboard driver (V__RAW).
4. Throw away all CCP keys (V__OFF).

Functions are provided by the STD-BIOS INT 16H driver to select any of the above mappings.

5.3.1.3 8041 Interface Driver

The 8041 interface driver (V__8041) sends translated scancodes to the 8041 controller chip. If the 8041 controller is busy this driver queues the scancode to be sent later when the 8041 controller is ready. In addition to passing scancodes from the keyboard to the 8041 controller, V__8041 processes keyboard controller commands to set keyboard LED's and change keyboard typematic rates.

5.3.2 Data Structures

The EX-BIOS keyboard input system uses one data structure. The Keyboard ISR Event Record is a set of register definitions for inter-driver communication of input events. Table 5.10 contains the Keyboard ISR Event Record definition.

Table 5.10

Keyboard ISR Event Record

AH = F_ISR (00H)
BH = Keyboard State (Only if state bit set in Data Type)

Bit	Data	Definition
07H	1	Left Unlabeled key pressed*
06H	1	Right Unlabeled key pressed*
05H	1	<Num lock> state active
04H	1	<Caps lock> state active
03H	1	<CTRL> key pressed
02H	1	Right <Shift> key pressed
01H	1	Left <Shift> key pressed
00H	1	<Alt> key pressed

BL = Scancode

Bit	Data	Definition
07H	1	Break indicator
	0	Make indicator
06H-00H	—	Scancode

CX = Number of bytes in buffer (scancode strings only)
DH = Data Type
DL = Logical keyboard drivers vector address / 6
BP = HP-HIL device n vector address
ES:SI = Pointer to buffer (scancode strings only)

* These keys are located to the immediate left and right of the space bar. They are only available on some international keyboards.

The Data Type field (DH) contains a code representing the current type of scancode contained in the ISR Event Record. When the logical keyboard driver calls a translator service, the Data Type will match the keypad group from which the scancode originated. After translation, the Data Type for the ISR Event Record returned to the logical keyboard driver should be T_KC_IBM_PC. See table 5.11 for a complete list of keyboard data types.

Table 5.11

Keyboard Event Data Types

Type	Value	Definition
T_KC_R0	00H	Reserved
T_KC_R1	01H	Reserved
T_KC_ASCII	02H	ASCII data
T_KC_R3	03H	Reserved
T_KC_ITF	04H	HP150 keyboard (ITF) scancode
T_KC_R5	05H	Reserved
T_KC_WILD	06H	Device definable type
T_KC_HPHIL_ENVOY	07H	HP Vectra Keyboard set
T_KC_IBM_AT	08H	IBM-AT scancode set
T_KC_BUTTON	09H	Button data type
T_KC_IBM_PC	0AH	IBM-PC scancode set
T_KC_HP_SOFTKEY	0BH	Softkey keypad (f1-f8)
T_KC_IS_FUNCTION	0CH	Function key keypad (F1-F10)
T_KC_HP_CCP	0DH	HP Cursor Control Pad keypad
T_KC_QWERTY	0EH	Qwerty keypad
T_KC_NUMPAD	0FH	Numeric keypad
T_STRING	10H	This is not a data type but an indicator bit for the keyboard data types only. If bit 4 is set then the ISR Event record is for a string of scancodes pointed to by ES:SI and enumerated in CX, i.e., 00 × 1 ttttB indicates a string of data bytes of type defined by the lower nibble 'tttt'.
T_STATE	20H	This is not a data type but an indicator bit for the keyboard data types only. If bit 5 is set it indicates that the corresponding ISR Event record contains the current state in BH.

5.3.3 Logical Keyboard Driver

The logical keyboard driver determines the keypad group the scancode belongs to and sets the Data type field in the ISR event record. Based on the Data type a translator service is called to handle the scancode. For example, If the "Q" key scancode comes through, the logical keyboard driver determines the data type to be T__KC__QWERTY and calls the V__QWERTY translator. If the translator called by the logical keyboard driver is responsible for any of the keyboard modifier keys the current state variable is placed in the ISR Event Record and the state indicator bit is set in the Data Type field. Table 5.12 contains the scancode range to translator service assignments.

Table 5.12

Scancode to Translator Assignments

Driver Name	Scancode Range	Translation Performed
V__QWERTY	00H-36H 38H-3AH 55H-5FH 6BH-6FH 78H-7FH	None
V__SOFTKEY	70H-77H	3BH—42H (F1—F8)
V__FUNCTION	3BH-44H	None
V__NUMPAD	37H, 45H-54H	None
V__CCP	60H-6AH	Cursor Always—Regardless of state of the <Num lock> and <Shift> keys.

If the translation was successful the returned ISR Event Record is passed to the logical keyboard drivers parent (V__8041).

Before passing a successful translation to its parent (V__8041) the logical keyboard driver performs two conditional tasks. First, it checks the state bit in the returned Data Type, if set the master copy of the keyboard state variable is updated with the copy returned in the ISR Event Record. Second, if the ISR event went to the V__CCP translator the logical keyboard driver takes the necessary steps to insure that cursor control keys are generated regardless of the <num lock> and <shift> key states.

If a translator wants to remove the scancode from the scancode stream it must return a status code of RS__DONE to the logical keyboard driver (See the CCP2GID driver in Appendix G).

Table 5.13 contains a summary of the logical keyboard driver functions.

Table 5.13

Logical Keyboard Driver Function Code Summary

Vector Address	Func. Value	Function Equate	Definition
xxxH		Keyboard Driver	(This driver does not have a fixed HP__VECTOR__TABLE address)
	00	F__ISR	Logical Interrupt
	02	F__SYSTEM	System Intrinsics
	02/00	SF__INIT	Driver initialization
	02/06	SF__VERSION__DESC	Reports HP version number

Logical Keyboard Driver Function Definitions

F__ISR (AH = 00H)

This function processes the Keyboard ISR Event Record. It determines the range of the scancode, then calls the appropriate translation service.

On Entry: AH = F__ISR (00H)
BH = Keyboard State (only if state bit set in Date type)
BL = Scancode
CX = Number of bytes in buffer (scancode strings only)
DH = Scancode type
DL = Vector address of keyboard / 6
BP = HP-HIL device n vector address
ES:SI = Pointer to buffer (scancode strings only)

On Exit: AH = Return Status Code

Registers Altered: AX, BX, CX, DX, SI, BP, ES, DS

SF__INIT (AX = 0200H)

This subfunction is called to initialize the driver. Refer to Section 9 for a complete discussion of the protocol utilized in data space allocation ("last used DS" passed in register BX).

```
On Entry:  AH  =  F__SYSTEM (02H)
           AL  =  SF__INIT (00H)
           BX  =  ''Last used DS'' in HP Data Area
           BP  =  HP-HIL device n vector address

On Exit:   AH  =  Return Status Code
           BX  =  New ''last used DS'' is HP Data Area

Registers Altered:  AX, BX, BP, DS
```

SF__VERSION__DESC (AX = 0206H)

This subfunction returns the release date code and a double word pointer to the current version number. The date code consists of two BCD coded bytes containing the year and week of release. The BL register contains the number of years since 1960 and the BH register contains the week of the year.

```
On Entry:  AH  =  F__SYSTEM (02H)
           AL  =  SF__VERSION__DESC (06H)
           BP  =  HP-HIL device n vector address

On Exit:   AH  =  Return Status Code
           BX  =  Release date code
           CX  =  Number of bytes in current version number
        ES:DI  =  Pointer to the current version number

Registers Altered:  AX, BX, CX, DI, ES, BP, DS
```

5.3.4 Keyboard Translators

There is one keyboard translator service for each of the five keypad groups on the keyboard, see figure 5.2. Two of the five services are special cases in that they are actually chains of translators to facilitate keyboard mapping. Figure 5-1 shows the translators and their mapping possibilities.

Applications may install routines to replace (or chain to) any one or all of the translators presented here. The INT 16H driver provides three functions to get the current HP__VECTOR__TABLE entries for the five keypad translators, to set these same values, and to reset them to their default values. The V__SYSTEM driver in Section 9 provides functions to get or set any fixed HP__VECTOR__TABLE entry (all EX-BIOS translators presented in this section have fixed entries). The V__SYSTEM functions allow replacement of translators other than the main five called by the logical keyboard driver (those in translator chains).

Applications that do not wish to overlay existing translators, may install entirely new translators instead and map themselves into the HP Softkey and CCP translator chains as the parent drivers of the V__SOFTKEY and V__CCP services respectively. This method only works for the HP proprietary keypads.

5.3.4.1 V__SOFTKEY (BP = 003CH)

This translator service verifies the Data Type is T__KC__HP__SOFTKEY and then passes the ISR Event Record to its parent. By default this translator is mapped to the V__SKEY2FKEY service, alternative mappings are presented in table 5.14.

Table 5.14

V__SOFTKEY Driver Mapping Alternatives

Driver Name	Function
V__OFF	Discards the ISR event.
V__RAW	Returns the scancode untranslated.
V__SKEY2FKEY	Translates the HP Softkeys into their respective industry standard function key equivalents.

F__ISR (AH = 00H)

This function verifies the passed in Data Type and passes the ISR event on to its parent.

```
On Entry:  AH = F__ISR (00H)
           BH = Keyboard state (only if state bit set in type)
           BL = Scancode
           DH = Scancode type (T__KC__HP__SOFTKEY = 0BH)
           DL = Source vector address / 6
           BP = V__SOFTKEY (003CH)

On Exit:   AH = Return Status Code
           BL = Translated scancode
           BH = New keyboard state (only if state bit set in type)
           DH = New scancode type (T__KC__IBM__PC = 0AH)
```

Registers Altered: AX, BX, DH, BP, DS

SF__INIT (AX = 0200H)

This subfunction is called to initialize the driver. Refer to Section 9 for a complete discussion of the protocol utilized in data space allocation (''last used DS'' passed in register BX).

```
On Entry:  AH  =  F__SYSTEM (02H)
           AL  =  SF__INIT (00H)
           BX  =  ''Last used DS'' in HP Data Area
           BP  =  V__SOFTKEY (003CH)

On Exit:   AH  =  Return Status Code
           BX  =  ''New last used DS'' in HP Data Area
```

Registers Altered: AX, BX, BP, DS

SF__VERSION__DESC (AX = 0206H)

This subfunction returns the release date code and a double word pointer to the current version number. The date code consists of two BCD coded bytes containing the year and week of release. The BL register contains the number of years since 1960 and the BH register contains the week of the year.

```
On Entry:  AH  =  F__SYSTEM (02H)
           AL  =  SF__VERSION__DESC (06H)
           BP  =  V__SOFTKEY (003CH)

On Exit:   AH  =  Return Status Code
           BX  =  Release date code
           CX  =  Number of bytes in current version number
        ES:DI  =  Pointer to the current version number
```

Registers Altered: AX, BX, CX, DI, ES, BP, DS

5.3.4.2 V__QWERTY (BP = 0036H)

The V__QWERTY service verifies the correct Data Type. This service also maintains the state of the left and right <Shift> keys, the <CTRL> key, the <Alt> key, the left and right unlabeled keys and the <Caps lock> key.

F__ISR (AH = 00H)

This function verifies the Data Type, updates the keyboard state variable, and returns.

On Entry: AH = F__ISR (00H)
BH = Keyboard state (only if state bit set in type)
BL = Scancode
DH = Scancode type (T__KC__QWERTY = 0EH)
DL = Source vector address / 6
BP = V__QWERTY (0036H)

On Exit: AH = Return Status Code
BH = New keyboard state (only if state bit set type)
DH = New scancode type (T__KC__IBM__PC = 0AH)

Registers Altered: AX, BH, DH, BP, DS

SF__VERSION__DESC (AX = 0206H)

This subfunction returns the release date code and a double word pointer to the current version number. The date code consists of two BCD coded bytes containing the year and week of release. The BL register contains the number of years since 1960 and the BH register contains the week of the year.

On Entry: AH = F__SYSTEM (02H)
AL = SF__VERSION__DESC (06H)
BP = V__QWERTY (0036H)

On Exit: AH = Return Status Code
BX = Release date code
CX = Number of bytes in current version number
ES:DI = Pointer to the current version number

Registers Altered: AX, BX, CX, DI, ES, BP, DS

5.3.4.3 V__FUNCTION (BP = 0042H)

This service verifies the Data Type, sets a new Data Type and returns.

F_ISR (AH = 00H)

This function verifies the Data Type, and sets the new one.

On Entry: AH = F_ISR (00H)
BH = Keyboard state (only if state bit set in type)
BL = Scancode
DH = Scancode type (T_KC_IS_FUNCTION = 0CH)
DL = Source vector address
BP = V_FUNCTION (0042H)

On Exit: AH = Return status code
DH = New scancode type (T_KC_IBM_PC = 0AH)

Registers Altered: AX, DH, BP, DS

SF_VERSION_DESC (AX = 0206H)

This subfunction returns the release date code and a double word pointer to the current version number. The date code consists of two BCD coded bytes containing the year and week of release. The BL register contains the number of years since 1960 and the BH register contains the week of the year.

On Entry: AH = F_SYSTEM (02H)
AL = SF_VERSION_DESC (06H)
BP = V_FUNCTION (0042H)

On Exit: AH = Return Status Code
BX = Release date code
CX = Number of bytes in current version number
ES:DI = Pointer to the current version number

Registers Altered: AX, BX, CX, DI, ES, BP, DS

5.3.4.4 V_NUMPAD (BP = 0048H)

The V_NUMPAD service is the scancode translator for the numeric keypad. It verifies the Data Type is correct and maintains the state of the <Num lock> and <ScrLck> keys.

F__ISR (AH = 00H)

Verify Data Type and update state variable.

On Entry: AH = F__ISR (00H)
BH = Keyboard state (only if state bit set in type)
BL = Scancode
DH = Scancode type (T__KC__NUMPAD = 0FH)
DL = Source vector address / 6
BP = V__NUMPAD (0048H)

On Exit: AH = Return status code
BH = New keyboard state (only if state bit set in type)
DH = New scancode type (T__KC__IBM__PC = 0AH)

Registers Altered: AX, BH, DH, BP, DS

SF__VERSION__DESC (AX = 0206H)

This subfunction returns the release date code and a double word pointer to the current version number. The date code consists of two BCD coded bytes containing the year and week of release. The BL register contains the number of years since 1960 and the BH register contains the week of the year.

On Entry: AH = F__SYSTEM (02H)
AL = SF__VERSION__DESC (06H)
BP = V__NUMPAD (0048H)

On Exit: AH = Return Status Code
BX = Release date code
CX = Number of bytes in current version number
ES:DI = Pointer to the current version number

Registers Altered: AX, BX, CX, DI, ES, BP, DS

5.3.4.5 V__CCP (BP = 004EH)

This translator service verifies the Data Type is T__KC__HP__CCP and then passes the ISR Event Record to its parent. By default this translator is mapped to the V__CCPCUR service, alternative mappings are presented in table 5.15.

Table 5.15

V_CCP Driver Mapping Alternatives

Driver Name	Function
V_OFF	Discards the ISR event.
V_RAW	Returns the scancode untranslated.
V_CCPNUM	Translates the cursor control pad scancodes into cursor or numeric key pad scancodes, depending on the <Num Lock> and <Shift> states.
V_CCPCUR	Translates the cursor control pad scancodes into cursor scancodes, regardless of the <Num Lock> and <Shift> states.

F_ISR (AH = 00H)

This function verifies the Data Type and passes the event to its parent.

On Entry: AH = F_ISR (00H)
BH = Keyboard state (only if state bit set in type)
BL = Scancode
DH = Scancode type (T_KC_HP_CCP = 0DH)
DL = Source vector address / 6
BP = V_CCP (004EH)

On Exit: AH = Return Status Code
BL = Translated scancode
BH = New keyboard state (only if state bit set in type)
DH = New scancode type (T_KC_IBM_PC = 0AH)

Registers Altered: AX, BX, DH, BP, DS

SF_INIT (AX = 0200H)

This subfunction is called to initialize the driver. Refer to Section 9 for a complete discussion of the protocol utilized in data space allocation (''last used DS'' passed in register BX).

On Entry: AH = F_SYSTEM (02H)
AL = SF_INIT (00H)
BX = ''Last used DS'' in HP Data Area
BP = V_CCP (004EH)

On Exit: AH = Return Status Code
BX = New "last used DS" in HP Data Area

Registers Altered: AX, BX, BP, DS

SF__VERSION__DESC (AX = 0206H)

This subfunction returns the release date code and a double word pointer to the current version number. The date code consists of two BCD coded bytes containing the year and week of release. The BL register contains the number of years since 1960 and the BH register contains the week of the year.

On Entry: AH = F__SYSTEM (02H)
AL = SF__VERSION__DESC (06H)
BP = V__CCP (004EH)

On Exit: AH = Return Status Code
BX = Release date code
CX = Number of bytes in current version number
ES:DI = Pointer to the current version number

Registers Altered: AX, BX, CX, DI, ES, BP, DS

5.3.4.6 V__OFF Driver (BP = 0009CH)

The V__OFF driver effectively turns off any translator mapped to it. It returns a Return Status Code of RS__DONE, this indicates to the driver which called that all processing is complete, and to return. Returning this status code effectively terminates processing of the scancode.

F__ISR (AH = 00H)

This function sets a return status of RS__DONE and exits.

On Entry: AH = F__ISR (00H)
BH = Keyboard state (only if state bit set in type)
BL = Scancode
DH = Scancode type (any type accepted)
DL = Source vector address / 6
BP = V__OFF (009CH)

```
On Exit:     AH  =  RS_DONE
```

Registers Altered: AX, BP, DS

SF_VERSION_DESC (AX = 0206H)

This subfunction returns the release date code and a double word pointer to the current version number. The date code consists of two BCD coded bytes containing the year and week of release. The BL register contains the number of years since 1960 and the BH register contains the week of the year.

```
On Entry:    AH  =  F_SYSTEM (02H)
             AL  =  SF_VERSION_DESC (06H)
             BP  =  V_OFF (009CH)

On Exit:     AH  =  Return Status Code
             BX  =  Release date code
             CX  =  Number of bytes in current version number
          ES:DI  =  Pointer to the current version number
```

Registers Altered: AX, BX, CX, DI, ES, BP, DS

5.3.4.7 V_RAW Driver (BP = 0090H)

The V_RAW driver sets the data type to T_KC_IBM_PC (0AH) and returns, leaving the scancode untranslated.

F_ISR (AH = 00H)

This function sets a Data Type of T_KC_IBM_PC and a return status of RS_SUCCESSFUL.

```
On Entry:    AH  =  F_ISR (00H)
             BH  =  Keyboard state (only if state bit set in type)
             BL  =  Scancode
             DH  =  Scancode type (any accepted)
             DL  =  Source vector address / 6
             BP  =  V_RAW (0090H)
```

On Exit: AH = Return Status Code
DH = New scan code type (T_KC_IBM_PC = 0AH)

Registers Altered: AX, DH, BP, DS

SF_VERSION_DESC (AX = 0206H)

This subfunction returns the release date code and a double word pointer to the current version number. The date code consists of two BCD coded bytes containing the year and week of release. The BL register contains the number of years since 1960 and the BH register contains the week of the year.

On Entry: AH = F_SYSTEM (02H)
AL = SF_VERSION_DESC (06H)
BP = V_RAW (0090H)

On Exit: AH = Return Status Code
BX = Release date code
CX = Number of bytes in current version number
ES:DI = Pointer to the current version number

Registers Altered: AX, BX, CX, DI, ES, BP, DS

5.3.4.8 V_CCPNUM (BP = 0096H)

The V_CCPNUM driver converts scancodes from the HP cursor control pad to their respective Numeric keypad equivalents. The resultant scancodes will be either numeric or cursor scancodes, depending on the state of the <Num Lock> and <Shift> keys.

F_ISR (AH = 00H)

This function translates the scancode, sets a new Data Type and exits.

On Entry: AH = F_ISR (00H)
BH = Keyboard state (only if state bit set in type)
BL = Scancode
DH = Scancode type (T_KC_HP_CCP = 0DH)
DL = Source vector address / 6
BP = V_CCPNUM (0096H)

On Exit: AH = Return Status Code
BH = New keyboard state (only if state bit set in type)
BL = Translated scancode
DH = New scancode type (T__KC__IBM__PC = 0AH)

Registers Altered: AX, BX, DH, BP, DS

SF__VERSION__DESC (AX = 0206H)

This subfunction returns the release date code and a double word pointer to the current version number. The date code consists of two BCD coded bytes containing the year and week of release. The BL register contains the number of years since 1960 and the BH register contains the week of the year.

On Entry: AH = F__SYSTEM (02H)
AL = SF__VERSION__DESC (06H)
BP = V__CCPNUM (0096H)

On Exit: AH = Return Status Code
BX = Release date code
CX = Number of bytes in current version number
ES:DI = Pointer to the current version number

Registers Altered: AX, BX, CX, DI, ES, BP, DS

5.3.4.9 V__CCPCUR (BP = 008AH)

The V__CCPCUR service converts scancodes from the HP cursor control pad to their respective numpad or cursor control equivalents. The <Shift> key states in the keyboard state variable are adjusted to cancel the effect of the <Num lock> key and force the Numeric keypad to operate in cursor mode. Upon return from this translator chain, the logical keyboard driver generates the appropriate <Shift> scancodes to account for the change to the keyboard state variable.

F__ISR (AH = 00H)

This function translates the scancode to its Numeric keypad equivalent, changes the Data Type to T__KC__IBM__PC, and adjusts the keyboard state variable to force the Numeric keypad into cursor mode.

On Entry: AH = F_ISR (00H)
BH = Keyboard state (only if state bit set in type)
BL = Scancode
DH = Scancode type (T_KC_HP_CCP = 0DH)
DL = Source vector address / 6
BP = V_CCPCUR (008AH)

On Exit: AH = Return Status Code
BH = New keyboard state (only if state bit set in type)
BL = Translated scancode
DH = New scancode type (T_KC_IBM_PC = 0AH)

Registers Altered: AX, BX, DH, BP, DS

SF_VERSION_DESC (AX = 0206H)

This subfunction returns the release date code and a double word pointer to the current version number. The date code consists of two BCD coded bytes containing the year and week of release. The BL register contains the number of years since 1960 and the BH register contains the week of the year.

On Entry: AH = F_SYSTEM (02H)
AL = SF_VERSION_DESC (06H)
BP = V_CCPCUR (008AH)

On Exit: AH = Return Status Code
BX = Release date code
CX = Number of bytes in current version number
ES:DI = Pointer to the current version number

Registers Altered: AX, BX, CX, DI, ES, BP, DS

5.3.4.10 V_SKEY2FKEY (BP = 00A8H)

The V_SKEY2FKEY service translates HP Softkey scancodes into their industry standard function key equivalents. The driver makes no attempt to verify that the scancode passed is in the range for an HP Softkey.

F_ISR (AH = 00H)

This function translates the scancode, sets the Data Type to T_KC_IBM_PC and returns.

On Entry: AH = F_ISR (00H)
BH = Keyboard state (only if state bit set in type)
BL = Scancode
DH = Scancode type (T_KC_HP_SOFTKEY = 0BH)
DL = Source vector address / 6
BP = V_SKEY2FKEY (00A8H)

On Exit: AH = Return Status Code
BL = Translated scancode
DH = New scancode type (T_KC_IBM_PC = 0AH)

Registers Altered: AX, BL, DH, BP, DS

SF_VERSION_DESC (AX = 0206H)

This subfunction returns the release date code and a double word pointer to the current version number. The date code consists of two BCD coded bytes containing the year and week of release. The BL register contains the number of years since 1960 and the BH register contains the week of the year.

On Entry: AH = F_SYSTEM (02H)
AL = SF_VERSION_DESC (06H)
BP = V_SKEY2FKEY (00A8H)

On Exit: AH = Return Status Code
BX = Release date code
CX = Number of bytes in current version number
ES:DI = Pointer to the current version number

Registers Altered: AX, BX, CX, DI, ES, BP, DS

5.3.5 V_8041 Driver (BP = 00AEH)

This driver provides an interface to the HP 8041 keyboard controller chip. It responds to 8041 service requests and Input System logical interrupt requests (F_ISR's) to output scancodes to the 8041 chip. It also provides an application interface to 8041 timer services and switch settings. Table 5.16 contains a function code summary for this driver.

Table 5.16

V__8041 Driver Function Code Summary

Vector Address	Func. Value	Function Equate	Definition
00AEH		V__8041	8041/keyboard interface. provides HP extensions to INT 16H
00AEH	00	F__ISR	Processes ISR event record
00AEH	02	F__SYSTEM	System functions
00AEH	02/00	SF__INIT	Initializes driver
00AEH	02/02	SF__START	Driver Start-up
00AEH	02/06	SF__VERSION__DESC	Reports HP version number
00AEH	04	F__IO__CONTROL	Driver Dependant Functions
00AEH	04/00-08		Reserved
00AEH	04/0A	SF__CREAT__INTR	Create interval entry
00AEH	04/0C	SF__DELET__INTR	Delete interval entry
00AEH	04/0E	SF__ENABL__INTR	Enable interval
00AEH	04/10	SF__DISBL__INTR	Disable interval
00AEH	04/12	SF__SET__RAMSW	Set RAM switch to one (1)
00AEH	04/14	SF__CLR__RAMSW	Set RAM switch to zero (0)
00AEH	04/16	SF__SET__CRTSW	Set CRT switch to one (1)
00AEH	04/18	SF__CLR__CRTSW	Set CRT switch to zero (0)
00AEH	04/1A	SF__PASS__THRU	Pass data byte to 8041

V__8041 Driver Function Definitions

F__ISR (AH = 00H)

This function processes a Keyboard ISR Event Record. It checks to see if the 8041 will accept another scancode. If not, the scancode is placed in a queue. If the 8041 can accept a scancode, it writes the scancode out. The scancode queue has room for 127 entries plus one overrun character.

On Entry: AH = F_ISR (00H)
BH = Keyboard state (only if state bit set in type)
BL = Scancode
CX = Number of scancodes in buffer (string type only)
DH = Scancode type
DL = Source vector address / 6
BP = V_8041 (00AEH)
ES:SI = Pointer to buffer (string type only)

On Exit: AH = Return Status Code

Registers Altered: AX, BP, DS

SF_INIT (AX = 0200H)

This subfunction is called to initialize the driver. Refer to Section 9 for a complete discussion of the protocol utilized in data space allocation (''last used DS'' passed in register BX).

On Entry: AH = F_SYSTEM (02H)
AL = SF_INIT (00H)
BX = ''Last used DS'' in HP Data Area
BP = V_8041 (00AEH)

On Exit: AH = Return Status Code
BX = New ''last used DS'' in HP Data Area

Registers Altered: AX, BX, BP, DS

SF_START (AX = 0202H)

This subfunction starts the 8041 driver.

On Entry: AH = F_SYSTEM (02H)
AL = SF_START (02H)
BP = V_8041 (00AEH)

On Exit: AH = Return Status Code

Registers Altered: AX, BP, DS

SF_VERSION_DESC (AX = 0206H)

This subfunction returns the release date code and a double word pointer to the current version number. The date code consists of two BCD coded bytes containing the year and week of release. The BL register contains the number of years since 1960 and the BH register contains the week of the year.

On Entry: AH = F_SYSTEM (02H)
AL = SF_VERSION_DESC (06H)
BP = V_8041 (00AEH)

On Exit: AH = Return Status Code
BX = Release date code
CX = Number of bytes in current version number
ES:DI = Pointer to the current version number

Registers Altered: AX, BX, CX, DI, ES, BP, DS

SF_CREAT_INTR (AX = 040AH)

The 8041 driver will call up to eight drivers at 1/60 second intervals. This subfunction creates an entry in the table of driver vectors which are called. Note that this subfunction only creates the entry; it does not enable the interval service. This is accomplished with the SF_ENABL_INTR subfunction.

On Entry: AH = F_IO_CONTROL (04H)
AL = SF_CREAT_INTR (0AH)
BH = Vector number (vector address divided by six) of driver requesting service
BP = V_8041 (00AEH)

On Exit: AH = Return Status Code
RS_FAIL indicates driver vector table full.

Registers Altered: AX, BP, DS

SF_DELET_INTR (AX = 040CH)

This function removes the passed in vector number from the interval service table.

On Entry: AH = F_IO_CONTROL (04H)
AL = SF_DELET_INTR (0CH)
BH = Vector number (vector address divided by six) of driver to delete from table
BP = V_8041 (00AEH)

On Exit: AH = Return Status Code
RS_FAIL indicates vector not in table.

Registers Altered: AX, BP, DS

SF_ENABL_INTR (AX = 040EH)

This function enables interrupt service for a driver. The vector number passed is checked against the table. If an entry with that vector number is found, interval service is enabled. When the interval expires all enabled drivers in the list will be interrupted with a function code of F_SYSTEM (02H) in AH and a subfunction code of SF_INTERVAL (14H) in AL.

On Entry: AH = F_IO_CONTROL (04H)
AL = SF_ENABL_INTR (0EH)
BH = Vector number (vector address divided by six) of driver requesting service
BP = V_8041 (00AEH)

On Exit: AH = Return Status Code
RS_FAIL indicates vector not in table.

Registers Altered: AX, BP, DS

SF_DISBL_INTR (AX = 0410H)

This function disables interrupt service for a driver. The vector number passed is checked against the table. If an entry with that vector number is found, interval service is disabled.

On Entry: AH = F_IO_CONTROL (04H)
AL = SF_DISBL_INTR (10H)
BH = Vector number (vector address divided by six) of driver to be disable
BP = V_8041 (00AEH)

On Exit: AH = Return Status Code
RS_FAIL indicates vector not in table.

Registers Altered: AX, BP, DS

SF_SET_RAMSW (AX = 0412H)

This function sets the industry standard extended RAM "switch" in the 8041 status register. This switch indicates that the second 256K RAM bank on the system board is enabled (default condition).

On Entry: AH = F__IO__CONTROL (04H)
AL = SF__SET__RAMSW (12H)

On Exit: AH = Return Status Code

Registers Altered: AX, BP, DS

SF__CLR__RAMSW (AX = 0414H)

This function clears the industry standard extended RAM "switch" in the 8041 status register. When this switch is off it indicates that the second 256K RAM bank is disabled. Since this can never happen in the system this function should never be called.

On Entry: AH = F__IO__CONTROL (04H)
AL = SF__CLR__RAMSW (14H)

On Exit: AH = Return Status Code

Registers Altered: AX, BP, DS

SF__SET__CRTSW (AX = 0416H)

This function sets the industry standard primary CRT "switch" in the 8041 status register. When the switch is set it indicates the primary display is attached to the Multimode graphics adapter (Default condition).

On Entry: AH = F__IO__CONTROL (04H)
AL = SF__SET__CRTSW (16H)

On Exit: AH = Return Status Code

Registers Altered: AX, BP, DS

SF__CLR__CRTSW (AX = 0418H)

This function clears the industry standard primary CRT "switch" in the 8041 status register. When this switch is clear it indicates the primary display is attached to the monochrome display adapter.

On Entry: AH = F__IO__CONTROL (04H)
AL = SF__CLR__CRTSW (18H)

On Exit: AH = Return Status Code

Registers Altered: AX, BP, DS

SF__PASS__THRU (AX = 041AH)

This function outputs the byte in BL to the 8041 using the pass thru command to prevent the 8041 from interpreting the data as a scancode or a command.

On Entry: AH = F__IO__CONTROL (04H)
AL = SF__PASS__THRU (1AH)
BL = data byte to pass thru the 8041

On Exit: AH = Return Status Code

Registers Altered: AX, BP, DS

5.4 8041 Keyboard Controller

5.4.1 Overview

The primary function of the 8041 Keyboard controller is to emulate the industry standard 8042 keyboard interface. (Directly accessing this hardware interface may affect program portability and is not recommended). The 8042 interface, in turn, emulates the keyboard interface of the IBM-PC. The 8041 keyboard controller acts as a command buffer from the STD-BIOS keyboard driver to the Input System while it acts as a loopback buffer for the Input System to the STD-BIOS keyboard driver. The 8041 is implemented in such a way as to maintain standard AT compatibility, while at the same time supporting all of the features of the Input System.

The 8041 keyboard controller accepts commands from the STD-BIOS drivers that control the operation of the controller and the keyboard itself. These commands are detailed in the next subsections of this manual. Some of these commands are executed by the 8041 keyboard controller, but most are passed on the V__8041 interface driver for execution.

When the 8041 keyboard controller receives a command from the system that it cannot execute, it writes that command to its Keyboard Request Service Port (SVC). This port resides in the system I/O port address space at 069H. Whenever a byte is written to this port, the 8041 also generates a hardware interrupt to notify the V__8041 interface driver of the request.

The V__8041 driver reads the 8041 Keyboard Request Service Port, then performs a write to Port 06AH. Any value written to this port sends the 8041 an acknowledgement that the byte has been read, and clears the service request interrupt.

The V__8041 driver then determines if it can execute the command. If it cannot, it calls its child driver, the V__HPHIL driver. The V__HPHIL driver will transmit the command to the keyboard. Examples commands executed by the keyboard are set typematic rate and delay values, set the state of keyboard LEDs, etc.

The keyboard 8041 controller will accept and execute two sets of industry standard commands. One set is controller commands, the other set is keyboard commands, both sets are listed in table 5.17. Controller commands are executed by the 8041 controller or the V__8041 interface driver. Keyboard commands are executed by the keyboard directly. (In actuality, due to the keyboard implementation some of the keyboard commands are implemented by the V__8041 interface driver.)

Each of the command sets has its own protocol. The 8041 has two I/O ports, a command port (I/O address 64H) and a data port (I/O address 60H). Controller commands are written to the command port. If the command has parameters associated with it, the parameters are written to the data port. Keyboard commands are written to the data port. All data written to the data port is interpreted as a keyboard command, unless the previous command written to the command port required parameters.

5.4.2 8041 Controller and Keyboard Commands

There are two sets of commands that are written to the 8041 chip. The first set controls the actions and state of the 8041 Keyboard controller chip. The second set is either passed on to the physical keyboard or emulated by the 8041 controller chip as if it were passed on to the physical keyboard to be executed. 8041 Controller Commands are written to output port 64H. If there is a data byte required by the command then it is written to (or read from) input port 60H. Keyboard Commands, however, are written to output port 60H. Again, if there is a data byte required it is written to output port 60H.

The following code writes a one byte command to the 8041 controller to disable the keyboard interface.

```
hp8041_cmd_port         equ     64h             ; IBM cmd/status port
hp8041_status_port      equ     64h             ; IBM cmd/status port
hp8041_data_port        equ     60h             ; IBM data port
hp8041_jbf_mask         equ     02h             ; Input buffer full mask

hp8041_jface_dis        equ     0ADh            ; Disable interface

dis_8041        proc    near
                push    cx                      ; save working set of regs
                push    ax
                xor     cx,cx                   ; loop 64k times (if necessary)
                cli                             ; ints must be off for this loop

dis_8041_10:
                in      al,hp8041_status_port   ; get status and see if 8041
                test    al,hp8041_jbf_mask      ; input buffer if full
                loopnz  dis_8041_10             ; loop if it is

                mov     al,hp8041_jface_dis     ; load disable command and
                out     hp8041_cmd_port,al      ; ship it out
                sti

                pop     ax
                pop     cx
                ret
dis_8041        endp
```

The following code writes a two byte command to the 8041 to turn on all the keyboard LED's at once.

```
hp8041_cmd_port         equ     64h             ; Hp8041 cmd/status port
hp8041_status_port      equ     64h             ; Hp8041 cmd/status port
hp8041_data_port        equ     60h             ; Hp8041 data port
hp8041_set_led          equ     0edh            ; Set keyboard leds command

hp8041_jbf_mask         equ     02h             ; Input buffer full mask
led_data                equ     07h             ; Led mask to send out
```

```
set_8041        proc    near
                push    cx                      ; save working set of regs
                push    bx
                push    ax
                xor     cx,cx                   ; loop 64k times (if necessary)
                mov     bh,led_data             ; load data for loop
                mov     bl,hp8041_set_led       ; load command
                cli                             ; ints must be off for this loop

set_8041_10:
                in      al,hp8041_status_port   ; get status and see if 8041
                test    al,hp8041_ibf_mask      ; input buffer if full
                loopnz  set_8041_10             ; loop if it is

                mov     al,bl                   ; load command and
                out     hp8041_data_port,al     ; ship it out
                cmp     bh,al                   ; did we output both bytes
                je      set_8041_20             ; yes, skip out
                mov     bl,bh                   ; set up for next iteration
                xor     cx,cx
                jmp     short set_8041_10       ; loop

set_8041_20:
                sti                             ; CHANGE this to restore
                                                ; int flag to previous state
                                                ; instead of on (if needed)

                pop     ax
                pop     bx
                pop     cx
                ret
set_8041        endp
```

Table 5.17 lists the 8041 Controller Commands. These commands are categorized as READ, SNGL, or DBL. READ commands cause the 8041 Controller to place the indicated data byte in it's output buffer, input port 60H, to be read by the 80286. SNGL commands are commands written to output port 64H. DBL byte commands are written to output port 64H with the following data byte being written to output port 60H.

Table 5.17

Controller Commands

Command	Type	Description
020H	READ	Reads byte zero of the 8041's internal RAM. This byte is the last Keyboard Command Sent to the 8041.
021H-03FH	READ	Reads the byte specified by the lower five bits of the command in the 8041's internal RAM. E.g. 8041 Controller command 34H will report contents of the 14H byte of the 8041's RAM.
060H-07FH	DBL	Writes the data byte to the address specified in the low five bits of the command.
0AAH	SNGL	Initiate Self-Test. This command instructs the 8041 to perform a self test. If no errors are detected, 55H is returned in the Data Port.
0ABH	SNGL	Initiate Interface Test. This command instructs the 8041 to test the interface between itself and the keyboard. (Always returns 0 = successful)
0ACH	READ	Diagnostic Dump. The contents of the 8041 internal RAM registers (16 bytes), output port, input port, and status word are sent to the system. All diagnostic data is sent to the system in the same manner as scancodes. (Not supported)
0ADH	SNGL	Disable Keyboard. This command disables the keyboard. Bit 4 of the current command byte will be set to '1' in the 8041. This is equivalent to issuing a command byte with bit 4 set to '1'. Note that this command will have no effect if bit 3 of the command byte is set to '1'.
0AEH	SNGL	Enable Keyboard. This command re-enables the keyboard. Bit 4 of the current command byte is cleared in the 8041. This is equivalent to issuing a command byte with bit 4 set to '0'.
0C0H	READ	Read Input Port. The current value of the input port is returned. Bit 7 indicates the status of the front panel keylock. Bits 0—3 will always be reported as '1'. Bits 4—6 are undefined.
0D0H	READ	Read Output Port. The current value of the output port is returned. See table 5.20 for bit definitions.

Command	Type	Description
0D1H	DBL	Write Output Port. The next byte written to the data port will be written to the 8041 output port. The bit definitions for this port are given in table 5.20.
	WARNING The System Reset bit should not be written low. To reset the system, use the Pulse Output Port command.	
0DDH	SNGL	Disable Address Bit 20. Disables the A20 address of the processor address bit. This is the normal state of this pin the in real addressing mode.
0DFH	SNGL	Enable address Bit 20. Enables the A20 address of the processor address bit. This state is only used in protected mode.
0E0H	READ	Read Test Inputs. This command will output the current state of the 8041 test inputs, T0 and T1. The current state of T0 is stored in bit 0 and T1 in bit 1. Both bits will be reported as '1', unless the keyboard interface is inhibited. Bits 2 through 7 are undefined.
0F0H-0FFH	SNGL	Pulse Output Port. Bits 0—3 of the output port may be pulsed low for approximately 6 microseconds. Bits 0 through 3 the command contain a mask which is interpreted by the 8041 to determine which bits are pulsed. A bit is pulsed if its corresponding mask bit is '0'; if it is '1' its current state is maintained.
	Note The System Reset bit is connected to bit 0. If the system needs to be reset, this command should be used (i.e., the bit should be pulsed, not brought low indefinitely.)	

Table 5.18 indicates the format of the data byte written to the 8041 Controller subsequent to the 8041 Command 20H listed in table 5.17.

Table 5.18

Command Byte Format

Bit	Data	Definition
07H	0	Reserved—must always be 0.
06H		Scancode conversion mode.
	1	The scancodes received from the keyboard are converted into PC/XT scancodes.
	0	Convert to AT scancodes.
05H		Acts as a NOP.
04H		Disable Keyboard. Data will not be sent or received by the keyboard.
	1	Disables the keyboard.
	0	Restore operation.
03H		Inhibit override.
	1	Prevents the keyboard from being disabled via bit 4.
02H		System Flag. The value of this bit is stored as the System Flag Bit. This bit may be read via port 60H.
01H		Reserved—must always be 0.
	1	Instructs the 8041 to issue an OBF interrupt when data is in the output buffer.
	0	Disables this feature.

Table 5.19 indicates the format of the data byte written to the 8041 Controller subsequent to the 8041 Command 'Write Output Port' 0D1H, or read from the 8041 Controller subsequent to the 8041 Command 'Read Output Port' 0D0H.

Table 5.19

Output Port Bit Mask

Bit	Data	Definition
07H-05H		Undefined
04H	1	Output Buffer Full Interrupt (OBF)
03H	1	HP SVC Interrupt
02H	1	HP-HIL Controller AutoPoll
01H	1	A20 Gate
00H	1	System Reset

Table 5.20 lists the Keyboard Commands. These commands are categorized as SNGL or DBL. SNGL commands are commands written to output port 60H. DBL byte commands are written to output port 60H with the subsequent data byte, also, being written to output port 60H. The coding examples given for 8041 Controller commands is similar to the procedure for writing Keyboard Commands. The notable exception being the I/O address 60H is substituted for the I/O address 64H (defined with the equate, hp8041__cmd__port).

Table 5.20

Keyboard Commands

Command	Type	Description
0EDH	DBL	Set/Reset Mode Indicators. The keyboard has three status indicators; <Caps lock>, <Num lock>, and <ScrLck>. This command is used to turn these indicators on and off. After the command is issued, the system must wait for an ACK from the keyboard (see below). When it is received, a second byte is issued to the keyboard. Bits 0—2 represent the <ScrLck>, Num Lock, and <Caps lock>, respectively. Setting their respective bits to 1 turns the indicator on while a 0 turns it off. Bits 3—7 should be set to 0. (See table 5.21)
0EEH	SNGL	Echo. This is a diagnostic tool. When this command is issued, the keyboard returns an EEH.
0EFH-0F2H	SNGL	No Operation (NOP). These codes are reserved for future use. The keyboard will acknowledge these codes, but no other action will be performed.
0F3H	DBL	Set Typematic Rate/Delay. This command sets the values for the typematic rate and delay. The typematic rate is the number of make scancodes per second sent in the typematic (repeat) mode. The delay is the amount of time a key must be held down until it enters the typematic mode. The rate and delay are passed in the next byte after the command. Bits 0 through 4 contain the rate and bits 5 and 6 contain the delay. Bit 7 is unused. The HP8041 chip accepts STD AT typematic commands which are composed of two bits of delay (6,5) and five bits of rate (4-0). The two low order bits of the rate value are stripped off by the 8041 and the result translated into the HP-HIL typematic space. (See tables 5.25 and 5.26)

Command	Type	Description
0F4H	SNGL	Enable. This command enables keyboard action. The keyboard will issue an 'ACK' response, then begin sending scancodes as keys are pressed.
0F5H	SNGL	Default Disable. This command sets the keyboard parameters to their power-on default state and disables the transmission of scancodes. The keyboard will send an 'ACK' response to this command.
0F6H	SNGL	Set Default. This command sets the keyboard parameters to their power-on state and sends an 'ACK' response. the keyboard will continue to transmit scancodes after receipt of this command.
0F7H-0FDH	SNGL	No Operation (NOP). These codes are reserved for future use. The keyboard will acknowledge these codes, but no other action will be performed.
0FEH	SNGL	Resend. This command may be sent to the keyboard whenever an error is detected by the system. This command must be sent before the next scancode is to be transmitted. If the last code sent by the keyboard was a Resend command, the keyboard will send the prior code.
0FFH	SNGL	This command instructs the keyboard to perform its Power-On Reset function. This step takes at least 300 milliseconds, during which the keyboard is disabled.

Table 5.21 indicates the format of the data byte written to the output port 60H subsequent to the Keyboard Command 'Set Mode Indicators' 0EDH.

Table 5.21

Set Mode Indicators Data Byte Format

Bit	Data	Definition
07H-03H		Reserved, should be set to zero
02H		Caps Lock Mode Indicator
	0	Turns off Caps Lock indicator
	1	Turns on Caps Lock Indicator
01H		Num Lock Mode Indicator
	0	Turn off Num Lock indicator
	1	Turn on Num Lock indicator
00H		Scroll Lock Mode Indicator
	0	Turn off Scroll Lock indicator
	1	Turn on Scroll Lock indicator

5.4.3 8041 to STD-BIOS Scancodes and Commands

The keyboard (emulated by the 8041) sends scancodes and commands to STD-BIOS driver system . The scancodes/commands are read from the 8041 Data port (Input Port 60H). Table 5.22 lists the keyboard codes returned by the keyboard. As with the controller commands, some of these codes are initiated by 8041 interface driver and not the physical keyboard on the HP-HIL link.

Table 5.22

8041 to STD-BIOS Scancodes and Commands

Code/ Command	Description
00H	OVERRUN. This code indicates that the 16 character keyboard buffer has overflowed.
01H-77H	Keyboard Scancodes. These represent the keys on the 81H-0F7H keyboard. The scancodes are listed in table 5.4.
0AAH	The 8041 Controller will report this byte when it completes the 8041 Controller's Self Test. This test is executed at Power on and after receiving the Keyboard Command OFFH, Reset. Note: any other byte reported at these times indicates failure.
0EEH	ECHO: this code is sent in response to the keyboard ECHO__COMMAND command, OEEH.
0F0H	Break Prefix code. This code is sent to indicate a key break. This code is followed by the scancode of the key being released. This code will be sent only in the AT scancode set mode.
0FAH	ACK. this code is sent to acknowledge receipt of a command (except Echo and Resend).
0FDH	Diagnostic Failure. This code is sent if a keyboard failure is detected.
0FEH	Resend. This code is sent if the keyboard receives an invalid command or detects an error in the transmission.

5.4.4 8041 to Logical Keyboard Driver Communication

The 8041 acts as an intelligent bi-directional buffer between the logical keyboard driver (Input System) and the INT 09H driver and system software. The INT 09H driver and system software communicate with the 8041 via the command and data ports (I/O addresses 64H and 60H respectively). The 8041 has two additional ports which are used to communicate with the logical keyboard driver.

The output port 068H is used by the logical keyboard driver to transfer data and commands to the 8041 without overlapping with the industry standard keyboard commands. Data such as keyboard scancodes and commands are transmitted in this manner. The HP specific commands are listed in table 5.23.

Table 5.23

HP-Specific Commands to the HP-8041

Keycode Value	Keycode/Command Definition
00H-054H	Industry standard make scancodes. The data byte will be put into an 8041 internal scancode buffer, it will loopback the scancode buffer when the 8041's output port is empty.
80H-0D4H	Industry standard break scancodes. The data byte will be put into an 8041 internal scancode buffer, it will loopback the scancode buffer when the 8041's output port is empty.
055H-077H	HP enhanced keyboard make scancodes. The data byte will be put into an 8041 internal scancode buffer, it will loopback the scancode buffer when the 8041's output port is empty.
0D5H-0F7H	HP enhanced keyboard break scancodes. The data byte will be put into an 8041 internal scancode buffer, it will loopback the scancode buffer when the 8041's output port is empty.
078H	Reserved
079H	Reserved
07AH	Pass through the next data byte written to output port 068H. The data byte will be put into an 8041 internal scancode buffer, it will loopback the scancode buffer when the 8041's output port is empty.
07BH	Set the RAM Switch to '0'.
07CH	Set the RAM Switch to '1' (Default).
07DH	CRT__OFF: Set the CRT Switch to '0'. Indicates the primary display is a monochrome-printer adapter.
07EH	CRT__ON: Set the CRT Switch set to '1'. Indicates the primary display adapter is the Color/Graphics or Multimode adapter (Default).
07FH	HP Reserved
0F8H	ENABLE__AUTOPOLL: Enables the SVC Port request AUTOPOLL__EVENT to be sent to the system. This command allows the 80286 to take over the HP-HIL polling function. The AUTOPOLL__EVENT SVC request is made approximately 60 times a second whenever this command is in effect.
0F9H	DISABLE__AUTOPOLL: Disable the AUTOPOLL__EVENT SVC request.
0FAH-0FEH	Reserved
0FFH	KEYBOARD__OVERRUN: This is passed through as any normal keyboard scancode. This command is sent from the 8041 driver to the logical keyboard to the 8041 chip to indicate the logical keyboard's data buffer was overrun.

To verify that the command has been read, the software can read the IBF bit in the status register of the controller.

The 8041 transfers data and commands to the logical keyboard driver through the SVC (Service) port (I/O address 69H). When data is present on this port, the 8041 issues an interrupt alerting the 8041 interface driver of the data. The 8041 interface driver reads the data from the SVC port, then writes any value to the Acknowledge port (I/O address 6AH) which sends an acknowledge signal to the 8041 and clears the interrupt. Table 5.24 defines the SVC Register request functions.

Table 5.24

SVC Port Request

KBD Command Hex	HP SVC Request Binary		Function
0FFH	yr00	0000	HP Reserved
001H	yr00	0001	RESET__KBD: resets the keyboard to power-on state, clear scancode buffer, flash LED's on then off, and set default typematic rate and delay. At completion the keyboard is enabled.
002H	yr00	0010	Reserved
003H	yr00	0011	AUTOPOLL__EVENT: a programmatic autopoll interval occurred
004H	yr00	0100	Reserved
005H	yr00	0101	Reserved
006H	yr00	0110	SELECT__PC__SET: select the PC compatible scancode set.
007H	yr00	0111	SELECT__AT__SET: select the AT compatible scancode set.
008H	yr00	1000	BUFFER__ROOM: The internal 8041 scancode buffer has available room for scancodes.
0F6H	yr00	1001	DEFAULT__KBD: set default keyboard values: clear scancode buffer, and set default typematic rate and typematic delay (keyboard LED are not affected).
0F5H	yr00	1010	DISABLE__KBD: disables the keyboard: set default values same as DEFAULT__KBD command, except the keyboard is left in the disabled state.
0F4H	yr00	1011	ENABLE__KBD: enables the keyboard, clear scancode buffer, and leave the keyboard in the enabled state.
0F3H	yrld	dttt	SET__TYPEMATIC: set typematic repeat rate and delay before repeat. The lower three bits, 'ttt', is an index which specifies the repeat rate and the bits four and five, 'dd' specifies the delay before the first key is repeated (See tables 5.25 and 5.26).
0EDH	yr01	mmmm	SET__MODE__INDICATORS: turns the keyboard LED's on or off, where 'mmmm' is the led mask A one, '1', will turn on and LED while a '0' will turn the LED off.
	y–	——	When bit seven 'y' is one '1' then logical keyboard is inhibited from writing scancodes into the 8041. When 'y' is zero '0' then the logical keyboard can write scancodes into the 8041.
	-r–	——	Bit six is reserved.

Table 5.25 defines the HP-HIL command and approximate delay before repeat value used for each of the HP8041 delay possibilities. This table assumes an HP-HIL poll rate of 60 cycles per second.

Table 5.25

Typematic Delay Coversion

HP8041 Delay 'dd'	Cursor Pad HP-HIL Command	Delay Period	Non-cursor Pad HP-HIL Command	Delay Period
00b	04H	0.283	02H	0.283
01b	07H	0.483	04H	0.550
10b	0CH	0.817	06H	0.817
11b	0EH	0.950	07H	0.950

Table 5.26 defines the HP-HIL command and approximate typematic rate value used for each of the HP8041 typematic rate possibilities. This table assumes an HP-HIL poll rate of 60 cycles per second.

Table 5.26

Typematic Repeat Rate Conversion

HP8041 Typematic Rate'ttt'	Cursor Pad HP-HIL Command	Typematic Rate	Non-cursor pad HP-HIL Command	Typematic Rate
000b	01H	30.00/sec	01H	30.00/sec
001b	02H	20.00/sec	02H	20.00/sec
010b	03H	15.00/sec	03H	15.00/sec
011b	05H	10.00/sec	05H	10.00/sec
100b	07H	7.50/sec	07H	7.50/sec
101b	09H	6.00/sec	09H	6.00/sec
110b	0BH	5.00/sec	0BH	5.00/sec
111b	0EH	4.00/sec	0EH	4.00/sec

SECTION 6

TABLE OF CONTENTS

SECTION 6. MOUSE

The mouse driver discussed in this section provides the HP Mouse with the Microsoft Mouse (tm) compatible (INT 33H) application interface. There are two additional mouse drivers supplied with the system, the pointer driver (simple mouse) discussed in Section 4 and the cursor key emulator discussed in Appendix G. Some of the terminology in this section is defined in Section 4.

6.1 Overview

The industry standard mouse is accessed through software interrupt 33H. The INT 33H driver receives data from the logical mouse driver (V__LHPMOUSE). If the HP-HIL mouse is present at boot time, V__LHPMOUSE initializes INT 33H to the industry standard interface driver. The industry standard interface supports both a polled mode and interrupt mode of data retrieval. The following data flow outlines the process of mouse data input.

1. The mouse is moved. This causes the physical device to generate input data and interrupt the hardware interface level drivers.

2. The hardware interface level processes the interrupt and passes the data (ISR Event Record) to the logical mouse driver (V__LHPMOUSE).

3. V__LHPMOUSE scales and clips the input data and stores it for the industry standard interface to use.

4. If using polled mode the application must inquire if the data is available. If using interrupt mode the application will be interrupted to notify it that the data is available (via INT 33H).

Steps 1 and 2 above have been discussed in Section 4. Step 3 involves processing the ISR Event Record into the data format used by the INT 33H driver. At this point, if the user has defined and installed an interrupt handler with function F33__SET__USR (0CH), that routine will be called. INT 33H also defines functions to allow the application to poll for mouse data.

The screen modes supported by the mouse driver are shown in table 6.1 The (0,0) origin for the display is in the upper left hand corner of the display. All data reported is in the ranges: 0 to 199 for y-axis and 0 to 639 for the x-axis.

Table 6.1

Video Display Modes Supported

Mode	X range	Y range	Comments
80x25	0..632	0..192	X-axis data is in multiples of 8, y-axis data is in multiples of 8
40x25	0..624	0..192	X-axis data is in multiples of 16, y-axis data is in multiples of 8
320x200	0..638	0..199	X-axis data is in multiples of 2
640X200	0..639	0..199	Reports full range for both axes

6.2 Mouse Driver (INT 33H)

The following section discusses the INT 33H driver. Table 6.2 contains a function summary of the INT 33H driver.

Table 6.2

Mouse Driver Function Code Summary

INT Hex	Function Equate	Function Value	Definition
33H	INT__HPMOUSE		
	F33__INSTALL	00H	Mouse installed flag
	F33__ENABLE	01H	Puts cursor on screen
	F33__DISABLE	02H	Turn off cursor
	F33__REPORT__DATA	03H	Get position/button information
	F33__PUT__CURSOR	04H	Position the cursor
	F33__REPORT__PRESS	05H	Report button press status
	F33__REPORT__RELEASE	06H	Report button release status
	F33__SET__HORIZ	07H	Sets min/max horizontal values
	F33__SET__VERT	08H	Sets min/max vertical values
	F33__GRAPH__CURSOR	09H	Define graphics cursor
	F33__TEXT__CURSOR	0AH	Define text cursor
	F33__MOTION	0BH	Report motion counters
	F33__SET__USR	0CH	Define user subroutine
	F33__ENABLE__LIGHT	0DH	Unsupported
	F33__DISABLE__LIGHT	0EH	Unsupported
	F33__RATIO	0FH	Set pixel movement ratio
	F33__COND__OFF	10H	Define conditional off area
	F33__XTEND__GCSR	12H	Extended sprite graphics entry point
	F33__SPEED	13H	Sets mouse movement doubling
	F33__INQUIRE	6F00H	EX-BIOS mouse driver present

Mouse Driver Function Definitions

F33__INSTALL (AX = 0000H)

This function is called by the application to determine if the mouse is connected to the HP-HIL link. If the mouse is connected, the physical GID driver for the mouse is mapped to the V__LHPMOUSE, and the internal data area is set to its default values. If the mouse is connected a −1 is returned in AX, otherwise a zero is returned.

The default values set are:

cursor position	screen center
internal cursor flag	cursor off
graphic cursor shape/hot spot	arrow/(− 1, − 1)
text cursor	inverting box
user-defined call mask	all zeros
light pen emulation mode	disabled
X axis mickies to pixel ratio	8 to 8
Y axis mickies to pixel ratio	16 to 8
min/max cursor position X axis	0/639
min/max cursor position Y axis	0/199

On Entry: AX = F33__INSTALL (0000H)

On Exit: AX = mouse status
BX = number of buttons

Registers Altered: AX, BX

The following example shows how the mouse driver is called.

```
MOV  AX, F33__INSTALL      ; load function code
INT  INT__HPMOUSE          ; call the driver (33H)
```

F33__ENABLE (AX = 0001H)

This function increments the internal cursor flag. If the flag is 0, the cursor is displayed on the screen. When the cursor is on the screen, moving the mouse will cause the mouse cursor to also move.

On Entry: AX = F33__ENABLE (0001H)

On Exit: None

Registers Altered: None

F33__DISABLE (AX = 0002H)

This function decrements the cursor flag count. If the flag has a non-zero value, the cursor is removed from the display.

On Entry: AX = F33__DISABLE (0002H)

On Exit: None

Registers Altered: None

F33__REPORT__DATA (AX = 0003H)

This function reads the position (x,y) of the mouse and the state of the mouse buttons. The button status is described in table 6.3.

Table 6.3

Mouse Button Status Table

Bit	Data	Button Status Definition
0FH-02H	—	Reserved
01H	0	Right button up
	1	Right button down
00H	0	Left button up
	1	Left button down

On Entry: AX = F33__REPORT__DATA (0003H)

On Exit: BX = button status
CX = x position
DX = y position

Registers Altered: BX, CX, DX

F33_PUT_CURSOR (AX = 0004H)

This function changes the cursor position on the screen. If the new cursor position is within the currently defined limits, the cursor is moved to the new position. If the new position is outside of the limits, the cursor is removed from the screen. The new position of the cursor must be set to values supported by the current screen mode.

On Entry: AX = F33_PUT_CURSOR (0004H)
CX = new x cursor position
DX = new y cursor position

On Exit: None

Registers Altered: None

F33_REPORT_PRESS (AX = 0005H)

This function reports the button press information. The press count button status and cursor position of the last press is returned. The button status is defined in table 6.3. Notice that the position represents the position of the cursor at the last press, and may not reflect the current cursor position. The press count is cleared after the call.

On Entry: AX = F33_REPORT_PRESS (0005H)
BX = button number

On Exit: AX = button status
BX = press count
CX = x position at last press
DX = y position at last press

Registers Altered: AX, BX, CX, DX

F33_REPORT_RELEASE (AX = 0006H)

This function reports the button release information. The release count button status and cursor position of the last release is returned. The button status is defined in table 6.3. Notice that the position represents the position of the cursor at the last press, and may not reflect the current cursor position. The release count is cleared after the call.

On Entry: AX = F33_REPORT_RELEASE (0006H)
BX = button number

On Exit: AX = button status
BX = release count
CX = x position at last release
DX = y position at last release

Registers Altered: AX, BX, CX, DX

F33__SET__HORIZ (AX = 0007H)

This function defines the minimum and maximum horizontal positions reported. If the cursor is outside the new boundary, the cursor is moved just inside the boundary. If the minimum parameter is greater than the maximum parameter, the parameters are swapped.

On Entry: AX = F33__SET__HORIZ (0007H)
CX = minimum position
DX = maximum position

On Exit: None

Registers Altered: None

F33__SET__VERT (AX = 0008H)

This function defines the minimum and maximum vertical positions that are reported. If the cursor is outside the new boundary, the cursor is moved just inside the boundary. If the minimum parameter is greater than the maximum parameter, the parameters are swapped.

On Entry: AX = F33__SET__VERT (0008H)
CX = minimum position
DX = maximum position

On Exit: None

Registers Altered: None

F33__GRAPH__CURSOR (AX = 0009H)

This function defines the graphics cursor or sprite. This allows the programmer to define what the 16 pixel by 16 pixel sprite is to look like. The programmer defines both the AND mask and the XOR mask. The masks must be defined in contiguous memory. You must also pass in the sprite hot spot. The hot spot must be in the range of −16 to 16. The term''hot spot'' refers to the point, inside or outside of the sprite, which positions the sprite. The hot spot origin is defined by the upper left hand corner of the sprite.

```
On Entry:  AX  = F33__GRAPH__CURSOR (0009H)
           BX  = horizontal hot spot
           CX  = vertical hot spot
        ES:DX  = pointer to AND and XOR masks
```

On Exit: None

Registers Altered: None

The following example shows how to define the graphics cursor. The hot spot for the example cursor given is at (5,1).

```
SPRITE  DW   0F9FFH      ; 1111100111111111     "*" marks the
        DW   0F0FFH      ; 11110*0011111111         Hot Spot
        DW   0E07FH      ; 1110000001111111
        DW   0E07FH      ; 1110000001111111
        DW   0C03FH      ; 1100000000111111
        DW   0C03FH      ; 1100000000111111
        DW   0801FH      ; 1000000000011111
        DW   0801FH      ; 1000000000011111
        DW   0000FH      ; 0000000000001111
        DW   0000FH      ; 0000000000001111
        DW   0F0FFH      ; 1111000011111111
        DW   0F0FFH      ; 1111000011111111
        DW   0F0FFH      ; 1111000011111111
        DW   0F0FFH      ; 1111.000011111111
        DW   0F0FFH      ; 1111000011111111
        DW   0F0FFH      ; 1111000011111111
;
; Define the XOR mask
;
        DW   00000H      ; 0000000000000000     "*" marks the
        DW   00600H      ; 00000*1000000000         Hot Spot
        DW   00F00H      ; 0000111100000000
        DW   00F00H      ; 0000111100000000
        DW   01F80H      ; 0001111110000000
        DW   01F80H      ; 0001111110000000
        DW   03FC0H      ; 0011111111000000
        DW   03FC0H      ; 0011111111000000
        DW   07FEOH      ; 0111111111100000
        DW   00600H      ; 0000011000000000
        DW   00600H      ; 0000011000000000
        DW   00600H      ; 0000011000000000
        DW   00600H      ; 0000011000000000
        DW   00600H      ; 0000011000000000
        DW   00600H      ; 0000011000000000
        DW   00000H      ; 0000000000000000
```

```
MOV     AX, F33_GRAPH_CURSOR    ; load the function code
MOV     BX, 5                   ; hot spot at (5,1)
MOV     CX, 1
PUSH    DS
POP     ES                      ; set up the es register
LEA     DX, SPRITE              ; load offset of sprite
INT     INT_HPMOUSE             ; call mouse driver (33H)
```

F33_TEXT_CURSOR (AX = 000AH)

This function defines either a software text cursor, or what the hardware text cursor looks like. The parameter in BX selects the cursor type. When BX equals one, the hardware cursor is defined. When BX equals 0, the software cursor is selected. If the hardware cursor is selected, then the parameters in CX and DX define the first and last scan line of the hardware cursor. If the software cursor is selected, then CX defines the AND mask for the character and attribute bytes. DX defines the new character and attribute bytes.

On Entry: AX = F33_TEXT_CURSOR (000AH)
BX = Cursor Type

Data	Definition
0	Software cursor
1	Hardware cursor

For software cursor:
CX = attribute/character AND mask
DX = attribute/character XOR mask

For hardware cursor:
CX = first scan line
DX = last scan line

On Exit: None

Registers Altered: None

F33_MOTION (AX = 000BH)

This function reads the mouse motion counters. Both the X and Y motions are reported. A positive X motion indicates a movement to the right. A positive Y motion represents a movement to the bottom of the screen. The motion counters are cleared after the function call.

On Entry: AX = F33__MOTION (000BH)

On Exit: CX = X axis count
DX = Y axis count

Registers Altered: CX, DX

F33__SET__USR (AX = 000CH)

This function defines the user-defined subroutine to be called at interrupt service time. The function allows the programmer to select which events the routine is to handle. The condition mask is defined in table 6.4. A call to F33__INSTALL disables this feature.

Table 6.4

User-defined Routine Event Definition

Bit	Value	Definition of Event
0FH-05H	—	Reserved
04H	1	Right button released
03H	1	Right button pressed
02H	1	Left button released
01H	1	Left button pressed
00H	1	Any mouse movement

When the subroutine is invoked, the following information is in the registers:

Register	Data
AX	Event mask which describes the event. The table 6.4 defines the events. A set bit indicates the event.
BX	button state (see table 6.3)
CX	X position
DX	Y position

On Entry: AX = F33__SET__USR (000CH)
CX = condition mask
ES:DX = address of the user defined subroutine

On Exit: None

Registers Altered: None

F33__ENABLE__LIGHT (AX = 000DH)

This function is not currently supported.

F33__DISABLE__LIGHT (AX = 000EH)

This function is not currently supported.

F33__RATIO (AX = 000FH)

This function sets the sensitivity of the mouse movement. Logical mouse movement, in pixels, corresponds to an amount of actual physical device movement, in mickies. This ratio of logical to physical movement specifies the number of pixels to move for some number of mickies. This function allows you to change the ratio to any value in the range 1 to 32767.

On Entry:	AX	=	F33__RATIO (000FH)
	CX	=	mickies to pixels ratio for X axis
	DX	=	mickies to pixels ratio for Y axis

On Exit: None

Registers Altered: None

F33__COND__OFF (AX = 0010H)

This function defines an area on the screen which is considered a fast update area. If the cursor is within this area, then the cursor is removed from the screen, and the area can be quickly updated. If the cursor is not within the specified area, then it is not removed from the screen. After a call to this function is made, a call to F33__ENABLE must always be made to turn the cursor back on. If the upper and lower coordinates are entered in reverse order, the values are swapped.

On Entry:	AX	=	F33__COND__OFF (0010H)
	CX	=	upper x screen coordinate (closest to (0,0))
	DX	=	upper y screen coordinate
	SI	=	lower x screen coordinate (farthest from (0,0))
	DI	=	lower y screen coordinate

On Exit: None

Registers Altered: None

F33__XTEND__GCSR (AX = 0012H)

This function defines the graphics cursor sprite. The new sprite can be larger or smaller than the previous sprite. The maximum size of the graphics cursor sprite is 144. This number is the product of number of scan lines (CH) times the number of bytes (BH*2) the sprite spans. This function allows you to define a sprite similar to F__GRAPH__CURSOR.

```
On Entry:   AX  = F33__XTEND__GCSR (0012H)
            BH  = number of words the sprite spans in X axis
            BL  = hot spot x
            CL  = hot spot y
            CH  = # of scanlines in sprite
         ES:DX  = point to the new sprite masks
```

On Exit: AX = −1

Registers Altered: AX

F33__SPEED (AX = 0013H)

This function sets the minimum distance doubling parameter. This allows you to set the sensitivity such that the physical mouse need not travel as far to go across the entire screen. If the mouse moves the number of mickies defined by this function, then the movement for the mouse is doubled.

```
On Entry:   AX  = F33__SPEED (0013H)
            DX  = minimum to double
```

On Exit: None

Registers Altered: None

F33__INQUIRE (AX = 6F00H)

This function can be used to determine if the mouse driver being used was written by HP.

On Entry: AX = F33__INQUIRE

On Exit: BX = 'HP' (4850H)

Registers Altered: BX

6.3 V__LHPMOUSE Driver (BP = 00CCH)

This section describes the EX-BIOS calls for compatible mouse driver. These functions constitute the interface between this driver and the Input System. Table 6.5 contains a function summary of V__LHPMOUSE.

Table 6.5

V__LHPMOUSE Driver Function Code Summary

Vector Address	Func. Value	Function Equate	Definition
00CCH	00	F__ISR	Logical Interrupt
00CCH	02	F__SYSTEM	System Intrinsics
00CCH	02/00	SF__INIT	Initializes driver
00CCH	02/02	SF__START	Starts driver
00CCH	04	F__IO__CONTROL	I/O control driver functions
00CCH	04/00	SF__MOUSE__COM	BIOS mouse install function
00CCH	04/02	SF__MOUSE__OVERRIDE	Install INT 33H even though mouse is not connected

HP Mouse Driver Function Definitions

F__ISR (AH = 00H)

This function receives an ISR Event Record from physical GID drivers. This function translates the physical event into the logical coordinate system used by the INT 33H mouse driver. This function is responsible for updating the INT 33H data area. This includes calculating the mickies to pixel ratio, updating the motion counters, and displaying the mouse cursor.

On Entry: AH = F__ISR (00H)
DH = Data Type (see Table 4.12)
DL = Physical device driver's vector index.
ES:0 = Pointer to Physical Describe Record.
BP = V__LHPMOUSE (00CCH)

For Button Event:

BX = Button information.

Bit	Value	Definition
0FH-08H	—	Reserved
07H	1	Button up
	0	Button down
06H-00H	—	Button number (0-7)

On Exit: AH = Return Status Code

Registers Altered: AX, BP, DS

SF__INIT (AX = 0200H)

This subfunction is called to initialize the driver. Refer to Section 9 for a complete discussion of the protocol used in data space allocation (''last used DS'' passed in register BX).

On Entry: AH = F__SYSTEM (02H)
AL = SF__INIT (00H)
BX = ''Last used DS'' in HP Data Area
BP = V__LHPMOUSE (00CCH)

On Exit: AH = Return Status Code
BX = New''last used DS'' in HP Data Area

Registers Altered: AX, BX, BP, DS

SF__START (AX = 0202H)

This subfunction starts the driver.

On Entry: AH = F__SYSTEM (02H)
AL = SF__START (02H)
BP = V__LHPMOUSE (00CCH)

On Exit: AH = Return Status Code

Registers Altered: AX, BP, DS

SF_MOUSE_COM (AX = 0400H)

This function is used by the BIOS to initialize INT 33H after MS-DOS has been initialized. This function checks for the presence of a mouse. If the mouse is found then INT 33H, is set up to point to the mouse driver. If no mouse is found, then INT 33H is not initialized.

On Entry: AH = F_IO_CONTROL (04H)
AL = SF_MOUSE_COM (00H)
BP = V_LHPMOUSE (00CCH)

On Exit: AH = Return Status Code

Registers Altered: AX, BP, DS

SF_MOUSE_OVERRIDE (AX = 0402H)

This function is used to set up INT 33H even if there is no mouse present. This function is provided in case an application wishes to map any GID device to the V_LHPMOUSE driver. Since no mouse is connected to the HP-HIL link, the mouse driver will not be installed, thus this function enables you to override what is currently at INT 33H.

On Entry: AH = F_IO_CONTROL (04H)
AL = SF_MOUSE_OVERRIDE (02H)
BP = V_LHPMOUSE (00CCH)

On Exit: AH = Return Status Code

Registers Altered: AX, BP, DS

SECTION 7

TABLE OF CONTENTS

SECTION 7. SERIAL AND PARALLEL I/O

This section covers the ROM BIOS support for the system serial and parallel I/O ports. The ROM BIOS supports up to four parallel ports and up to four serial ports. However, DOS only provides logical devices for three parallel printer ports and two serial ports.

7.1 Overview

The ROM BIOS provides two STD-BIOS drivers that control the serial (INT 14H) and parallel (INT 17H) ports. The functions in these drivers provide a means of setting communication parameters and transmitting data. These drivers have expanded functionality that provide the programmer with the additional ability to set higher baud rates and to transfer strings of data. In addition to these drivers, the print screen driver (INT 05H) will be discussed in this section.

7.1.1 Serial And Parallel Port Addresses

The STD-BIOS data area contains two tables used by the serial and parallel port drivers. The Serial Base Port Address Table contains the base port addresses for the serial ports. The Parallel Base Port Address Table contains the base addresses of the parallel ports. The ROM BIOS checks during SYSGEN for the presence of serial and parallel adapter cards at the addresses listed in table 7.1. When a valid port is found, the base address of that port is placed in the next available entry of the appropriate table. Application programs may add additional parallel ports or serial ports to the port tables. An application program can also replace the values in the table with new ones to support non-standard port addresses. Each table contains space for four entries.

Table 7.1

Serial and Parallel Port Addresses

I/O Address	IRQ	INT
3F8H	4	0CH
2F8H	3	0BH
3E8H	10	72H
2E8H	11	73H
3BCH	—	—
378H	5	0DH
278H	7	0FH

Port addresses are added to the base port address tables in the sequence listed in table 7.1. If the system has only two parallel I/O ports at addresses 378H and 278H then 378H becomes the first entry in the table (Port 0) and 278H becomes the second (Port 1). The potential parallel port at 3BCH would not be Port 0 as it is not present in the system.

The functions supported by the serial and parallel port drivers rely on the values contained in the serial base port address table and the parallel base port address table. The ports are referenced by indexes into the tables (port numbers 0–3).

7.1.2 Print Screen Driver

The print screen driver provides a simple method for application programs and system software to print a copy of the screen contents to the system printer (port 0). The ROM BIOS print screen driver will only print the screen if the display adapter is in one of the alphanumeric modes. Support for printing the screen when in graphics modes is provided by the DOS command GRAPHICS.

7.1.3 Polled and Interrupt Driven Operations

Both the serial and parallel ports on the system may be operated in either a polled or interrupt mode. The drivers in the ROM BIOS only support polled operation. Four system interrupts, 0BH, 0CH, 72H and 73H, are reserved for system serial ports. Two system interrupts, 0DH and 0FH, are reserved for system printers. Application programs and system software may use these interrupts to operate the ports in an interrupt mode.

7.2 Data Structures

The data structures for the serial port, parallel port, and print screen drivers are located in the STD-BIOS data area. The data structures for each of the drivers are discussed separately.

7.2.1 Serial Port Driver Data Structures

The serial port driver uses two data structures in the STD-BIOS data area; a base port address table, and a timeout counter table. The addresses of these data structures are listed in table 7.2. The equipment word in the STD-BIOS data area (40:10H), contains the number of serial and parallel ports configured in the system. The equipment byte can be read by the INT 11H equipment determination function.

Table 7.2

Serial Port Data Structures

Port Number	**Port Address Table Entry**	**Timeout Table Entry**	**Timeout (Default)**
0	40:00H	40:7CH	(01H)
1	40:02H	40:7DH	(01H)
2	40:04H	40:7EH	(01H)
3	40:06H	40:7FH	(01H)

Each serial port is comprised of eight 80286 I/O addresses. The base address of each block of I/O addresses is stored in the base port address table. For more information see *Vectra Technical Reference Manual Volume I*. The table consists of 4 words (8 bytes), one for each of the four possible serial ports. A zero value for any of the words is interpreted by the driver to mean the port is not present.

The second data structure used by the serial port driver is the timeout table. This data structure consists of 4 bytes, one for each of the serial ports. Whenever the driver attempts to read or write data or parameters it reads the status port on the serial port. To prevent an error condition on the serial port from hanging up the system it uses a timeout loop. If a valid status byte cannot be read within the time allotted, the driver will return with a timeout error status code. The length of the timeout is determined by the entries in the timeout table. Each of the four serial ports can be given a different timeout value by an application program.

7.2.2 Parallel Port Driver Data Structures

The parallel port driver uses two data structures that are similar to those used by the serial port driver. The addresses of the parallel base port address and timeout tables are listed in table 7.3.

Table 7.3

Parallel Port Data Structures

Port Number	Port Address Table Entry	Timeout Table Entry	Timeout (Default)
0	40:08H	40:78H	(14H)
1	40:0AH	40:79H	(14H)
2	40:0CH	40:7AH	(14H)
3	40:0EH	40:7BH	(14H)

Each of the parallel ports occupy four I/O addresses. The base or first address of each is contained in the base address table. A zero value for any of the words is interpreted by the driver to mean the requested parallel port adapter is not present.

The parallel printer port driver checks the status of the port before it outputs a character to determine if the printer is busy. To prevent an error condition on the parallel port from hanging up the system, a timeout loop is used. The length of the timeout is determined by the values stored in the timeout table. The timeout values for each of the parallel ports can be set independently of each other.

7.2.3 Print Screen Driver Data Structures

The print screen driver uses a single byte data structure, located at 0040:0100H (see Appendix B). The print screen driver places a status byte at this location, indicating whether or not a print screen operation is underway. The possible values for this status byte are:

Data	Definition
0	The print screen driver has not been called or it completed the previous operation successfully.
1	Printing is in progress.
0FFH	Error occurred during printing.

If this byte indicates a print screen operation is currently in progress, the driver will return. This prevents more than one print screen operation from occurring at the same time.

7.3 Serial Port Driver (INT 14H)

The functions supported by the serial port driver can be divided into two groups; those that set and report communication protocol or status, and those that transmit and receive data. The driver supports nine functions. Four of these functions implement the features of the industry standard INT 14H driver. The remaining five functions are EX-BIOS extensions. The ROM BIOS supports several features not found in the industry standard INT 14H driver. Among these features is the ability to select a communication speed of up to 19.2 K baud per second and the support of block (multi-byte) data transfer.

The following is a list of descriptions for each of the INT 14H functions. A summary of these functions is shown in table 7.4.

Table 7.4

Serial Port Driver Function Code Summary

INT Hex	Function Equate	Function Value	Definition
14H	INT__SERIAL		Serial
	F14__INIT	00H	Initialize Serial Port Parameters
	F14__XMIT	01H	Send Out One Character
	F14__RECV	02H	Receive One Character
	F14__STATUS	03H	Get Serial Port Status
	F14__INQUIRE	6F00H	EX-BIOS present
	F14__EXINIT	6F01H	Initialize serial port (19.2 Kbaud)
	F14__PUT__BUFFER	6F02H	Write a buffer of data
	F14__GET__BUFFER	6F03H	Read a buffer of data
	F14__TRM__BUFFER	6F04H	Read a buffer of data, terminate on specified condition

Serial Port Driver Function Definitions

All of the following functions range check (between 0 and 3 inclusive) the requested port number specified in the DX register. If legal, the function looks up the I/O address contained in the STD-BIOS data area. If the port table entry is non-zero the port is assumed to exist. If the port table entry is zero the function returns without altering any registers.

F14__INIT (AH = 00H)

The initialize function, F14__INIT, sets the baud rate, number of stop bits, parity and character length of the specified serial port. On return it reports the current contents of the line status register and the modem status register of the specified port.

On Entry: AH = F14__INIT (00H)
AL = Port attribute

Bit	Data	Definition
07H-05H	111	9600 baud rate
	110	4800 baud rate
	101	2400 baud rate
	100	1200 baud rate
	011	600 baud rate
	010	300 baud rate
	001	150 baud rate
	000	110 baud rate
04H-03H	x0	no parity
	11	even parity
	01	odd parity
02H	0	1 stop bit
	1	2 stop bits
01H-00H	00	undefined
	01	undefined
	10	7 bit character
	11	8 bit character

DX = Port number (0, 1, 2, 3)

On Exit: AH = Line status (see table 7.5)
AL = Modem status (see table 7.6)

Registers Altered: AX

Table 7.5 defines the Serial Port Line Status.

Table 7.5

Line Status Register Report

Bit	Data	Definition
7	1	Timeout Error (Not applicable on F14__INIT, F14__EXINIT or F14__STATUS)
6	1	Transmit Shift Register Empty
5	1	Transmit Hold Register Empty
4	1	Break Received
3	1	Character Framing Error
2	1	Parity Error
1	1	Overrun Error
0	1	Data Set Ready

Table 7.6 defines the Serial Port Modem Status.

Table 7.6

Modem Status Register Report

Bit	Data	Definition
7	1	Receive Line Signal Detected
6	1	Ring Indicator Line State
5	1	Data Set Ready Line State
4	1	Clear to Send Line State
3	1	Change in Receive Line Detected
2	1	Trailing Edge of Ring Detected
1	1	Change in Data Set Ready
0	1	Change in Clear to Send State

Example:

```
MOV  AH, F14__INIT       ; (AH = 0H)
MOV  AL, 11100111B       ; HP Laserjet factory default
                         ; 9600 baud
                         ; No parity
                         ; 2 stop bits
                         ; 8 bit character
                         ; setting
MOV  DX, 0               ; Port 0 specification
INT  INT__SERIAL         ; Call serial driver (INT 14H)
```

F14__XMIT (AH = 01H)

Transmits a data byte on the serial port specified by the DX register. The function enables the REQUEST-TO-SEND and DATA-TERMINAL-READY signals, and then waits on the DATA-SET-READY, CLEAR-TO-SEND, and REGISTER-EMPTY signals until the character is transferred or a timeout occurs.

On Entry: AH = F14__XMIT (01H)
AL = Data byte to be transmitted
DX = Port number (0, 1, 2, 3)

On Exit: AH = Line status (see table 7.5)
AL = Modem status (see table 7.6)

Registers Altered: AX

Example:

```
MOV  AH, F14__XMIT       ; (AH = 02H)
MOV  AL, 'G'             ; ASCII 'G' character to send
MOV  DX, 0               ; Port 0 specification
INT  INT__SERIAL         ; Call serial driver (INT 14H)
TEST AH, 10000000B       ; Check for error
JNZ  short ERROR__HANDLER
```

F14_RECV (AH = 02H)

This function reads a data byte from the serial port specified by the DX register. The signal DATA-TERMINAL-READY is enabled in the modem control register indicating to the remote device that data can be sent. The modem status register signal DATA-SET-READY and the line status register signal DATA-READY are polled until a data byte is available to read or the timeout count has expired.

On Entry: AH = F14_RECV (02H)
DX = Port number (0, 1, 2, 3)

On Exit: AH = Line status (see table 7.5)
AL = If no error: Data byte received
If error: Null character, zero

Registers Altered: AX

Example:

```
MOV   AH, F14_RECV      ; (AH = 2)
MOV   DX, 0             ; Port 0 specification
INT   INT_SERIAL        ; Call serial driver (INT 14H)
TEST  AH, 10000000B     ; Check for error
JNZ   short ERROR_RECEIVE
```

F14_STATUS (AH = 03H)

This subfunction returns the status of the serial port specified by the DX register.

On Entry: AH = F14_STATUS (03H)
DX = Port number (0, 1, 2, 3)

On Exit: AH = Line status (see table 7.5)
AL = Modem status (see table 7.6)

Registers Altered: AX

F14_INQUIRE (AX = 6F00H)

This function determines whether or not the extended EX-BIOS functions are available. If the EX-BIOS functions are available, the BX register will be set to 4850H (which are the ASCII characters 'HP').

On Entry: AX = F14_INQUIRE (6F00H)
BX = Any value except 4850H ('HP')

On Exit: BX = 'HP'

Registers Altered: AX, BX

Example:

```
MOV   AX, F14_INQUIRE  ; (AH = 6F00H)
XOR   BX, BX           ; Clear out BX
INT   INT_SERIAL       ; Call serial driver (INT 14H)
CMP   BX, 'HP'         ; Check?
JNE   short ERROR_NO_EXTENDED_FUNCTIONS
```

F14_EXINIT (AX = 6F01H)

This function is similar to the STD-BIOS function, F14_INIT, but provides the ability to set a baud rate beyond 9600.

On Entry: AX = F14_EXINIT (6F01H)
BX = Port attributes

Bit	Data	Definition
08H-05H	1000	19200 baud rate
	0111	9600 baud rate
	0110	4800 baud rate
	0101	2400 baud rate
	0100	1200 baud rate
	0011	600 baud rate
	0010	300 baud rate
	0001	150 baud rate
	0000	110 baud rate
04H-03H	x0	no parity
	11	even parity
	01	odd parity
02H	0	1 stop bit
	1	2 stop bits
01H-00H	00	undefined
	01	undefined
	10	7 bit character
	11	8 bit character

DX = Port number (0, 1, 2, 3)

On Exit: AH = Line status (see table 7.5)
AL = Modem status (see table 7.6)

Registers Altered: AX

Example:

```
MOV  AX, F14__EXINIT          ; (AH = 6F01H)
MOV  BX, 0000000100011010B    ; Port attributes
                              ; 19.2 K baud
                              ; parity even
                              ; 1 stop bit
                              ; 7 bit character
MOV  DX,1                     ; Port 1 specification
INT  INT__SERIAL              ; Call serial driver (INT 14H)
```

F14__PUT__BUFFER (AX = 6F02H)

This function transmits data from a buffer as long as there is data in the data buffer and no error is encountered.

For each data byte transferred, the function enables the REQUEST-TO-SEND and DATA-TERMINAL-READY signals, and then waits on the DATA-SET-READY, CLEAR-TO-SEND, and REGISTER-EMPTY signals until the character is transferred or a timeout occurs. The timeout count is reset for each byte transferred.

On Entry: AX = F14__PUT__BUFFER (6F02H)
CX = number of characters in the data buffer
DX = Port number (0, 1, 2, 3)
ES:DI = Pointer to a data buffer of characters

On Exit: AH = Line status (see table 7.5)
AL = Modem status (see table 7.6)

Normal Completion:
CX = Number of bytes transferred successfully
ES:DI = Base of data buffer

Error Completion (bit 7 of AH register non-zero):
CX = Number of bytes transferred successfully
ES:DI = pointer to next byte to be transferred

Registers Altered: AX, CX, DI, ES

Example:

```
STRING      DB       'Hello'
END__STRING:
START:
            MOV   AX, seg STRING              ; set pointer to string
            MOV   ES, AX                      ;
            MOV   DI, offset STRING           ;
            MOV   AX, F14__PUT__BUFFER        ; (AX = 6F02H)
            MOV   CX, END__STRING-STRING      ; length of character
                                              ; string
            MOV   DX, 0                       ; Port 0 specification
            INT   INT__SERIAL                 ; Call serial driver
                                              ; (INT 14H)
            TEST  AH, 10000000B               ; test for errors
            JNZ   short ERROR__PUT__STRING
```

F14__GET__BUFFER (AX = 6F03H)

This function reads characters into the specified data buffer until the buffer is full or a timeout occurs. For each byte, the signal DATA-TERMINAL-READY is enabled in the modem control register indicating to the remote device that data can be sent. The modem status register signal DATA-SET-READY and the line status register signal DATA-READY are polled until a data byte is available to read or the timeout count has expired.

On Entry: AX = F14__GET__BUFFER (6F03H)
CX = maximum buffer size
DX = Port number (0, 1, 2, 3)
ES:DI = Pointer to a data buffer

On Exit: AH = Line status (see table 7.5)

Normal Completion:
AL = last byte read
CX = Number of bytes transferred successfully
ES:DI = Base of data buffer

Error Completion (bit 7 of AH register non-zero):
AL = 0, the null byte
CX = Number of bytes transferred successfully
ES:DI = pointer to next byte to be transferred

Registers Altered: AX, CX, DI, ES

Example:

```
IN__BUFFER    DB     512 DUP (20H)
END__BUFFER:
START:
          MOV   AX, seg IN__BUFFER               ; set pointer to string
          MOV   ES, AX                           ;
          LEA   DI, offset IN__BUFFER            ;
          MOV   AX, F14__GET__BUFFER             ; (AX = 6F03H)
          MOV   CX, END__BUFFER—IN__BUFFER       ; length of character
                                                 ; string
          MOV   DX, 0                            ; Port 0 specification
          INT   INT__SERIAL                      ; Call serial driver (INT 14H)
          TEST  AH, 10000000B                    ; test for errors
          JNZ   short ERROR__PUT__STRING
```

F14__TRM__BUFFER (AX = 6F04H)

This function will read characters into the specified data buffer until any one of the following three conditions occurs:

- The data buffer is filled with characters.

- A character is read which is between the upper bound and the lower bound, inclusive.

- An error or timeout condition is encountered.

For each byte, the signal DATA-TERMINAL-READY is enabled in the modem control register indicating to the remote device that data can be sent. The modem status register signal DATA-SET-READY and the line status register signal DATA-READY are polled until a data byte is available to read or the timeout count has expired. After the data byte is read it is inspected to see if it lies between the two boundary bytes. If the byte is in between the two bytes then the transfer is terminated. This function is useful for transferring logical records.

On Entry:	AX	=	F14__TRM__BUFFER (6F04H)
	BL	=	lower bound of termination character
	BH	=	upper bound of termination character
	CX	=	maximum buffer size
	DX	=	Port number (0, 1, 2, 3)
	ES:DI	=	Pointer to a data buffer

On Exit: AH = Line status (see table 7.5)

Normal Completion Full Transfer:
AL = last byte read
CX = Number of bytes transferred successfully
ES:DI = Base of data buffer

Normal Completion Terminate Character Detected:
AL = last byte read (terminate byte)
CX = Number of bytes transferred successfully
ES:DI = Base of data buffer

Error Completion (bit 7 of AH register non-zero):
AL = 0, the null byte
CX = Number of bytes transferred successfully
ES:DI = pointer to next byte to be transferred

Registers Altered: AX, CX, DI, ES

Example:

```
IN__BUFFER     DB      512 DUP (20H)
END__BUFFER:
START:
               MOV    AX, seg IN__BUFFER                ; set pointer to string
               MOV    ES, AX                            ;
               LEA    DI, offset IN__BUFFER             ;
               MOV    AX, F14__TRM__BUFFER              ; (AX = 6F04H)
               MOV    CX, END__BUFFER—IN__BUFFER        ; length of character
                                                        ; string
               MOV    DX, 0                             ; Port 0 specification
               INT    INT__SERIAL                       ; Call serial driver
                                                        ; (INT 14H)
               TEST   AH, 10000000B                     ; test for errors
               JNZ    short ERROR__PUT__STRING
               CMP    AL, BL                            ; lower bound?
               JL     NOT__BETWEEN
               CMP    AL, BH                            ; upper bound?
               JG     NOT__BETWEEN
NOT__BETWEEN:
```

7.4 Parallel Port Driver (INT 17H)

The parallel port driver provides several functions that support data transfer on the parallel ports and return status. These functions implement the features of the industry standard INT 17H driver and the EX-BIOS extended functions. The EX-BIOS functions implement features not found in the industry standard functions, such as block (multi-byte) data transfer.

The following is a list of descriptions for each of the INT 17H functions. A summary of these functions is shown in table 7.7.

Table 7.7

Parallel Port Driver Function Code Summary

INT Hex	Function Equate	Function Value	Definition
17H	INT__PRINTER		Printer
	F17__PUT__CHAR	00H	Send printer one byte
	F17__INIT	01H	Initialize printer port
	F17__STATUS	02H	Get printer port status
	F17__INQUIRE	6F00H	EX-BIOS present
	F17__PUT__BUFFER	6F02H	Write a buffer to printer port

Parallel Port Driver Function Definitions

The following functions range check (between 0 and 3, inclusive) the requested port address specified in the DX register. If legal, the function looks up the I/O address contained in the STD-BIOS data area. If the port table entry is non-zero the port is assumed to exist. If the port table entry is zero the function returns without altering any registers.

F17__PUT__CHAR (AH = 00H)

This function prints a character on the parallel port. Valid data is set up on the printer interface for at least 900 nanoseconds. If the BUSY signal indicates that the device is busy, it executes an INT 15H function F15__DEV__BUSY. When it returns from F15__DEV__BUSY, the function waits until the BUSY signal indicates the device is not busy. The function generates a 500 nanosecond data strobe and holds the data valid for at least 900 nanoseconds. The function returns with the port status in the AH register.

```
On Entry:  AH = F17__PUT__CHAR (00H)
           AL = Data byte to be transmitted
           DX = Port number (0, 1, 2, 3)

On Exit:   AH = Printer port status (see table 7.8)
```

Registers Altered: AH

Table 7.8 defines the parallel printer port status byte.

Table 7.8

Printer Status

Bit	Data	Definition
7	0	Printer Busy
	1	Printer Not Busy
6	0	Not ready for Data
	1	Data Acknowledged
5	1	Out of Paper
4	0	Printer Offline
	1	Printer On Line (Selected)
3	1	I/O Error
2	x	Not Used
1	x	Not Used
0	1	Printer Error or Timed out

Example:

```
MOV   AH, F17__PUT__CHAR      ; (AH = 00H)
MOV   AL, 'W'                 ; character to print
INT   INT__PRINTER            ; Call printer driver (INT 17H)
TEST  AH, 00000001B           ; test for error?
JNZ   short ERROR__PRINT
```

F17__INIT (AH = 01H)

This function initializes a parallel printer port. It enables the PRINTER-SELECT signal and activates the PRINTER-INITIALIZE signal. The PRINTER-INITIALIZE signal is held active for at least 50 microseconds. The function returns with the printer port status in the AH register.

On Entry: AH = F17_INIT (01H)
DX = Port number (0, 1, 2, 3)

On Exit: AH = Printer port status (see table 7.8)

Registers Altered: AH

Example:

```
MOV   AH, F17_INIT     ; (AH = 01H)
INT   INT_PRINTER      ; Call printer driver (INT 17H)
TEST  AH, 0000001B     ; Test for error
```

F17_STATUS (AH = 02H)

This function returns the status of the specified parallel printer port.

On Entry: AH = F17_STATUS (02H)
DX = Port number (0, 1, 2, 3)

On Exit: AH = Printer port status (see table 7.8)

Registers Altered: AH

F17_INQUIRE (AX = 6F00H)

This subfunction determines whether or not the extended EX-BIOS functions are available. If the EX-BIOS functions are available, the BX register will be set to 4850H (which are the ASCII characters 'HP').

On Entry: AX = F17_INQUIRE (6F00H)
BX = Any value except 4850H ('HP')

On Exit: BX = 'HP'

Registers Altered: AX, BX

Example:

```
MOV   AX, F17_INQUIRE       ; (AH = 6F00H)
XOR   BX,BX                 ; Clear out BX
INT   INT_PRINTER           ; Call printer driver (INT 17H)
CMP   BX,'HP'               ; Check?
JNE   short ERROR_NO_EXTENDED_FUNCTIONS
```

F17__PUT__BUFFER (AX = 6F02H)

This function transmits data from a buffer as long as there is data in the buffer and no error is encountered. Valid data is set up on the printer interface for at least 900 nanoseconds. If the BUSY signal indicates that the device is busy, it executes an INT 15H function F15__DEV__BUSY. When it returns from F15__DEV__BUSY, the function waits until the BUSY signal indicates the device is not busy. The function generates a 500 nanosecond data strobe and holds the data valid for at least 900 nanoseconds. The function returns with the port status in the AH register.

```
On Entry:   AX  = F17__PUT__BUFFER (6F02H)
            CX  = Number of characters in the data buffer
            DX  = Port number (0, 1, 2, 3)
         ES:DI  = Pointer to a data buffer of characters

On Exit:    AH  = Printer port status (see table 7.8)

Normal Completion:
            CX  = Number of bytes transferred successfully
         ES:DI  = Base of data buffer

Error Completion (bit 0 of AH register non-zero):
            CX  = Number of bytes transferred successfully
         ES:DI  = pointer to next byte to be transferred
```

Registers Altered: AH, CX, DI, ES

Example:

```
STRING     DB     'Hello'
END__STRING:
START:
           MOV    AX, seg STRING              ; set pointer to string
           MOV    ES, AX                      ;
           MOV    DI, offset STRING           ;
           MOV    AX, F17__PUT__BUFFER        ; (AX = 6F02H)
           MOV    CX, END__STRING-STRING      ; length of character
                                              ; string
           MOV    DX, 0                       ; Port 0 specification
           INT    INT__PRINTER                ; Call printer driver (INT 17H)
           TEST   AH, 00000001B               ; test for errors
           JNZ    short ERROR__PUT__STRING
```

7.5 Print Screen Driver (INT 05H)

The print screen driver prints the contents of the screen. Each time an INT 05H instruction is executed, the contents of the screen will be printed on the system printer (Port 0). If a print screen operation is already in progress the driver returns without printing the contents of the screen. The print screen driver does not execute functions in the same manner as the other drivers. It performs a single task, so there are no functions.

The print screen driver is called by the keyboard driver (INT 9H) when the scancode (06AH) for the <Prt Sc> key is detected. In addition, application programs may execute an INT 05H instruction any time a copy of the contents of the screen is desired.

The print screen driver can only print the contents of a screen if the display adapter is in one of its alphanumeric modes.

SECTION 8

TABLE OF CONTENTS

SECTION 8. DISC

This section discusses the ROM BIOS disc drivers. The disc driver (INT 13H) provides a set of functions that control the disc drives and data transfer between the disc drives and the system.

8.1 Overview

The disc driver supports three disc types; standard capacity flexible discs (360KB), high capacity flexible discs (1.2 MB), and hard discs. The structure of the disc driver allows additional drives to be easily integrated into the system.

The disc driver consists of two separate code modules; a module that supports flexible disc drives, and one that supports hard disc drives. The code module that provides the flexible disc support is contained in the ROM BIOS that resides on the Processor Extension Card. The code module for the hard disc drive is resident in a ROM on the hard disc adapter card.

8.1.1 Physical Drive Numbers

Each drive in the system has a physical drive number. Physical drive numbers for flexible discs start with 0, while physical drive numbers for the hard disc start with 80H. In a typical system configured with one high capacity flexible disc drive, one standard capacity flexible disc drive, and one 20MB hard disc drive, the physical drive numbers would be 0, 1, and 80H respectively. Flexible disc drives have a one-to-one correspondence between physical drives and volumes. However, hard disc drives may have more than one volume, and consequently more than one physical drive number. The optional 40MB hard disc drive can be configured as two 20MB volumes. A 40MB hard disc will have two physical drive numbers assigned to it (80H and 81H).

Physical voluming of disc drives is not the same as operating system partitions, and the two should not be confused. See the *Vectra MS-DOS Programmer's Reference Manual* for more information on disc partitions.

8.1.2 Flexible Disc Drive Support

The disc driver provides support for both standard and high capacity flexible disc drives. The disc driver supports dual format operation (i.e. reading and writing both types of flexible discs) in the high capacity disc drive(s). The flexible disc drives are supported with six functions that perform read, write, verify, reset, format, and return status tasks.

8.1.3 Hard Disc Drive Support

The system can be configured with an optional hard disc drive. When an internal hard disc drive is added to the system, the disc driver is ''expanded'' to include the functions contained in the BIOS code on the hard disc adapter card.

The hard disc BIOS is integrated into the system during the system generation process (SYSGEN). Early in the SYSGEN process the software interrupt vector INT 13H is initialized to point to the flexible disc driver code module. Later in the SYSGEN process the address space between 0C8000H and 0DFFFFH is searched for option ROM modules. This search is explained in greater detail in Section 10. SYSGEN detects the hard disc option ROM and calls it to initialize. During this initialization process the hard disc driver links into the INT 13H disc driver chain. This process is explained in greater detail later in this section.

When an INT 13H is executed the hard disc code is called first. The hard disc code checks the physical drive number specified. If it is a hard disc drive number (greater than or equal to 80H) the function is executed by the hard disc driver code module. If the physical drive number indicates a flexible disc drive (less than 80H), the hard disc code module passes control to the flexible disc driver code module by executing an INT 40H.

8.1.4 External Disc Drives

External disc drives can easily be added to the system. There are two methods for doing this. The external disc can supply BIOS code in an option ROM and enter the system in a manner similar to the internal hard disc. As an alternative, the system could use a DOS installable device driver.

Discs using installable device drivers can not be used as boot devices, since they are loaded in RAM by the operating system. Further, operating systems other than DOS may not recognize the disc in the system. For more information on installable device drivers consult the *Vectra MS-DOS Programmer's Reference Manual*.

Using the option ROM entry mechanism described in the following section, the external hard disc becomes an integrated part of the system and is treated as if it were an internal drive. The first physical hard disc drive, 80H, can then be used as the system boot device.

8.1.5 INT 13H Disc Chain

The INT 13H disc driver chain is a linked list of disc driver entry points. This chain accesses the BIOS based flexible disc driver and the hard disc driver. This linked list is configured during SYSGEN. The following description specifies how the disc chain is implemented and how it is created in the system.

8.1.5.1 INT 13H Disc Chain Linkage

The hard disc driver is linked into the INT 13H driver chain during the option ROM initialization. The process can be extended by other option ROM based disc adapters. The following is a description of how the HP hard disc ROM BIOS enters the INT 13H device chain during the option ROM initialization process. The relationship between the flexible and hard disc driver code modules is illustrated in figure 8.1.

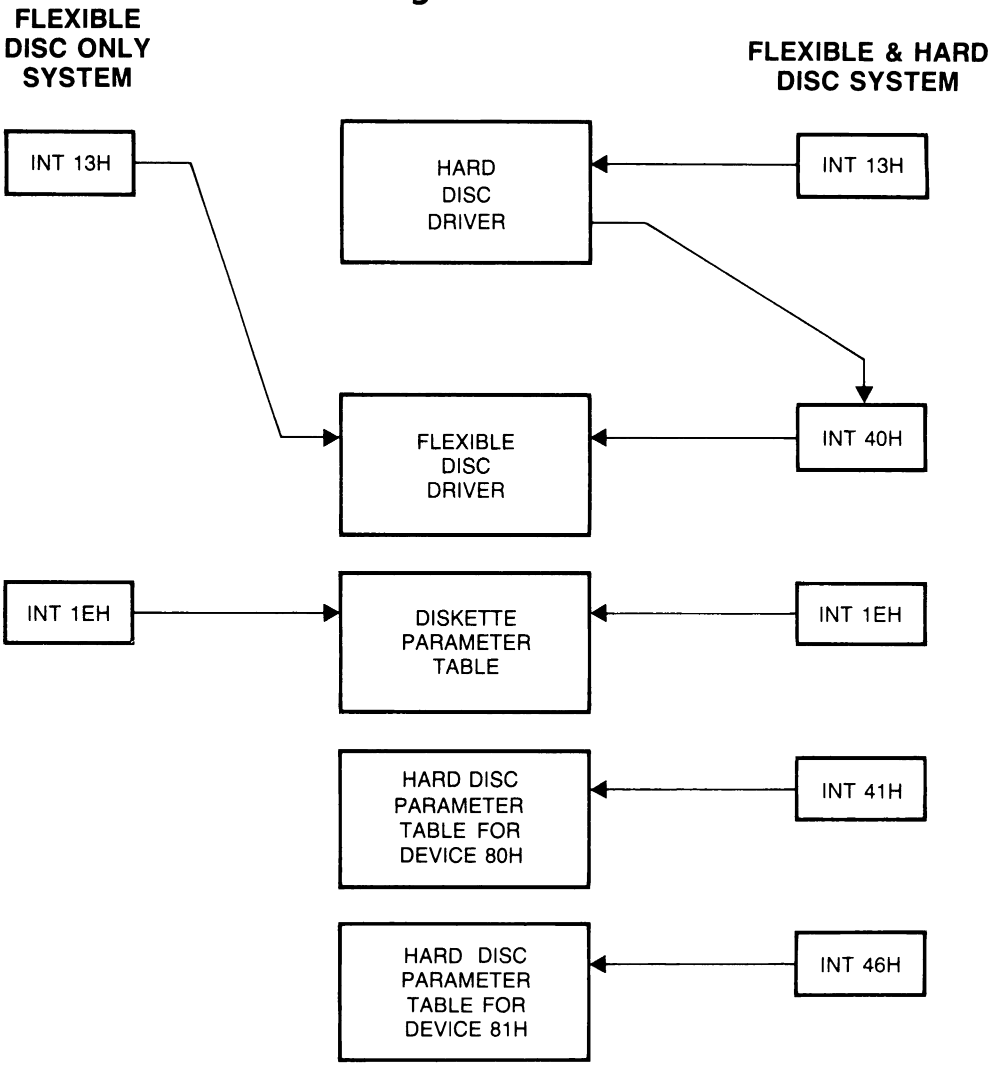
- SYSGEN first configures INT 13H to point to the entry point of the flexible disc driver code.

- SYSGEN detects the hard disc driver's option ROM module and call the initialization entry point of the option ROM.

- The hard disc driver code initializes the disc adapter and the disc drive.

- After the disc adapter hardware is initialized the disc drive is ready to install itself in the INT 13H disc driver chain. The hard disc driver calls the INT 13H function F13__GET__HPARMS (08H) to determine how many other hard discs have entered the system. The lowest hard disc device number (80H) is used for the call. If the call is successful, then the DL register contains the current number, 'n', of hard discs already linked in the INT 13H disc chain. If there are no other hard discs linked into the system, the call will return the status, bad command error, and the current number of drives 'n' is set to 00H.

Figure 8.1


FLEXIBLE DISC ONLY SYSTEM
FLEXIBLE & HARD DISC SYSTEM
INT 13H
HARD DISC DRIVER
INT 13H
FLEXIBLE DISC DRIVER
INT 40H
INT 1EH
DISKETTE PARAMETER TABLE
INT 1EH
HARD DISC PARAMETER TABLE FOR DEVICE 80H
INT 41H
HARD DISC PARAMETER TABLE FOR DEVICE 81H
INT 46H

Flexible and hard disc code modules

- The hard disc driver calculates and saves its own starting device number. The device number is 80H + 'n' where 'n' is the current number of drives determined in the previous step.

- If this is the first hard disc configured in the system then the flexible disc driver address in INT 13H (0:04CH) must be moved to INT 40H (0:100H). If this is not the first hard disc driver then the address of the previously added hard disc driver in INT 13H is saved in RAM for future calls to that previously linked driver.

- The new hard disc driver entry point is loaded into INT 13H. Entry into the chain is complete.

Note

> Many industry standard disc controllers (for example the IBM-PC/XT Fixed Disc Adapter) do not implement the disc drive chain mechanism in exactly the same way. However, the above definitions operate transparent to the industry standard if the HP disc adapter card is set to a memory address greater than IBM-PC/XT compatible adapters.

When the hard disc initialization is complete the system hardware is reconfigured as follows:

- The STD-Slave controller's interrupt is enabled on the master 8259.

- The Hard Disc Interrupt (either the default IRQ 14H or the optional IRQ 15H) is disabled on the STD-slave 8259.

- The diagnostic bit is set in CMOS indicating whether the C: drive (physical device number 80H) is usable as a boot device.

8.1.5.2 Disc Access

When a driver in the INT 13H chain is called by either DOS or an application, the driver should compare the requested device number with the starting physical device numbers it supports. The driver takes one of the following four actions:

- If the requested disc device is supported by this driver then the function is serviced by this driver.

- If the driver is the first hard disc driver in the chain (physical device number 80H) and the requested device is less than 80H then the hard disc driver calls, via INT 40H, the flexible disc driver.

- If the driver is not the first hard disc driver in the chain the driver passes the function on to the next driver in the chain via a PUSHF, CALL FAR instruction combination which simulates an INT instruction. The address was previously saved in RAM during configuration of the chain. The exception to this rule is the F13__GET__HPARMS function which all hard disc drivers execute.

The function F13__GET__HPARMS (08H) returns the total number of hard disc drives in the DL register regardless of an intended specific physical device number requested. Each chained INT 13H hard disc driver checks all commands that are passed through for the F13__GET__HPARMS function. If this function is decoded then the chained driver intercepts the return parameters and adds the number of devices it is serving to the total being accumulated in the DL register.

- If this is the last hard disc driver in the chain and the requested physical device number is larger than this disc driver's number then it will return a bad command error.

8.2 Data Structures

There are separate data structures for the hard disc and the flexible disc drivers. The flexible disc has three data structures. The diskette parameter table holds information necessary for initializing and supporting the NEC flexible disc controller chip. The diskette status table holds information about the status of the previous flexible disc operation. The diskette operation table contains various disc operating parameters such as drive status, flexible disc data transfer rate, etc. The hard disc has only one data structure. However, each hard disc driver maintains it's own copy. The hard disc parameter table is similar to the flexible diskette status table. It contains the physical device characteristics for a particular hard disc attached to the system.

8.2.1 Diskette Operation Table

The diskette operation table is located in the STD-BIOS data area starting at memory location 0040:008BH (0048BH). It contains parameters used by the disc driver to perform its functions. Data stored in this table allow the high capacity drives to read or write either standard or high capacity flexible discs. The contents of the operating parameter table are listed in table 8.1.

Table 8.1

Diskette Operation Table

Offset	Length in Bytes	Description
8BH	1	Data transfer rate of previous operation
8FH	1	Bit 0 is set to '1' for combined 360kb/1.2Mb diskette controller.
90H	2	Current media type table
92H	2	Work area to generate current media types
94H	2	Table of current head positions

8.2.2 Diskette Parameter Table

The diskette parameter table contains information that controls the overall operation of the flexible disc controller. This table is pointed to by INT 1EH (0:78H). A default table is provided in the ROM BIOS at address 0F000H:0EFC7H. The parameters used to control the NEC flexible disc controller can be changed by providing a new diskette parameter table pointer in INT 1EH (0:78H). This is detailed in table 8.2.

Table 8.2

Diskette Parameter Table

Offset	Length in Bytes	Description
00H	1	NEC Specify command byte 1: step-rate time and head unload time
01H	1	NEC Specify command byte 2: head load time and DMA mode
02H	1	Motor wait time
03H	1	Bytes per sector; 0 = 128, 1 = 256, 2 = 512
04H	1	Last sector number on track
05H	1	Read/write gap length between sectors
06H	1	Data length for read/write operations
07H	1	Format gap length between sectors
08H	1	Format filler byte
09H	1	Head settle time after seek command
0AH	1	Motor start time in ⅛ seconds

8.2.3 Diskette Status Table

The status table for the internal flexible disc driver begins at memory location 0040:003EH (0043EH) in the STD-BIOS Data Area. The contents of this table are listed in table 8.3.

Table 8.3

Diskette Status Table

Offset	Length in Bytes	Description
3EH	1	Flag byte
3FH	1	Motor status
40H	1	Motor turn off counter
41H	1	Status of previous diskette operation
42H	7	Status bytes returned by the NEC controller from the previous operation

8.2.4 Hard Disc Parameter Table

The optional hard disc drive has a set of parameters which are quite different from the flexible disc. Therefore, the contents of the hard disc parameter table are not the same as its flexible disc counterpart.

Each hard disc volume has its own disc parameter table. Thus, a system with a 20MB hard disc will have one table, while a system with a 40MB hard disc (configured as two 20MB volumes) will have two tables. The tables do not have a specific location in memory. Instead, two of the system interrupt vectors are used as pointers. These vectors must be initialized to point to the tables by the hard disc BIOS when it is initialized. Interrupt vector 41H contains the address of the first hard disc table while interrupt vector 46H stores the address of the second hard disc table. The contents of the tables are listed in table 8.4.

Table 8.4

Hard Disc Parameter Table

Offset	Length in Bytes	Description
00H	2	Total number of cylinders
02H	1	Total number of Read/Write Heads
03H	2	Reserved
05H	2	Starting cylinder for write precompensation
07H	5	Reserved
0CH	2	Cylinder to use as landing zone
0EH	1	Number of sectors per track
0FH	1	Reserved

8.3 Disc Driver (INT 13H)

The following is a list of descriptions for each of the INT 13H functions. All registers not specified in the exit parameters are returned unchanged. Following the function description is a list of the return status codes used by the INT 13H drivers. A summary of these functions is shown in table 8.5.

Table 8.5

Disc Driver Function Code Summary

INT Hex	Function Equate	Function Value	Definition
13H	INT_DISC		Disc Functions
	F13_RESET_DISC	00H	Reset Disc
	F13_RD_LSTATUS	01H	Read Status of Last Operation
	F13_RD_SECTORS	02H	Read Sectors
	F13_WR_SECTORS	03H	Write Sectors
	F13_VR_SECTORS	04H	Verify Sectors
	F13_FORMAT_FLEX	05H	Format Flexible Disc Track
		06H	Reserved
	F13_FORMAT_HDISC	07H	Format Hard Disc
	F13_GET_HPARMS	08H	Hard Disc Parameters
		09H-0BH	Reserved
	F13_TRACK_SEEK	0CH	Seek to Track
	F13_ALT_RESET	0DH	Alternate Hard Disc Reset
		0EH-014H	Reserved
	F13_GET_DASD	15H	Read Disc Type (DASD)
	F13_CHG_STATUS	16H	Get Disc Change Line Status
	F13_SET_DASD	17H	Disc Type for Formatting (DASD)

Disc Driver Function Definitions

F13_RESET_DISC (AH = 00H)

All discs in the INT 13H device chain are reset. A reset command is issued to each hard disc adapter in the system. For the flexible discs the Read/Write heads are recalibrated back to track 0 and the software services are re-initialized. This call should be used after an error occurs while using the disc. This function does not write on a disc loaded in the flexible disc drive.

On Entry: AH = F13_RESET_DISC (00H)
DL = physical device number

if DL < 80H then reset flexible discs
if DL ≥ 80H then reset all discs

On Exit: AH = return status

Registers Altered: AH

F13__RD__LSTATUS (AH = 01H)

The status of the last disc operation performed is preserved until the next operation occurs. This function will return the value stored as the status of the last operation.

```
On Entry:  AH = F13__RD__LSTATUS (01H)
           DL = device number
                if DL < 80H then return flexible disc status
                if DL ≥  80H return hard disc status
```

On Exit: AH = Status from the last disc operation

Registers Altered: AX

F13__RD__SECTORS (AH = 02H)

Based on the supplied parameters one or more sectors are transferred from the disc into system memory. It is the programmer's responsibility to insure that the data area provided is large enough to contain the requested data. For the hard disc, the maximum data request is 128 sectors (at 512 bytes per sector) or 64KB of data. For the system to transfer the maximum amount of data the programmer must supply a buffer address that is paragraph aligned (address mod16 = 0) otherwise the DMA Boundary error will be returned. For data requests that are less than the maximum there are no addressing restrictions.

For the flexible disc the maximum sector request is the total number of sectors per track. This number varies depending on the drive and media type being used (see the table in the parameter section). Data can only be read from one track at a time. To read data from another track, another read command must be issued with the appropriate parameters.

At least three retries of a flexible disc driver command should be made before an error is indicated. Each retry should be preceded by a reset command, i.e., F13__RESET__DISC.

On Entry: AH = F13__RD__SECTORS (02H)
AL = number of sectors to transfer

For hard discs the sector range is 1–128 assuming 512 byte sectors

For flexible discs the following formats are recognized:

Media	Sector Range
320KB	1–8
360KB	1–9
1.2MB	1–15

DL = device number (Flexible < 80H Hard disc > = 80H)
DH = head number (0–15 not verified)
CH = track/cylinder number (not verified)
For hard discs the high two bits of CL are the MSB of the cylinder number in CH, making a 10 bit value. The valid range is therefore 0–1023. For the flexible discs the valid ranges are:

Media	Track Range
320KB	0–39
360KB	0–39
1.2MB	0–79

CL = sector number (not verified)

For the hard disc the valid value range is 1–17.

For the flexible disc the values in the Sector Range column are also the valid input values for this parameter.

ES:BX = address of data buffer for transfers

On Exit: AH = Return Status Code (See table 8.7)

Registers Altered: AX

Example:

```
                MOV  CX,3                ; retry count
UNTIL_RETRIED:
                PUSH CX                  ; save retry count
                MOV  AH,F13_RD_SECTORS   ; read a sector
                MOV  AL,1                ; transfer 1 sector
                MOV  DL,0                ; Driver A:
                MOV  DH,0                ; head 0
                MOV  CH,0                ; track 0
                MOV  CL,4                ; sector 4
                PUSH CS                  ; use current code segment
                POP  ES                  ; as the segment of the data
                MOV  BX,200H             ; buffer offset 200H
                INT  13H                 ; call disc drivers
                POP  CX                  ; restore retry count
                JNC  NO_ERROR            ; exit, all OK!
                MOV  AH,F13_RESET_DISC   ; reset all drives
                INT  13H                 ; call disc drivers
                LOOP UNTIL_RETRIED       ; loop till no count,no error
                                         ; report error is real to
                                         ; application/user
                .
                .
NO_ERROR:
                .
                .
```

F13_WR_SECTORS (AH = 03H)

This function parallels the read function. Data is written from memory to the disc. See the description of the F13_RD_SECTORS function above.

On Entry: AH = F13__WR__SECTORS (03H)
AL = number of sectors to transfer

For hard discs the sector range is 1–128 assuming 512 byte sectors

For flexible discs the following formats are recognized:

Media	Sector Range
320KB	1–8
360KB	1–9
1.2MB	1–15

DL = device number (Flexible < 80H, Hard disc > = 80H)
DH = head number (0–15 not verified)
CH = track/cylinder number (not verified)
For Hard discs the high two bits of CL are the MSB of the cylinder number in CH, making a 10 bit value. The valid range is therefore 0–1023.
For the flexible discs the valid ranges are:

Media	Track Range
320KB	0–39
360KB	0–39
1.2MB	0–79

CL = sector number (not verified)

For the hard disc the valid value range is 1–17.

For the flexible disc the values in the Sector Range column are also the valid input values for this parameter.

ES:BX = address of data buffer for transfers

On Exit: AH = Return Status Code (See table 8.7)

Registers Altered: AX

F13__VR__SECTORS (AH = 04H)

This function performs a read function without transferring any data. This function ensures that the track and sector can be located on the disc, that the error correction circuitry (CRC) is working correctly and that the data can be read. Again, the discussion for F13__RD__SECTORS applies to this function.

On Entry: AH = F13__VR__SECTORS (04H)
AL = number of sectors to transfer

For hard discs the sector range is 1–128 assuming 512 byte sectors

For flexible discs the following formats are recognized:

Media	Sector Range
320KB	1–8
360KB	1–9
1.2MB	1–15

DL = device number (Flexible < 80H, Hard disc > = 80H)
DH = head number (0–15 not verified)
CH = track/cylinder number (not verified)
For Hard discs the high two bits of CL are the MSB of the cylinder number in CH, making a 10 bit value. The valid range is therefore 0–1023.
For the flexible discs the valid ranges are:

Media	Track Range
320KB	0–39
360KB	0–39
1.2MB	0–79

CL = sector number (not verified)

For the hard disc the valid value range is 1–17.

For the flexible disc the values in the Sector Range column are also the valid input values for this parameter.

On Exit: AH = Return Status Code (See table 8.7)

Registers Altered: AX

F13__FORMAT__FLEX (AH = 05H)

This function writes a pattern of the sectors on a track of the flexible disc. One entire track is formatted at a time, but the programmer can control the characteristics of each sector and the number of sectors in each track. To control the sector variables the programmer supplies a table that has one entry for each sector in the track being formatted. The entries are the sector headers that the drive hardware uses. Also embedded in each entry is a code indicating the desired sector size. (512 bytes is standard).

F13__SET__DASD (AH = 017H), which sets the DASD type, must be called prior to calling the F13__FORMAT__FLEX function. The Set DASD type function will ensure that the internal tables are correctly setup for the media/drive combination desired.

The programmer is also responsible for setting two values in the diskette parameter table. In formatting either 320KB or 360KB media the programmer must set the format gap length to 50H. The End of Track (EOT) value must be set to eight (8) for 320KB media or nine (9) for 360KB media. When the format is complete the programmer should restore the two locations to their original values. The diskette parameter table is described in table 8.2.

On Entry: AH = F13__FORMAT__FLEX (05H)
AL = number of sectors to create
For flexible discs the following formats are standard:

Media	Total Sectors
320KB	8
360KB	9
1.2MB	15

DL = device number (0–1)
DH = head number (0–1 not verified)
CH = track number (not verified)
For the flexible discs the valid ranges are:

Media	Track Range
320KB	0–39
360KB	0–39
1.2MB	0–79

CL = sector number (not verified)

For the flexible disc the values in the Sector Range column are also the valid input values for this parameter.

Media	Sector Range
320KB	1–8
360KB	1–9
1.2MB	1–15

ES:BX = Data buffer containing the values for the sector headers for the track being formatted. Each record is four bytes in length and there must be one record entry for each sector in the track being formatted. The records contain:

(Track,Head,Sector,Length)

Track	=	Current track number
Head	=	Current head number
Sector	=	Sector number
Length	=	Coded sector length 00 = 128 01 = 256 02 = 512 03 = 1024

On Exit: AH = Return Status Code (See table 8.7)

Registers Altered: AH

F13__FORMAT__HDISC (AH = 07)

This function formats the entire hard disc. Once started, this operation cannot be stopped, it must run to completion. This function accepts a table as a parameter that indicates the interleave factor to use for each track of the disc.

On Entry: AH = F13__FORMAT__HDISC (07H)
DL = device number (> = 80H)
ES:BX = Interleave description table

The table is 2*(sectors/track) bytes long. Each table entry is two bytes in length. The entries specify the ordering of the sectors for each track on the disc. The first byte of each entry is reserved and should be set to zero. As an example, a table for seventeen sectors per track with an interleave of two is shown in table 8.6.

Table 8.6

Physical to Logical Sector Conversion

Physical Sector	Logical Sector
1	00H,01H
2	00H,0AH
3	00H,02H
4	00H,0BH
5	00H,03H
6	00H,0CH
7	00H,04H
8	00H,0DH
9	00H,05H
10	00H,0EH
11	00H,06H
12	00H,0FH
13	00H,07H
14	00H,10H
15	00H,08H
16	00H,11H
17	00H,09H

On Exit: AH = Return Status Code (See table 8.7)

Registers Altered: AH

F13__GET__HPARMS (AH = 08H)

This function gets a description of the physical characteristics of one of the hard discs. It also returns the total number of hard discs available through the INT 13H interface.

On Entry: AH = F13__GET__HPARMS (08H)
DL = device number (> = 80H)

On Exit: AH = Return Status
DL = # of drives in system
DH = Maximum head address (Total heads – 1)
CH = Maximum cylinder address (Total cyls – 1)
CL = Maximum sector address (Total sectors/track)
high two bits of CL are the MSB of the cylinder number in CH, making a 10 bit value

Registers Altered: AH, CX, DX

F13__ALT__RESET (AH = 0DH)

This command issues a reset command to the hard disc controller. It is essentially the same as function 0H, F13__RESET__DISC, except that a reset is not issued to the flexible disc units.

On Entry: AH = F13__ALT__RESET (0DH)
DL = device number (> = 80H)

On Exit: AH = Return Status

Registers Altered: AH

F13__GET__DASD (AH = 15H)

This function returns the Direct Access Standard Device (DASD) type code for the attached device. It also returns the total number of sectors for the entire drive if it is a hard disc.

On Entry: AH = F13__GET__DASD (15H)
DL = device number

On Exit: AH = DASD type (if Carry Flag not set)
0 = No drive present
1 = Flexible disc, no disc change line available
2 = Flexible disc, disc change line is available
3 = Hard disc

When AH = 3 the following registers are valid:

CX = Most significant word for total number of sectors on medium
DX = Least significant word for total number of sectors on medium

Registers Altered: AH, CX, DX

F13__CHG__STATUS (AH = 16H)

If the flexible disc drive supports a disc change line then this function reports the status of the disc change line. If the routine indicates that the disc has been changed then the programmer must take the appropriate actions to update the system to use the new media placed in the drive. The 1.2MB drive supports a disc change line.

On Entry: AH = F13__CHG__STATUS (16H)
DL = device number (0-1)

On Exit: AH = 00 = disc change line not active
06 = disc change line active, Carry Flag will be set

Registers Altered: AH

F13__SET__DASD (AH = 17H)

This function must be called before the format function (AH = 05H) can be used to format a flexible disc. Based on the DASD type passed in as a parameter, registers in the flexible disc controller are initialized for the programmer.

On Entry: AH = F13__SET__DASD (17H)
AL = DASD type code
00 = not used
01 = 320KB/360KB media in 320KB/360KB drive
02 = 320KB/360KB media in 1.2MB drive
03 = 1.2 MB media in 1.2MB drive
DL = device number (0-1)

On Exit: AH = Return Status

Registers Altered: AH

8.4 Return Status Codes for INT 13H

There are two status signals returned to an INT 13H programmer. The first is the Carry Flag in the Processor Status Word. If any kind of error occurs this flag is set ("1"). If the function was successful then the Carry flag is cleared ("0").

The second status returned to the programmer is in the AH register. The register will be loaded with one of the return codes shown in table 8.7.

Table 8.7

STD-BIOS Disc Return Status Codes

Hex Value	Indication
000H	Successful execution, no error
001H	Unknown or bad command, bad device number
002H	Address mark could not be found
003H	Attempted to write on write protected disc
004H	Requested sector could not be found
005H	Reset function failed
007H	Initialization failed
008H	DMA overrun, Requested transfer would run over a physical 64KB boundary in RAM (flexible disc)
009H	DMA overrun, Requested transfer would run over a physical 64KB boundary in RAM (hard disc)
010H	Bad CRC encountered on flexible disc read
020H	Controller has failed
040H	Attempted Seek failed
080H	Time out occurred during disc operation
0AAH	Disc Drive reports "Not Ready"
0BBH	Undefined error occurred

SECTION 9

TABLE OF CONTENTS

9. SYSTEM DRIVERS

This section contains a description of the drivers which control the system functions. The drivers discussed in previous sections deal with system peripherals such as the disc drives, keyboard, video display adapter, etc. The drivers covered in this section control the system itself.

9.1 Overview

The system drivers are designed to provide program access to system operating parameters, and to support ROM BIOS drivers. These drivers allow programs to determine the system equipment configuration and amount of memory, provide ''hooks'' for future multi-tasking capability, control vectors in the HP__VECTOR__TABLE, allocate RAM in the EX-BIOS data area, control system strings, manage CMOS memory, and perform system clock functions. An overview of the capabilities of the drivers in each of these categories follows.

9.1.1 Memory Size And Equipment Determination

The ROM BIOS supports two industry standard drivers that report the current system equipment configuration and memory size. These tasks are supported by the INT 11H and INT 12H drivers, respectively.

The equipment determination driver (INT 11H) returns a word that describes the current system configuration. The definition of each bit or group of bits in the word is discussed later in this section. The number of printer ports, serial ports, presence of an 80287 math coprocessor, initial video display mode and number of flexible disc drives are reported by this driver. The default system configuration is read from a CMOS memory location during power-on. If this information does not match the current configuration, a power on error message is issued and the current configuration is saved for the INT 11H driver.

The memory size driver (INT 12H) returns a word that indicates the number of 1 KB blocks of system RAM present. The amount of memory reported does not include any extended memory, and is adjusted to exclude the amount of RAM occupied by the EX-BIOS data area. For example, in a system equipped with 640 KB of system RAM using a 4 KB EX-BIOS data area, the amount of memory reported by this driver will be 636 KB. The default amount of memory is read from a word of CMOS memory.

9.1.2 Extended System Support

The extended system support driver (INT 15H) provides support for several advanced system features. It provides ''hooks'' that allow programs to be written to support multi-tasking at a future date. In addition, it allows data to be transferred to and from extended memory, and allows placing the 80286 into its protected mode of operation.

9.1.3 EX-BIOS Driver Support

The V__SYSTEM driver is an EX-BIOS driver that provides support tasks for the EX-BIOS drivers. It contains functions that allocate RAM in the EX-BIOS data area and manipulate HP__VECTOR__TABLE entries.

9.1.3.1 RAM Allocation

The EX-BIOS data area contains three major data structures; the HP__VECTOR__TABLE, the global data area, and the driver's data area. Within each driver's data area is the driver header, describe record (if applicable), and variable storage area. Each entry in the HP__VECTOR__TABLE is three words long and consists of: Driver's IP, CS, and DS in that order. The HP__ENTRY__CODE (INT 6FH) loads the appropriate driver's data segment DS and jumps to the address CS:IP.

The global data area is used by system drivers that need to share data. Data structures like the EX-BIOS stack and memory management pointers are maintained here.

The driver data area for each driver is dynamically allocated by the V_SYSTEM driver. Each driver's data area is at its data segment (DS) and is generally composed of a standard header followed by any data particular to the driver. If the driver wishes a data area from the EX-BIOS memory it must follow the allocation process described below.

Space is allocated starting from the base of the global data area toward the top of the HP_VECTOR_TABLE as shown in figure 9.1. When a driver is initialized, the base address of the last driver data area ("last used DS") is passed to the driver. The driver decrements this value by the number of paragraphs (16 bytes) it needs for its data area, then returns this value as the new "last used DS".

Driver Data Area Allocation

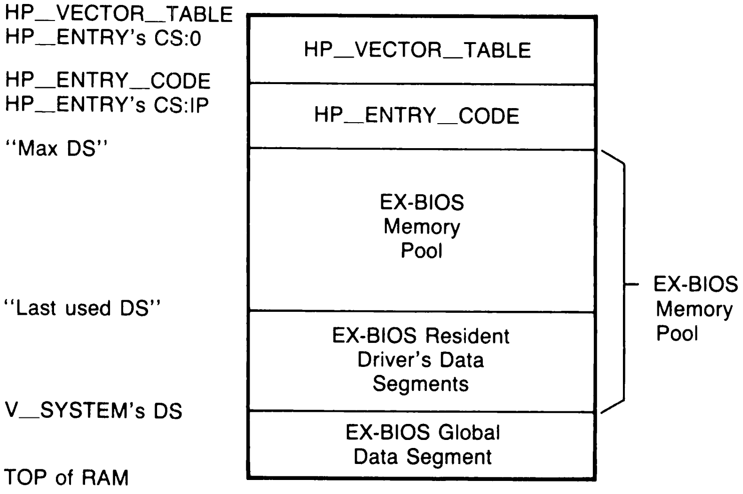


Figure 9.1

If a driver needs a particularly large data area, there might not be enough room. The driver must determine the amount of RAM it requires, then see if that amount is available by comparing its requirements against the amount of RAM available ("last used DS"—"Max DS").

If there is an insufficient amount of RAM available, the driver may increase the amount of RAM allocated to the EX-BIOS data area in the following manner. The memory size stored in CMOS RAM is the amount of physical RAM less the amount occupied by the EX-BIOS Data Area. When the system is booted, the boot code determines the amount of physical memory, then subtracts the "top of memory" stored in CMOS RAM to determine how much space to allocate for the EX-BIOS Data Area. Adjusting the memory size in CMOS RAM downward, then rebooting will increase the size of the EX-BIOS Data Area and hence the amount of RAM available to the driver. This technique may be used to create an EX-BIOS data area up to 64 KBytes in size. A program listing demonstrating this process follows. (Functions F_RAM_GET, F_RAM_RET, F_CMOS_GET and F_CMOS_RET are described in detail later in this section).

Example

```
        MOV     BP, V_SYSTEM            ; How much memory available in
                                        : EX-BIOS data area?
        MOV     AH, F_RAM_GET           ; F_RAM_GET returns:
        INT     HP_ENTRY                ;     BX = "last used DS"
                                        ;     DX = "Max DS"
;
        DEC     BX                      ; Allocate 3 paragraphs (48 bytes)
        DEC     BX                      ; application requires 44 bytes but
        DEC     BX                      ; must allocate in full paragraphs
;
        CMP     BX, DX                  ; New "last used DS"—"Max DS"
        JA      OK
;
NOT_ENOUGH_RAM:
        MOV     BL, 15H                 ; CMOS bytes 16H, 15H contain
                                        ; "top of memory"
        MOV     AH, F_CMOS_GET          ; value (in 1 KB units)
        MOV     BP, V_SYSTEM
        INT     HP_ENTRY                ; Get least significant byte
;
        DEC     AL                      ; Free up 1KB memory for
                                        ; EX-BIOS data area
        PUSHF
        MOV     BL, 15H
        MOV     AH, F_CMOS_RET
        MOV     BP, V_SYSTEM
        INT     HP_ENTRY                ; Store new "top of memory" in CMOS
;
        POPF
        JNC     RESET_PROCESSOR
;
        MOV     BL, 16H                 ; If necessary, decrement most
        MOV     AH, F_CMOS_GET          ; significant byte
        MOV     BP, V_SYSTEM
        INT     HP_ENTRY
        DEC     AL
        MOV     BL, 16H
        MOV     AH, F_CMOS_RET
        MOV     BP, V_SYSTEM
        INT     HP_ENTRY
;
```

```
RESET__PROCESSOR:                             ; Reboot system.
                                              ; This time with 1KB more
        CALL        FAR PTR 0FFFFH:0H         ; memory allocated to the
                                              ; EX-BIOS data area
;
OK:
        MOV         BP, V__SYSTEM             ; Set new "last used DS"
                                              ; and "Max DS"
        MOV         AH, F__RAM__RET           ; Memory is allocated
        INT         HP__ENTRY
         :
         .
```

9.1.3.2 HP__VECTOR__TABLE Manipulation

All drivers in the EX-BIOS code module are accessed through the HP__VECTOR__TABLE. The V__SYSTEM driver provides a set of functions which allows the entries in the HP__VECTOR__TABLE to be set and/or modified. There are nine functions, which represent the permutations of three parameters.

The first parameter determines whether a vector is to be inserted or exchanged with values passed in the 80286 registers. Vectors are typically inserted into the HP__VECTOR__TABLE during the boot process, whereas vector exchanges are used to implement driver mapping. For example, the V__QWERTY keyboard translator driver is installed in the HP__VECTOR__TABLE during the boot process. If keyboard scancodes from the QWERTY keypad were to be mapped to a Dvorak translator, the IP, CS, and DS of the Dvorak translator driver would be exchanged with the existing vector (so the vector could be restored to its original value at a later time).

The second parameter is the vector type. The HP__VECTOR__TABLE has three types of vectors; fixed, reserved, and free. Fixed vectors are those assigned to the default EX-BIOS drivers. The first 51 vectors in the HP__VECTOR__TABLE are fixed. Reserved vectors are set aside for future expansion. There are 24 reserved vectors, which are located at vector addresses 138H through 1C8H inclusive. Free vectors are provided to allow user-supplied drivers to be added to the system.

The final parameter involves the Data Segment (DS) of the driver. Drivers may allocate their data areas from the EX-BIOS data area as explained above, they may provide their own, or use the global data area of the EX-BIOS. The EX-BIOS drivers all use the DS allocation functions, while an external driver (for example, one installed as an MS-DOS device driver) may supply their own data area external to the EX-BIOS data area. Drivers supplying their own DS must pass it as a parameter to V__SYSTEM when the driver has completed initialization.

9.1.3.3 System String Control

The EX-BIOS provides a centralized and flexible mechanism for accessing and using strings. Each string in the system has a unique index number associated with it. Drivers and application programs can request access to a string via these indices. In addition, functions are available to return the index of a given string, return the next available index, and to add and delete strings from the system.

A string index may be any word value (0—0FFFFH). Certain ranges of indices have predefined meanings or uses. These predefined ranges are listed below.

0—2K Any index in this range is reserved for string names of EX-BIOS drivers.
2—4K This range is reserved for strings stored in the ROM-BIOS.
4—32K This range should be used by application programs to add strings to the system.
32—64K These indices are reserved for localized strings. Indices within this range are partitioned in the same way as in the lower 32K (i.e., 32—34K for string names of EX-BIOS drivers, etc.).

This index structure provides a powerful tool for localizing application programs. If an application program references messages as string indices, the program can easily be localized by loading a localized set of strings (using a device driver for example), and setting bit 15 of all string indices used.

System strings are grouped into buckets. A bucket is a collection of strings which are grouped together. There is no fixed limit on the number of strings which may be stored in a bucket. However, strings are added and deleted in buckets, not individually. Therefore, strings that are likely to be added or deleted together should be stored in the same bucket.

Each bucket consists of three separate data structures; the bucket header, bucket pointers, and the bucket itself. These components are illustrated in figure 9.2. The function of each is described below:

> *Bucket Header*—The bucket header is the top level data structure. All bucket headers are linked together in a chain. The first two fields in the header contain the offset and segment of the next bucket header in the chain. If these fields both contain 0FFFFH, then this bucket header is the last in the chain. The highest and lowest string indices contained in the bucket are stored in the next two fields. The following two fields contain the offset and segment of the bucket pointer. Finally, the last field contains the segment of the strings themselves.

Bucket Pointer—The bucket pointer consists of a series of offsets to the strings in the bucket. There must be one offset for every index in the range specified in the bucket header. The actual address of the string is determined by the segment (which is stored in the bucket header) and the offset stored in the bucket pointer. Note that all strings in a bucket must be in the same segment.

Bucket—The bucket contains the actual strings. Each string consists of a byte containing the number of characters in the string, the string itself, and a null byte (00H) which serves as a string terminator.

String control is accomplished through the appropriate functions in the V__SYSTEM driver. These functions provide complete control over system strings.

9.1.4 CMOS Memory Control

The system contains a CMOS Memory/Clock chip that serves as a real-time clock and provides 64 bytes of non-volatile memory storage. The CMOS RAM is used to store system parameters. The contents of the CMOS RAM are listed in Appendix C.

The CMOS Memory/Clock is accessed through two I/O ports. One port selects the clock register or memory byte to access, and the other is a bidirectional data port. There are a total of 64 addresses in the CMOS Memory/Clock chip; the first 14 are the clock registers, while the remaining 50 are the CMOS RAM.

The V__SYSTEM driver contains two functions which support reading and writing data to the CMOS Memory/Clock. These functions provide a simple access to the contents of the chip.

9.1.5 System Clock Functions

The system employs two separate clock systems to keep track of the time and date. The first is the CMOS Memory/Clock. The CMOS clock has a battery back-up which allows it to keep track of the current time when the system is turned off.

System String Data Structures

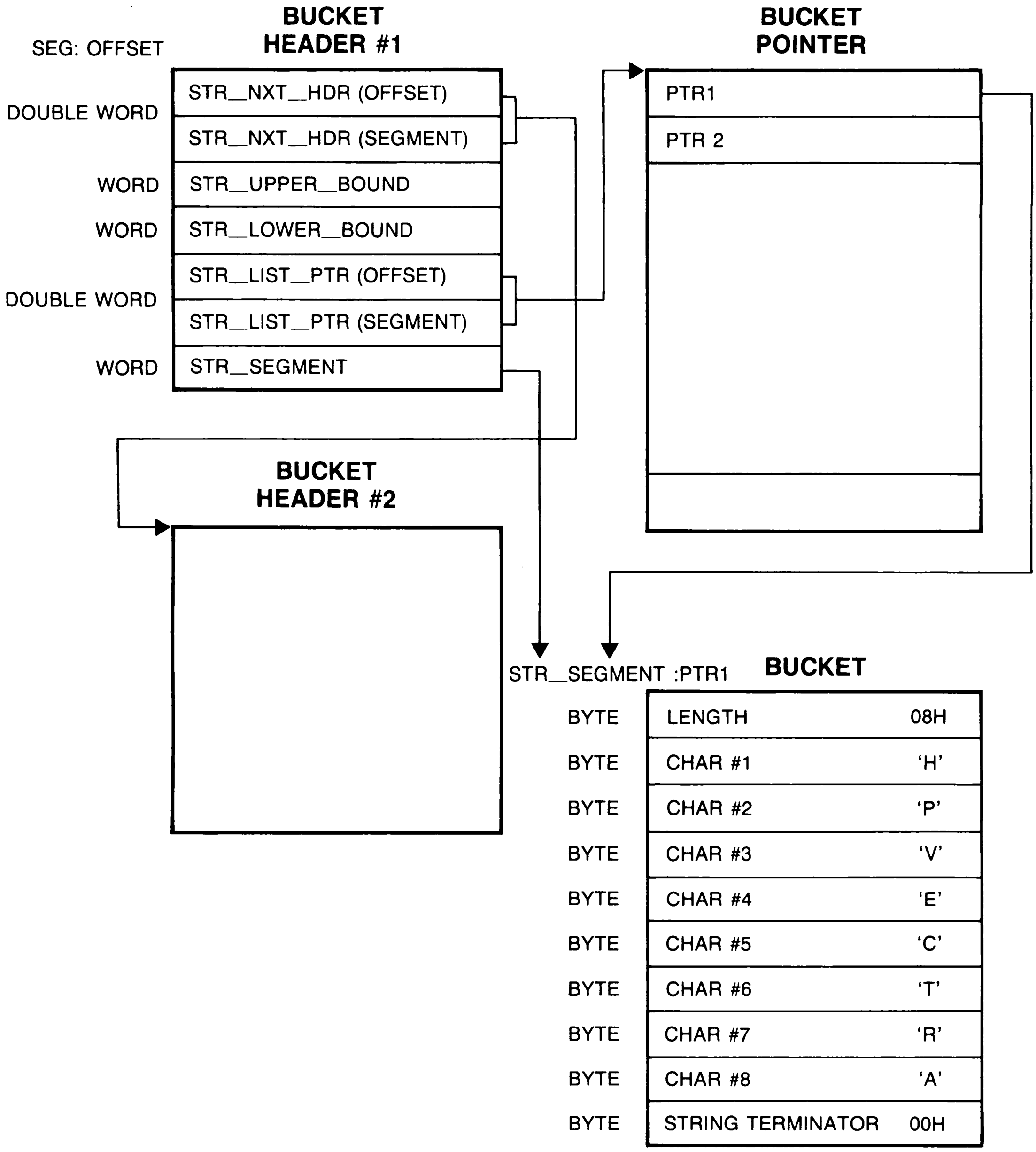


Figure 9.2

The second clock is a software clock. It uses Channel 1 of the 8254 counter/timer chip (refer to the *Vectra Technical Reference Manual, Volume I* for additional details). Channel 1 of the 8254 generates a hardware interrupt (IRQ 0) approximately 18.2 times per second. The ROM BIOS keeps time by incrementing a software clock each time the interrupt occurs. The software clock is used by the operating system for such tasks as time and date stamping of files.

The two clocks operate independently except at boot time. During the boot process the current time and date maintained by the CMOS clock is read and used to initialize the software clock. Changing the value of CMOS clock will not affect the software clock until the system is rebooted.

The STD-BIOS clock driver (INT 1AH) provides a convenient way to read or set the time and date from either of the system clocks. These functions are detailed later in this chapter.

In addition to keeping time, both clocks issue interrupts that call user or application program routines. The software clock interrupt service routine performs an INT 1CH each clock tick. If this vector is modified to point to a user routine, the routine will be called on each clock tick.

The CMOS clock has an ''alarm clock'' feature. It can be programmed to issue an interrupt at a specified time. The real-time clock hardware issues an INT 4AH each time the alarm timer is done. The interrupt 4AH vector can be modified to point to a user-supplied routine.

9.2 Data Structures

The system drivers use several data structures. The data structures for the STD-BIOS system drivers are contained in the STD-BIOS data area, while those used by the EX-BIOS drivers are in the EX-BIOS data area.

The STD-BIOS system drivers use four data structures. The memory size and equipment determination drivers each use a word, the ROM software clock uses five bytes. These data structures are located at 040:13H, 040:10H, and 040:6CH respectively. The extended system support driver uses 9 bytes starting at location 040:98H. The EX-BIOS system drivers use the global data area. These data structures are described in detail in Appendix B.

9.3 Equipment Determination Driver (INT 11H)

Returns information about the equipment attached to the system.

On Entry: No Inputs.

On Exit: AX = Word with all equipment information:

Bit	Value	Definition
15, 14		Number of printers attached.
13, 12		Not used.
11, 10, 9		Number of datacomm cards attached.
8		Not used.
7, 6		Number of diskettes attached:
	00	1 drive,
	01	2 drives, only if Bit 0 is also a 1
5, 4		Initial video mode selected:
	00	Other.
	01	40x25 color adapter.
	10	80x25 color adapter.
	11	80x25 monochrome adapter.
3, 2		Not used.
1		Math co-processor attached.
0	01	Diskette drives attached.

Registers Altered: AX.

9.4 Memory Size Determination Driver (INT 12H)

Returns the amount of RAM found in the system during the power-on and initialization routines.

On Entry: No Inputs.

On Exit: AX = Number of 1KB memory blocks found.

Registers Altered: AX

9.5 System Support Driver (INT 15H)

The extended system support driver (INT 15H) provides functions which allow data to be transferred to and from extended memory and allow placing the 80286 into its protected mode of operation. These functions are listed in table 9.1.

Table 9.1

System Support Driver Function Code Summary

INT Hex	Function Equate	Function Value	Definition
15H	INT__SYSTEM		System Functions Interrupt
		0-3	Unsupported
	F15__DEVICE__OPEN	80H	Device Open
	F15__DEVICE__CLOSE	81H	Device Close
	F15__PROG__TERM	82H	Program Termination
	F15__WAIT__EVENT	83H	Event Wait
	F15__JOYSTICK	84H	Joystick Support
	F15__SYS__REQ	85H	System Request Key Pressed
	F15__WAIT	86H	Wait Fixed Amount of Time
	F15__BLOCK__MOVE	87H	Move Block of Memory to/from Extended Memory
	F15__GET__XMEM__SIZE	88H	Get Extended Memory Size
	F15__ENTER__PROT	89H	Switch to Protected Mode
	F15__DEV__BUSY	90H	Device Busy Hook
	F15__INT__COMPLETE	91H	Set Interrupt Completed Flag

System Support Driver Function Definitions

F15__DEVICE__OPEN (AH = 80H)

Open device for I/O. This is a hook for multitasking systems. Currently the function just returns.

On Entry: AH = F15__DEVICE__OPEN (80H)
BX = Device Identifier
CX = Process Identifier

On Exit: No values returned.

Registers Altered: None.

F15__DEVICE__CLOSE (AH = 81H)

Close device for I/O. This is a hook for multitasking systems. Currently the function just returns.

On Entry: AH = F15__DEVICE__CLOSE (81H)
BX = Device Identifier
CX = Process Identifier

On Exit: No values returned.

Registers Altered: None

F15__PROG__TERM (AH = 82H)

Terminate Program. This is a hook for multitasking systems. Currently the function just returns.

On Entry: AH = F15__PROG__TERM (82H)
BX = Device Identifier.
CX = Process Identifier.

On Exit: No register modified.

Registers Altered: None

F15__WAIT__EVENT (AH = 83H)

Allows a process to wait for at least "x" microseconds before it continues. The process is notified that the requested amount of time has elapsed when the high bit at ES:BX is set to "1". If another process is already using this function, driver returns with the carry set. If the return status is successful (carry flag is clear) the process should poll the byte at ES:BX until the high bit is set.

On Entry: AH = F15_WAIT_EVENT (83H)
AL = Subfunction:
0 = Set the timer with the data passed in ES, BX, CX and DX registers.
1 = Cancel the current timer.
ES:BX = The byte at this address will have its high bit set as soon as possible after the "x" microseconds.
CX,DX = Minimum number, "x", of microseconds to wait before setting the high bit of the address above. CX is the most significant word.

On Exit: Carry = 1 If there was another process already waiting.
0 If the calling process will be notified after the time out.

Registers Altered: AX

F15_JOYSTICK (AH = 84H)

Read data from the joystick port.

On Entry: AH = F15_JOYSTICK (84H)
DX = Subfunctions
0 = Read the switch settings.
1 = Read resistive inputs.

On Exit: Carry Flag = 0 If no errors
1 If invalid DX or no adapter present.

If DX was 0, AL bits 7..4 contain switch positions.

If DX was 1, AX = X position of joystick 1
BX = Y position of joystick 1
CX = X position of joystick 2
DX = Y position of joystick 2

Registers Altered: AX, BX, CX, DX

Programming Example: To read all the data from the joystick adapter (switches and both joysticks).

```
MOV     AH, F15_JOYSTICK     ; Function 84H
MOV     DX, 00               ; Read the switch settings first
INT     INT_SYSTEM           ; Int 15H
JC      HANDLE_ERRORS
```

```
                MOV     SWITCH__STATE,AL       ; Save the state of the switches
                                               ; Bits 7..4 in AL.
                MOV     AH, F15__JOYSTICK      ; Call it again for joystick info
                MOV     DX, 01
                INT     INT__SYSTEM
                JC      HANDLE__ERRORS
                MOV     STICK1__X, AX          ; Save x and y position for both
                MOV     STICK1__Y, BX          ; joysticks.
                MOV     STICK2__X, CX
                MOV     STICK2__Y, DX
                        .
                        .                      ; Continues normally here
                        .
HANDLE__ERRORS:
                        .
                        .                      ; Error handler here
                        .
```

F15__SYS__REQ (AH = 85H)

This subfunction gets called by the keyboard interrupt handler (INT 9H) whenever the user presses the <Sys req> key. Currently the routine just returns but an application can trap this function to detect when the user presses this key.

On Entry: AH = F15__SYS__REQ (85H)
AL = 00, If user pressed the <Sys req> key down (make).
01, If user let go of the <Sys req> key (break).

On Exit: No values returned.

Registers Altered: None.

Example: Link into the current <Sys req> handler so that it prints "HELLO" everytime the <Sys req> key is hit.

```
INITIALIZATION__CODE:
            MOV     AH, 35H                ; Get the old INT 15H
            MOV     AL, INT__SYSTEM        ; Get CS:IP of INT 15H
            INT     21H                    ; This MSDOS Int does the work
            MOV     OLD__SEG, ES
            MOV     OLD__OFFSET, BX
            MOV     AH,25H                 ; Replace old INT 15H with
            MOV     AL, INT__SYSTEM        ; our routine
```

```
                    PUSH     CS
                    POP      DS
                    MOV      DX, offset OUR__INT15
                    INT      21H                      ; This MSDOS Int does the work
                             .
                             .
                             .
OUR__INT15:
                    CMP      AH, F15__SYS__REQ        ; See if it is function 85H?
                    JNE      DO__OLD__INT
                    PUSHA
                    PUSH     ES
                    MOV      AX, F10__WRS__01         ; Yes, call video "write string"
                    MOV      BL, 07                   ; function 1301H to write the
                    MOV      CX, 05                   ; string "HELLO"
                    MOV      BH, 00                   ; page 0
                    MOV      DX, 00                   ; row 0, column 0
                    PUSH     CS
                    POP      ES
                    MOV      BP, Offset HELLO__STR
                    INT      INT__VIDEO               ; Video function interrupt 10H
                    POP      ES
                    POPA
                    IRET
DO__OLD__INT:
                    PUSH     OLD__OFFSET              ; No, just go to regular routine.
                    PUSH     OLD__SEG
                    RET
HELLO__STR          DB       "HELLO"
```

F15__WAIT (AH = 86H)

Calling this function waits the specified number of microseconds (CX,DX) before returning to the caller.

On Entry: AH = F15__WAIT (86H)
CX,DX = Number of microseconds to wait. CX is the most significant word.

On Exit: Carry = 1, Some other process already waiting. So could not wait.
Carry = 0, Waited the amount of microseconds specified in the CX,DX register pair.

Registers Altered: None.

Example: Wait 10 seconds in a procedure.

```
            MOV         AH, F15_WAIT          ; 86H function
            MOV         CX, 0                 ; 10 * 1000 microseconds =
            MOV         DX, 10000             ; 10 seconds
            INT         INT_SYSTEM            ; INT 15H
            JC          HANDLE_ERRORS
                        .
                        .                     ; At least 10 seconds have elapsed
HANDLE_ERRORS:
                        .                     ; Do what's appropriate here.
```

F15_BLOCK_MOVE (AH = 87H)

Moves a block of memory from one location to another anywhere in the 16 megabyte addressing space of the 80286 processor. The number of words to move is passed in CX and the source and destination tables pointers are passed in a Global Descriptor Table (GDT) pointed to by ES:SI. The following data structure describes a sample GDT:

```
ADDRESS_DATA    STRUC
RESERVED_GDT    DB      8 DUP (?)     ; Descriptor used during move
CALLERS_GDT     DB      8 DUP (?)     ; Caller's GDT's during move
SOURCE_GDT      DB      8 DUP (?)     ; GDT describing source
DEST_GDT        DB      8 DUP (?)     ; GDT describing destination
BIOS_GDT        DB      8 DUP (?)     ; GDT of the BIOS routines
STACK_GDT       DB      8 DUP (?)     ; Stack's GDT.
ADDRESS_DATA    ENDS
```

The eight byte descriptor for source or destination has the following format:

```
SAMPLE_GDT       STRUC
SEG_LIMIT        DW      ?    ; Segment Limit
LOW_WORD         DW      ?    ; Low word of 24 bit address
HIGH_BYTE        DB      ?    ; High byte of 24 bit address
ACCESS_RIGHT     DB      ?    ; Segment access rights
                              ; should always be 93H
RESERVED_WORD    DW      ?    ; Reserved.
SAMPLE_GDT       ENDS
```

On Entry: AH = F15_BLOCK_MOVE (87H)
ES:SI = Pointer to descriptor tables.
CX = Number of words to move.

On Exit: AH = Return Status:
0, If successful.
1, If RAM parity error.
2, If exception interrupt error.
3, If gate address line 20 failed.

Carry Flag = 1, If failure.
Zero Flag = 1, If successful.

Registers Altered: AX

Example: Move the 16KB video buffer to the procedure's buffer.

```
            MOV     SI, offset DEST              ; Load table with 24 bit
                                                 ; destination address:
            MOV     BX, seg BUFFER               ; Isolate high nibble of segment
            AND     BX, 0F000H
            SHR     BX, 12
            MOV     AX, seg BUFFER               ; isolate rest of segment
            SHL     AX, 4
            ADD     AX, offset BUFFER            ; and form 24 bit address
            JNC     SKIP_INC
            INC     BX
SKIP_INC:
            MOV     BYTE PTR HIGH_BYTE[SI], BL
            MOV     WORD PTR LOW_WORD[SI], AX

            LES     SI, ACTUAL_TABLE
            MOV     CX, 8192                     ; Number of words to move
            MOV     AH, F15_MOVE_BLOCK           ; Function 87H.
            INT     INT_SYSTEM                   ; Int 15H
            JC      HANDLE_ERRORS
            JNE     HANDLE_ERRORS
                    .                            ; Continue if everything OKAY
HANDLE_ERRORS:
                    .                            ; Do Error processing here
; Actual Table of pointers passed to the routines. They use the
; Global descriptor structure described above.
ACTUAL_TABLE:
RESERVED    SAMPLE_GDT    <0,0,0,0,0>
CALLERS     SAMPLE_GDT    <0,0,0,0,0>
SOURCE      SAMPLE_GDT    <16384,8000H,0BH,93H,0>
```

```
DEST      SAMPLE_GDT   <16384,0,0,93H,0>   ; The high byte
                                           : and low word
                                           ; will be loaded
                                           : in the code
BIOS      SAMPLE_GDT   <0,0,0,0,0>
STACK     SAMPLE_GDT   <0,0,0,0,0>

BUFFER    DB     16384 DUP (?)      ; Actual destination buffer
```

F15_GET_XMEM_SIZE (AH = 88H)

Determine how much RAM there is above the first one megabyte of memory.

On Entry: AH = F15_GET_XMEM_SIZE (88H)

On Exit: AX = Total number of 1KB blocks above one megabyte.

Registers Altered: AX.

F15_ENTER_PROT (AH = 89H)

Allows a routine to enter protected mode. When the BIOS function has executed, the processor will be in protected mode and the routine specified will be called. The calling program must create a set of descriptor tables as follows:

Dummy Descriptor Table:	Initialize to zero.
Global Descriptor Table:	Load program dependant values.
Interrupt Descriptor Table:	Load program dependant values.
Data segment Descriptor:	Load program dependant values.
Extra segment Descriptor:	Load program dependant values.
Stack segment Descriptor:	Load program dependant values.
Code segment Descriptor:	Load program dependant values.
BIOS Descriptor Table:	Initialize to zero.

When calling this function, the user should be aware that: 1) the BIOS functions are not available, 2) the interrupt tables must be moved to avoid conflict with the 80286 interrupt vectors, 3) the user loaded descriptor tables must not overlap with the BIOS's descriptor tables and 4) because of the system's second (HP) 8259 slave controller, both the master 8259 and the HP slave must be reprogrammed by the user on entry to protected mode.

Upon return from protected mode the system BIOS will return control to the return point specified at 40H:67H. The user should recover the stack and continue.

There are a few points of caution that should be observed:

1. Any code which is expected to run mixed mode, that is both protected mode and real mode, must not make any far references, including far calls.

2. Also, any return addresses put on the stack must have been generated in the same mode in which the return code executes, or else they must be near returns.

3. The system address line A20 must be forced to 0 when the system is operating in real mode. This task is performed by the 8041 controller. When the system enters protected mode, A20 must be released, and when it enters real mode it must be forced to 0 again. It is the program's responsibility to issue the appropriate command to the 8041 controller before changing modes (see Section 5).

On Entry: AH = F15__ENTER__PROT (89H)
BH = Offset into interrupt table where interrupts coming from the Master 8259 will go (Interrupt level 1).
BL = Offset into interrupt table where interrupts coming from the industry standard (STD) slave 8259 will go (Interrupt level 2).
ES:SI = Pointer to a set of descriptor tables. The following descriptors must be passed by the calling routine: Dummy Descriptor (DUMMY), Global Descriptor Table (GDT), Interrupt Descriptor Table (IDT), Data Segment Descriptor Table (DS), Extra Segment Descriptor Table (ES), Stack Segment Descriptor Table (SS), Code Segment Descriptor Table (CS) and BIOS Descriptor Table (BIOS).

On Exit: AH = 0, If successfully entered Protected Mode.

Registers Altered: All.

Example: To enter protected mode and start executing the routine PROTECTED.

```
;
; Load up descriptor tables with appropriate values. See the
; iAPX 286 Programmer's Reference Manual for details.
;
```

```
                .
                .
                .
;
; Load registers for calling INT 15H function.
;
          MOV     AH, F15__ENTER__PROT       ; Enter protected mode
                                             ; function 89H.

          ; Offset for 8259's must be greater than 32 because 80286
          ; uses the first 32 interrupts vectors.
          MOV     BH, 40                     ; New offset for master 8259.
          MOV     BL, 48                     ; New offset for STD-slave 8259.
          MOV     ES, seg GLOBAL__TABLE      ; Table of descriptors.
          MOV     SI, offset GLOBAL__TABLE
          INT     INT__SYSTEM                ; Int 15H

PROTECTED:
;
; Code starts executing here after call to INT 15H
; sets up CS__DT to point to PROTECTED label.
;
; The first thing to do in this case is reprogram the master
; 8259 and the HP-slave (interrupt controller's):
SLV__M__PORT0     EQU     20H
SLV__M__PORT1     EQU     21H
SLV__S1__PORT0    EQU     7CH
SLV__S1__PORT1    EQU     7DH
;
; Program the master 8259:
;
          MOV     AL,11H                     ; Edge triggered cascade mode
          OUT     SLV__M__PORT0,AL
          JMP     $+2
          MOV     AL,40                      ; Interrupt TYPE 40.
          OUT     SLV__M__PORT1,AL
          JMP     $+2
          MOV     AL,06H                     ; Slaves mask, at interrupt levels
          OUT     SLV__M__PORT1,AL           ; 1 and 2.
          JMP     $+2
          MOV     AL,01                      ; 8259 in "8086" mode.
          OUT     SLV__M__PORT1,AL
          JMP     $+2
          MOV     AL,0FFH                    ; Disable all interrupts.
```

```
                OUT     SLV__M__PORT1,AL
                JMP     $+2
;
; PROGRAM HP-SLAVE'S 8259:
;
                MOV     AL,11H                  ; Edge triggered cascade mode
                OUT     SLV__S1__PORT0,AL
                JMP     $+2
                MOV     AL,56                   ; Interrupt type 56.
                OUT     SLV__S1__PORT1,AL
                JMP     $+2
                MOV     AL,01                   ; Slave ID
                OUT     SLV__S1__PORT1,AL
                JMP     $+2
                MOV     AL,01                   ; ''8086'' Mode
                OUT     SLV__S1__PORT1,AL
                JMP     $+2
                MOV     AL,0FFH                 ; Disable all interrupts
                OUT     SLV__S1__PORT1,AL
                JMP     $+2
                MOV     AL,68H                  ; Enable special mask mode.
                OUT     SLV__S1__PORT0,AL
                JMP     $+2
                        .
                        .                       ; Continue with protected mode here.
                        .
;
; Descriptor tables needed for this function call. The entries
; marked by 'F' must be filled in by the user. Those marked with
; '0' are filled by INT 15H. For a definition of the SAMPLE__GDT
; structure see the F15__BLOCK__MOVE example. For information as
; to how to fill this table see the iAPX 80286 Programmer's
; Reference Manual.
;
GLOBAL__TABLE:
RESERVED      SAMPLE__GDT     <0,0,0,0,0>
GLBL__DT      SAMPLE__GDT     <F,F,F,F,F>
IDT__DT       SAMPLE__GDT     <F,F,F,F,F>
DS__DT        SAMPLE__GDT     <F,F,F,F,F>
ES__DT        SAMPLE__GDT     <F,F,F,F,F>
SS__DT        SAMPLE__GDT     <F,F,F,F,F>
CS__DT        SAMPLE__GDT     <F,F,F,F,F>
BIOS__DT      SAMPLE__GDT     <0,0,0,0,0>
```

F15__DEV__BUSY (AH = 90H)

Device busy function. This is a "hook" for multitasking systems. Currently the function just clears the Carry flag and returns.

On Entry: AH = F15__DEV__BUSY (90H)
AL = Device Type:
0 thru 7FH = Device can not be shared. The operating system handling this "hook" must serialize access to this device.
80H thru 0BFH = Device can be shared among multiple processes. The operating system handling this "hook" must use the ES:BX registers to distinguish between calls.
0C0H thru 0FFH = Devices of this type must wait for a fixed amount of time. This amount of time is device dependant. Control should be returned to the device after the fixed amount time.
List of Device Types:
00H = Disc, time out required
01H = Diskette, time out required
02H = Keyboard, no time out required
80H = Network, no time out required
0FDH = Start diskette motor, time out required
0FFH = Printer, time out required.

On Exit: No values returned.

Registers Altered: None.

F15__INT__COMPLETE (AH = 91H)

Signals interrupt completed. This is a "hook" for multitasking systems. Currently the function does an IRET.

On Entry: AH = F15__INT__COMPLETE (91H)
AL = Device Type, see list of previous function.

On Exit: No registers used.

Registers Altered: None.

9.6 Time And Date Driver (INT 1AH)

Table 9.2 describes functions provided by the BIOS to manage the CMOS clock and the software clock.

Table 9.2

Time and Date Driver Function Code Summary

INT Hex	Function Equate	Function Value	Definition
1AH	INT__CLOCK		Time and date
	F1A__RD__CLK__CNT	00H	Read current clock count
	F1A__SET__CLK__CNT	01H	Set current clock count
	F1A__GET__RTC	02H	Read real-time clock
	F1A__SET__RTC	03H	Set real-time clock
	F1A__GET__DATE	04H	Read date from real-time clock
	F1A__SET__DATE	05H	Set date in real-time clock
	F1A__SET__ALARM	06H	Set alarm
	F1A__RESET__ALARM	07H	Reset alarm

Time and Date Driver Function Definitions

F1A__RD__CLK__CNT (AH = 00H)

Reads the current setting of the software clock. There are 18.2 counts per second.

On Entry: AH = F1A__RD__CLK__CNT (00H)

On Exit: AL = Zero if the timer has not overflowed (not passed 24 hours since the last read).
Nonzero if time has overflowed.
CX = High word of the count. (There are 18.2 counts per second).
DX = Low word of count.

Registers Altered: AX, CX, DX

F1A__SET__CLK__CNT (AH = 01H)

Sets the count in the software clock. And resets the 24 hour overflow bit.

On Entry: AH = F1A__SET__CLK__CNT (01H)
CX = High word of Count.
DX = Low word of Count.

On Exit: No values returned.

Registers Altered: None

F1A__GET__RTC (AH = 02H)

Gets the time from the real-time clock.

On Entry: AH = F1A__GET__RTC (02H)

On Exit: CH = Hours in BCD.
CL = Minutes in BCD.
DH = Seconds in BCD.
Carry flag = 1 if real-time clock is not operating.

Registers Altered: AH, CX, DH

F1A__SET__RTC (AH = 03H)

Sets the time of the real-time clock.

On Entry: AH = F1A__SET__RTC (03H)
CH = Hours in BCD.
CL = Minutes in BCD.
DH = Seconds in BCD.
DL = 1 if daylight savings time option.
0 otherwise.

On Exit: No values returned.

Registers Altered: AH.

F1A__GET__DATE (AH = 04H)

Gets the date from the real-time clock.

On Entry: AH = F1A__GET__DATE (04H)

On Exit: CH = 19 if 20th century or
20 if 21st century.
CL = Year in BCD.
DH = Month in BCD.
DL = Day in BCD.
Carry flag set if the real-time clock not operating.

Register Altered: AH, CX, DX.

F1A__SET__DATE (AH = 05H)

Sets the date of the real-time clock.

On Entry: AH = F1A__SET__DATE (05H)
CH = 19 if 20th century or
20 if 21st century.
CL = Year in BCD.
DH = Month in BCD.
DL = Day in BCD.

On Exit: No values returned.

Registers Altered: AH.

F1A__SET__ALARM (AH = 06H)

Sets the alarm to generate an INT 4AH when the specified amount of time has elapsed. The user must place an appropriate interrupt handling routine in the INT 4AH vector.

On Entry: AH = F1A__SET__ALARM (06H)
CH = Hours in BCD.
CL = Minutes in BCD.
DH = Seconds in BCD.

On Exit: Carry flag = 1 if the real-time clock is not operating or the alarm is already set.

Registers Altered: AH.

F1A__RESET__ALARM (AH = 07H)

Clears the current alarm if any was set.

On Entry: AH = F1A__RESET__ALARM (07H)

On Exit: No values returned.

Registers Altered: AH.

9.7 V__SCOPY Driver (BP = 0000H)

This driver does an IRET for all function calls.

9.8 V__DOLITTLE Driver (BP = 0006H)

This driver does an IRET for all function calls.

9.9 V__PNULL Driver (BP = 000CH)

This driver loads AH with RS__SUCCESSFUL and does an IRET for all function calls.

9.10 V__SYSTEM Driver (BP = 0012H)

Table 9.3 summarizes the V__SYSTEM Functions. A more detailed description follows the table.

Table 9.3

V__SYSTEM Driver Function Code Summary

Vector Address	Func. Value	Function Equate	Definition
0012H		V__SYSTEM	System Management Functions
0012H	00	F__ISR	Interrupt service routine (unsupported)
0012H	02	F__SYSTEM	Standard Driver Functions
0012H	02/00	SF__INIT	System initialization
0012H	04	F__INS__BASEHPVT	Returns HP__VECTOR__TABLE segment
0012H	06	F__INS__XCHGFIX	Exchanges fixed table entries
0012H	08	F__INS__XCHGRSVD	Sets next ''reserved'' entry in table
0012H	0A	F__INS__XCHGFREE	Sets next ''free'' entry in table
0012H	0C	F__INS__FIXOWNDS	Install fixed vector, user supplies DS
0012H	0E	F__INS__FIXGETDS	Install fixed vector, system supplies DS
0012H	10	F__INS__FIXGLBDS	Install fixed vector, DS set to global data area
0012H	12	F__INS__FREEOWNDS	Install next free vector, user supplies DS
0012H	14	F__INS__FREEGETDS	Install next free vector, system supplies DS
0012H	16	F__INS__FREEGLBDS	Install next free vector, DS set to global data area
0012H	18	F__INS__FIND	Search for matching device header
0012H	1E	F__RAM__GET	Get EX-BIOS memory pool address and size
0012H	20	F__RAM__RET	Set memory pool address and size
0012H	22	F__CMOS__GET	Read and verify CMOS memory
0012H	24	F__CMOS__RET	Write to CMOS memory
0012H	2A	F__YIELD	Just returns
0012H	2C		Reserved
0012H	2E		Reserved
0012H	30	F__SND__CLICK__ENABLE	Enable keyclick
0012H	32	F__SND__CLICK__DISABLE	Disable keyclick (Default)
0012H	34	F__SND__CLICK	Execute keyclick if enabled

Vector Address	Func. Value	Function Equate	Definition
0012H	36	F__SND__BEEP__ENABLE	Enables beep
0012H	38	F__SND__BEEP__DISABLE	Disables beep
0012H	3A	F__SND__BEEP	Beeps if enabled
0012H	3C	F__SND__SET__BEEP	Sets beep frequency
0012H	3E	F__SND__TONE	Produce tone, user supplied duration and frequency
0012H	40	F__STR__GET__FREE__INDEX	Return next free string index
0012H	42	F__STR__DEL__BUCKET	Delete bucket string list
0012H	44	F__STR__PUT__BUCKET	Add bucket to current string list
0012H	46	F__STR__GET__STRING	Search the list for index, return string
0012H	48	F__STR__GET__INDEX	Search list for a string, return index

Registers Altered: AH, DS, BP, ES

Example: Get the Base address of the HP__VECTOR__TABLE.

```
MOV     BP, V__SYSTEM                  ; HP vector (12H).
MOV     AH, F__INS__BASEHPVT           ; function 04H
PUSH    DS                             ; EX-BIOS destroys DS
INT     HP__ENTRY                      ; Int 6FH for EX-BIOS
MOV     AX, DS
POP     DS                             ; Restore DS
MOV     GLOBAL__DATA__AREA, AX
MOV     AX, ES
MOV     VECTOR__TABLE__SEGMENT, AX
```

The value returned in ES is the segment address of the HP__VECTOR__TABLE and the value returned in the DS register is the segment address of the EX-BIOS global data area.

V__SYSTEM Driver Function Definitions

F__ISR (AH = 00H)

Logical interrupt service routine. Currently, it loads AH with RS__UNSUPPORTED and does an IRET.

On Entry: BP = V__SYSTEM (12H)
AH = F__ISR (00H)

On Exit: AH = RS__UNSUPPORTED (02H)

Registers Altered: AH, BP, DS

SF__INIT (AX = 0200H)

System functions routines. The only function supported is SF__INIT (00H). The rest of the routines return with a status of RS__UNSUPPORTED in AH.

The SF__INIT routine sets up DS and initializes all the variables in the EX-BIOS global data area.

On Entry: BP = V__SYSTEM (12H)
AH = F__SYSTEM (02H)
AL = SF__INIT (00H)

On Exit: AH = Return Status Code
BX = DS of EX-BIOS global data area

Registers Altered: AH, BX, DS, BP

F__INS__BASEHPVT (04H)

Reports the segment where the HP__VECTOR__TABLE is located. This function can only be called after the V__SYSTEM driver has been initialized.

On Entry: BP = V__SYSTEM (12H)
AH = F__INS__BASEHPVT (04H)

On Exit: AH = Return Status Code
ES = Segment address of HP__VECTOR__TABLE.
DS = Segment of EX-BIOS global data area

F__INS__XCHGFIX (AH = 06H)

Exchanges the values in the registers for a particular entry in the HP__VECTOR__TABLE. This function can be used to replace an existing vector at a fixed location without initialization.

On Entry: BP = V__SYSTEM (12H)
AH = F__INS__XCHGFIX (06H)
BX = Vector address
DX = DS to be exchanged
ES:DI = CS:IP to be exchanged

```
On Exit:    AH = Return Status Code
                 0 = RS_SUCCESSFUL
            DX = DS from table
         ES:DI = CS:IP from table
```

Registers Altered: AH, BP, DS, ES, DI, DX

Example: Replace the EX-BIOS V_SVIDEO vector (54H).

```
MOV     BP, V_SYSTEM                   ; HP vector 12H.
MOV     AH, F_INS_XCHGFIX              ; Function 06H
MOV     BX, V_SVIDEO                   ; HP vector 54H
MOV     DI, CS                         ; Get CS, IP and DS of new
MOV     ES, DI                         ; video routines.
MOV     DI, offset NEW_VIDEO_ROUTINE
MOV     DX, DS
PUSH    DS                             ; EX-BIOS Destroys DS
INT     HP_ENTRY                       ; Int 6FH for EX-BIOS
POP     DS
MOV     OLD_CS, ES                     ; Save old CS, IP and DS
MOV     OLD_IP, DI                     ; just in case we need to
MOV     OLD_DS, DX                     ; put them back
```

F_INS_XCHGRSVD (AH = 08H)

Exchanges the values in the registers for the next reserved entry in the HP_VECTOR_TABLE. If a reserved vector is not available the function returns the RS_NO_VECTOR error code.

```
On Entry:   BP = V_SYSTEM (12H)
            AH = F_INS_XCHGRSVD (08H)
            DX = DS to be exchanged
         ES:DI = CS:IP to be exchanged

On Exit:    AH = Return Status Code
                 0    = RS_SUCCESSFUL
                 0F6H = RS_NO_VECTOR
            BX = Vector address
            DX = DS from table
         ES:DI = CS:IP to be exchanged
```

Registers Altered: AH, BP, DS, BX, ES, DI, DX

F__INS__XCHGFREE (AH = 0AH)

Exchanges the values in the registers for the next free entry in the HP__VECTOR__TABLE. If a free vector is not available, the function returns the RS__NO__VECTOR error code.

On Entry: BP = V__SYSTEM (12H)
AH = F__INS__XCHGFREE (0AH)
DX = DS to be exchanged
ES:DI = CS:IP to be exchanged

On Exit: AH = Return Status Code
0 = RS__SUCCESSFUL
0F6H = RS__NO__VECTOR
BX = Vector address
DX = DS from table
ES:DI = CS:IP to be exchanged

Registers Altered: AH, BP, DS, BX, ES, DI, DX

F__INS__FIXOWNDS (AH = 0CH)

Installs a given vector entry in the HP__VECTOR__TABLE and calls it with an SF__INIT function. Upon returning from initialization, the routine returns its data segment in the BX register.

Warning

> If the SF__INIT function returns with an error code of RS__FAIL (0FEH) the power-on self test sequence will be executed.

On Entry: BP = V__SYSTEM (12H)
AH = F__INS__FIXOWNDS (0CH)
BX = Vector address to be installed
ES:DI = CS:IP of the device

On Exit: AH = Return Status Code
0 = RS__SUCCESSFUL

Registers Altered: AH, BP, DS

F_INS_FIXGETDS (AH = 0EH)

Installs a given vector entry in the HP_VECTOR_TABLE and calls it with an SF_INIT function. This function should be used if the driver needs EX-BIOS RAM for its data segment. F_INS_FIXGETDS calls the routine to initialize with the "last used DS" in the BX register. The routine's initialization code decrements the "last used DS" value and returns to F_INS_FIXGETDS with this new value.

Warning

> If the SF_INIT function returns with an error code of RS_FAIL (0FEH) the power-on self test sequence will be executed.

On Entry: BP = V_SYSTEM (12H)
AH = F_INS_FIXGETDS (0EH)
BX = Vector address to be installed
ES:DI = CS:IP of the routine

On Exit: AH = Return Status Code
0 = RS_SUCCESSFUL

Registers Altered: AH, BP, DS

F_INS_FIXGLBDS (AH = 10H)

Installs a given vector entry in the HP_VECTOR_TABLE and calls it with an SF_INIT function. When F_INS_FIXGLBDS calls the initialization routine it passes the data segment of the EX-BIOS global data area in the BX register.

Warning

> If the SF_INIT function returns with an error code of RS_FAIL (0FEH) the power-on self test sequence will be executed.

On Entry: BP = V_SYSTEM (12H)
AH = F_INS_FIXGLBDS (10H)
BX = Vector address to be installed
ES:DI = CS:IP of the routine

On Exit: AH = Return Status Code
0 = RS_SUCCESSFUL

Registers Altered: AH, BP, DS

F__INS__FREEOWNDS (AH = 12H)

Installs a vector in the next free entry of the HP__VECTOR__TABLE and calls it with an SF__INIT function. Upon returning from initialization, the routine returns its DS in the BX register.

Warning

> If the SF__INIT function returns with an error code of RS__FAIL (0FEH) the power-on self test sequence will be executed.

```
On Entry:  BP  = V__SYSTEM (12H)
           AH  = F__INS__FREEOWNDS (12H)
           BX  = Vector address to be installed
        ES:DI  = CS:IP of the device

On Exit:   AH  = Return Status Code
                 0 = RS__SUCCESSFUL
```

Registers Altered: AH, BP, DS

F__INS__FREEGETDS (AH = 14H)

Installs a vector in the next free entry of the HP__VECTOR__TABLE and calls it with an SF__INIT function. This function is used if the driver needs EX-BIOS RAM for its data segment. F__INS__FREEGETDS calls the routine to initialize with the "last used DS" in the BX register. The routine's initialization code decrements the "last used DS" value and returns it to F__INS__FREEGETDS.

Warning

> If the SF__INIT function returns with an error code of RS__FAIL (0FEH) the power-on self test sequence will be executed.

```
On Entry:  BP  = V__SYSTEM (12H)
           AH  = F__INS__FREEGETDS (14H)
        ES:DI  = CS:IP of the routine

On Exit:   AH  = Return Status Code
                 0 = RS__SUCCESSFUL
```

Registers Altered: AH, BP, DS

Example: Install the ACME__INT vector in the next free vector and allocate two paragraphs of data when its initialization routine gets called.

```
            MOV      BP, V_SYSTEM                ; HP vector 12H for EX-BIOS.
            MOV      AH, F_INS_FREEGETDS         ; Function 14H
            MOV      DI, CS                      ; Get CS, IP of ACME_INT routines
            MOV      ES, DI
            MOV      DI, offset ACME_INT
            PUSH     DS                          ; EX-BIOS Destroys DS
            INT      HP_ENTRY                    ; Int 6FH for EX-BIOS
            POP      DS
            MOV      VECTOR_NUMBER, BX           ; Save the vector number
                                                 ; routines are installed.

; ACME_INT routine handles initialization and allocates 2
; paragraphs from EX-BIOS RAM for its data segment.

ACME_INT:
            CMP      AH, F_SYSTEM                ; Decode F_SYSTEM subfunction
            JNE      NOT_SUPPORTED               ; SF_INIT.
            CMP      AL, SF_INIT
            JE       ACME_INIT
NOT_SUPPORTED:                                   ; Any unknown functions should
            MOV      AH, RS_UNSUPPORTED          ; return with RS_UNSUPPORTED
            IRET                                 ; in AH.

ACME_INIT:
            SUB      BX, 2                       ; Decrement the "last used DS"
                                                 ; passed to us. This allocates 2
                                                 ; paragraphs and makes our data
                                                 ; segment the "last used DS". Make
                                                 ; sure to pass this new BX back to
                                                 ; F_INS_FREEGETDS code.
            MOV      DS, BX                      ; Now we can initialize the data in
                                                 ; our segment.
            ASSUME   DS:NOTHING
            MOV      ACME_ATTR, 55AAH            ; Put data into Attribute word
            MOV      ACME_NAME_INDEX, 55AAH      ; Put a dummy index for now.
                     .
                     .                           ; Initialize rest of data segment here.
                     .
            MOV      AH, RS_SUCCESSFUL           ; Always return this status if
                                                 ; successful initialization.
            IRET
```

```
;
; Sample segment for this routine
;
ACME_SEG             struc
ACME_ATTR            dw     0             ; Attribute word of ACME's data
                                          ; segment.
ACME_NAME_INDEX      dw     0             ; Index name of ACME routine.
ACME_REST            db     28 dup (?)    ; rest of data segment
ACME_SEG             ends
```

F_INS_FREEGLBDS (AH = 16H)

Installs a vector in the next free entry of the HP_VECTOR_TABLE and calls it with an SF_INIT function. When F_INS_FREEGLBDS calls the initialization routine it passes the data segment of the EX-BIOS global data area in the BX register.

Warning

> If the SF_INIT function returns with an error code of RS_FAIL (0FEH) the power-on self test sequence will be executed.

On Entry:	BP	=	V_SYSTEM (12H)
	AH	=	F_INS_FREEGLBDS (16H)
	ES:DI	=	CS:IP of the routine

On Exit:	AH	=	Return Status Code 0 = RS_SUCCESSFUL

Registers Altered: AH, BP, DS

F_INS_FIND (AH = 18H)

This function is used to search the HP_VECTOR_TABLE for drivers that have equal or similar values in a specified field of their data segment. Parameters passed to the function specify the location of the 16-bit field, the bits within the field to be compared (and_mask) and the pattern of bits the field is to be compared with. Given a starting vector address, the function searches the vector table for the next driver that matches the conditions specified and returns its vector address in SI.

```
On Entry:  BP  = V_SYSTEM (12H)
           AH  = F_INS_FIND (18H)
           AL  = 0 then respond on equality to pattern
                   ((field) .AND. (and_mask)) = pattern
                 2 then respond on non_equal
                   ((field) .AND. (and_mask)) < > pattern
           BX  = and_mask
           DX  = pattern
           SI  = vector address to start the search from.
           DI  = field to be used in the function, this is the offset into an HP header.

On Exit:   AH  = Return status
                  0   = RS_SUCCESSFUL
               0FEH   = RS_FAIL—No match found
           SI  = Vector address of the first entry that matched.
```

Registers Altered: AH, BP, DS, SI

Example: Find a vector that has the value X5AXH ("X" means allow these digits to take any value) in its attribute header (the first word of the driver's data segment)

```
        MOV     BP, V_SYSTEM            ; HP vector 12H
        MOV     AH, F_INS_FIND          ; Function 18H
        MOV     AL, 0                   ; Return RS_SUCCESSFUL when the
                                        ; value is equal
        MOV     DI, 0                   ; Look in the first word of driver's
                                        ; data segment
        MOV     DX, 05A0H               ; Look for value '5A' in the middle
                                        ; of the word.
        MOV     BX, 0FF0H               ; Mask off the don't care parts.
        MOV     SI, 0                   ; Start looking from the first vector
                                        ; position.
        PUSH    DS                      ; EX-BIOS destroys DS
        INT     HP_ENTRY                ; Int 6FH for EX-BIOS
        POP     DS
        CMP     AH, RS_SUCCESSFUL       ; See if it found a match ?
        JNE     VECTOR_NOT_FOUND
VECTOR_FOUND:                           ; Yes
        MOV     SAVED_VECTOR, SI
        .
        .
        .
VECTOR_NOT_FOUND:                       ; No
        .
        .
        .
```

F_RAM_GET (AH = 1EH)

This function gets the segment pointers of the EX-BIOS free RAM area. Two pointers are returned by this function call, the "last used DS" pointer marks the first paragraph of EX-BIOS RAM that is free for use. The "max DS" pointer marks the lowest value that "last used DS" can have. Figure 9.1 shows how the EX-BIOS memory is organized.

See the F_RAM_RET memory function.

```
On Entry:  BP  = V_SYSTEM (12H)
           AH  = F_RAM_GET (1EH)

On Exit:   AH  = RS_SUCCESSFUL
           BX  = "last used DS"
           DX  = "max DS"
```

Registers Altered: AH, BP, DS, BX, DX

F_RAM_RET (AH = 20H)

Sets the "last used DS" and "max DS" EX-BIOS pointers to the values passed in the BX and DX registers. This allows the calling routine to reserve a piece of the EX-BIOS memory.

Caution

> The F_INS_FIXGETDS and F_INS_FREEGETDS functions described above also modify these values. Use caution when allocating memory with both methods.

```
On Entry:  BP  = V_SYSTEM (12H)
           AH  = F_RAM_GET (20H)
           BX  = "last used DS"
           DX  = "max DS"

On Exit:   AH  = RS_SUCCESSFUL
```

Registers Altered: AH, BP, DS

Example: The following code allocates five paragraphs (80 bytes) of EX-BIOS memory.

```
;
;        Get the memory pointers first.
;
         MOV   BP, V_SYSTEM          ; HP vector 12H.
         MOV   AH, F_RAM_GET         ; function 1EH
```

```
        PUSH    DS                      ; EX-BIOS Destroys DS
        INT     HP__ENTRY               ; Int 6FH for EX-BIOS
        POP     DS
;
; Check to see if there is enough memory to allocate 5 paragraphs.
;
        SUB     BX, 0005H               ; Create a new "last used DS" by
                                        ; moving pointer towards "max DS".
        CMP     BX, DX                  ; Is "last used DS" > = "max DS"?
        JL      NO__MEMORY__LEFT
ENOUGH__MEMORY__LEFT:                   ; Yes: Allocate 5 paragraphs.
        MOV     BP, V__SYSTEM           ; HP vector 12H
        MOV     AH, F__RAM__RET         ; function 20H
        PUSH    DS                      ; EX-BIOS Destroys DS
        INT     HP__ENTRY               ; Int 6FH for EX-BIOS
        POP     DS
        MOV     MEMORY__SEG, BX         ; Save this new memory pointer for
                                        ; later use
        .
        .
        .                               ; Continue
        .
NO__MEMORY__LEFT:                       ; No:
;
; Typical thing to do here is to allocate more memory for the
; the EX-BIOS RAM and reboot system.
;
```

F__CMOS__GET (AH = 22H)

Read a byte from CMOS. It verifies the checksum on the industry standard CMOS area and returns RS__FAIL if the checksum is invalid.

On Entry: BP = V__SYSTEM (12H)
AH = F__CMOS__GET (22H)
BL = address of CMOS byte to read

On Exit: AH = Return Status Code
AL = byte of data from CMOS

Registers Altered: AX, BP, DS.

F_CMOS_RET (AH = 24H)

Write a byte to CMOS. Calculate a new checksum for both the industry standard CMOS area and the HP CMOS area.

```
On Entry:  BP = V_SYSTEM (12H)
           AH = F_CMOS_RET (24H)
           AL = byte of data to be written to CMOS
           BL = address of byte to be written to CMOS

On Exit:   AH = Return Status Code
```

Registers Altered: AX, BP, DS.

Example: Make the monochrome display the primary video adapter by setting this information in the equipment byte of CMOS memory.

```
;
; Read the equipment byte.
;
        MOV     BP, V_SYSTEM        ; HP vector 12H.
        MOV     AH, F_CMOS_GET      ; function 22H
        MOV     BL, 14H             ; Address of the equipment byte
        PUSH    DS                  ; EX-BIOS destroys DS
        INT     HP_ENTRY            ; Int 6FH for EX-BIOS
        POP     DS
        CMP     AH, RS_FAIL         ; See if CMOS is valid
        JE      INVALID_CMOS
;
; Isolate the video and set appropiate video bits.
;
        AND     AL, 11001111B
        OR      AL, 00110000B       ; Select monochrome display
;
; Write the equipment byte.
;
        MOV     BP, V_SYSTEM        ; HP vector 12H
        MOV     AH, F_CMOS_RET      ; function 24H
        PUSH    DS                  ; EX-BIOS destroys DS
        INT     HP_ENTRY            ; Int 6FH for EX-BIOS
        POP     DS
        .
        .
        .
```

INVALID__CMOS:

.
.
.

F__YIELD (AH = 2AH)

Currently loads AH with RS__SUCCESSFUL and does an IRET. This is a "hook" for multitasking systems.

On Entry: BP = V__SYSTEM (12H)
AH = F__YIELD (2AH)

On Exit: AH = Return Status Code

Registers Altered: AH, BP, DS

F__SND__CLICK__ENABLE (AH = 30H)

Enables the keyclick function and flushes any pending keyclicks.

On Entry: BP = V__SYSTEM (12H)
AH = F__SND__CLICK__ENABLE (30H)

On Exit: AH = Return Status Code

Registers Altered: AH, BP, DS.

F__SND__CLICK__DISABLE (AH = 32H)

Disables the keyclick function, sets the EX-BIOS global data area T__SND__CLICK__DURA byte to zero, and flushes any pending keyclicks.

On Entry: BP = V__SYSTEM (12H)
AH = F__SND__CLICK__DISABLE (32H)

On Exit: AH = Return Status Code

Registers Altered: AH, BP, DS

F__SND__CLICK (AH = 34H)

This functions under the following conditions:

- If greater than or equal to four clicks are pending then exit.
- If less than four clicks are pending then increment the count and exit.
- If no keyclicks are pending then execute the keyclick.

On Entry: BP = V__SYSTEM (12H)
AH = F__SND__CLICK (34H)

On Exit: AH = Return Status Code

Registers Altered: AH, BP, DS

F__SND__BEEP__ENABLE (AH = 36H)

Enables the beep function.

On Entry: BP = V__SYSTEM (12H)
AH = F__SND__BEEP__ENABLE (36H)

On Exit: AH = Return Status Code

Registers Altered: AH, BP, DS

F__SND__BEEP__DISABLE (AH = 38H)

Disables the beep function.

On Entry: BP = V__SYSTEM (12H)
AH = F__SND__BEEP__DISABLE (38H)

On Exit: AH = Return Status Code

Registers Altered: AH, BP, DS

F__SND__BEEP (AH = 3AH)

Makes a sound as defined by the current values of T__SND__BEEP__CYCLE and T__SND__BEEP__DURA in the EX-BIOS data area.

```
On Entry:  BP  = V__SYSTEM (12H)
           AH  = F__SND__BEEP (3AH)

On Exit:   AH  = Return Status Code
```

Registers Altered: AH, BP, DS

F__SND__SET__BEEP (AH = 3CH)

Defines beep frequency and duration.

```
On Entry:  BP  = V__SYSTEM (12H)
           AH  = F__SND__SET__BEEP (3CH)
           BX  = Frequency 1 to 25000 hz.
                 If (BX) = 0 then tone off.
           DX  = duration of tone in 10 microsecond increments

On Exit:   AH  = Return Status Code
```

Registers Altered: AH, DS, BP.

Example: Set Beep frequency to 660 Hz for duration of 1/2 second.

```
MOV    BP, V__SYSTEM            ; HP vector 12H
MOV    AH,F__SND__SET__BEEP     ; function 3CH
MOV    BX, 660                  ; Frequency in hertz
MOV    DX,50000                 ; 1/2 a second in 10
                                ; microseconds increments.
PUSH   DS                        ; EX-BIOS destroys DS
INT    HP__ENTRY                 ; Int 6FH for EX-BIOS
POP    DS
```

F__SND__TONE (AH = 3EH)

Generates a tone of the given frequency and duration with an approximate 0.5 percent error.

On Entry: BP = V__SYSTEM (12H)
AH = F__SND__TONE (3EH)
BX = Frequency 1 to 25000hz
If (BX) = 0 then tone off.
DX = Duration of tone in 10 microsecond increments.

On Exit: AH = Return Status Code

Registers Altered: AH, DS, BP

F__STR__GET__FREE__INDEX (AH = 40H)

Returns to caller the next string index that does not conflict with the ROM based string indices.

On Entry: BP = V__SYSTEM (12H)
AH = F__STR__GET__FREE__INDEX (40H)

On Exit: AH = RS__SUCCESSFUL
BX = Next free index.

Registers Altered: AH, BX, DS, BP

Example: This example gets the next string index available to the user.

```
MOV     BP, V__SYSTEM                  ; HP vector 12H
MOV     AH,F__STR__GET__FREE__INDEX    ; function 40H
PUSH    DS                             ; EX-BIOS destroys DS
INT     HP__ENTRY                      ; Int 6FH for EX-BIOS
POP     DS
MOV     FIRST__FREE__INDEX,BX          ; Save it for later use.
.
.
.
```

F__STR__DEL__BUCKET (AH = 42H)

Finds a header with the given address and deletes it from the bucket header list.

On Entry: BP = V__SYSTEM (12H)
AH = F__STR__DEL__BUCKET (42H)
DI = offset address of bucket header
ES = segment address of bucket header

On Exit: AH = RS_SUCCESSFUL if header found and deleted
RS_FAIL if header not found.

Registers Altered: AH, DS, BP.

F_STR_PUT_BUCKET (AH = 44H)

Takes a header and its corresponding pointers and adds them to the front of the list.

On Entry: BP = V_SYSTEM (12H)
AH = F_STR_PUT_BUCKET (44H)
DI = Offset address of header
ES = Segment address of header

On Exit: AH = RS_SUCCESSFUL

Registers Altered: AH, BP, DS.

Example: Adds a set of strings and its associated data structures for the ACME_INT driver.

```
;
; String data structures (see figure 9.2)
;
STR_HEADER          STRUC
STR_NXT_HDR         DD        (?)
STR_UPPER_BOUND     DW        (?)
STR_LOWER_BOUND     DW        (?)
STR_LIST_PTR        DD        (?)
STR_SEGMENT         DW        (?)
STR_HEADER          ENDS

;
; Now build a bucket (set of strings) for the ACME_INT:
;
; First list ACME_INT's strings:
size_acme_name      db        l_acme_name - f_acme_name - 1
f_acme_name         =         $
acme_name           db        'Acme Co.',0H
l_acme_name         =         $

size_item_1         db        l_item_1 - f_item_1 - 1
f_item_1            =         $
item_1              db        'Hello World',0H
l_item_1            =         $
```

```
size_item_2          db        l_item_2 - f_item_2 - 1
f_item_2              =        $
item_2               db        'Widgets',0H
l_item_2              =        $
;
; Now build table of bucket pointers:
;
acme_ptrs            label     near
                     dw        offset acme_name
                     dw        offset    item_1
                     dw        offset    item_2
;
; Now build the bucket header data structure
;
acme_bucket          label     near
                     dw        0FFFFH   ; This is the only bucket.
                     dw        0FFFFH
                     dw        1002H    ; Adding string indexes 1000..1002
                     dw        1000H
                     dw        offset acme_ptrs   ; address of pointer list
                     dw        segment acme_ptrs
                     dw        segment acme_name   ; segment of all strings
;
; Do the function call to add bucket.
;
                    MOV     BP, V_SYSTEM              ; HP vector 12H
                    MOV     AH, F_STR_PUT_BUCKET      ; function 44H
                    MOV     DI,  offset acme_bucket
                    MOV     ES,  segment acme_bucket
                    PUSH    DS                        ; EX-BIOS Destroys DS
                    INT     HP_ENTRY                  ; Int 6FH for EX-BIOS
                    POP     DS
```

F_STR_GET_STRING (AH = 46H)

Given an index, this function searches the list of bucket headers for the bucket pointer with the given index. It returns a pointer to the string.

```
On Entry:  BP = V_SYSTEM (12H)
           AH = F_STR_GET_STRING (46H)
           BX = String index
```

On Exit: AH = RS__SUCCESSFUL if index found in a bucket
CX = How many characters are in the string exclusive of the byte count and the zero byte at the end.
DS:SI = Address of header where string was found.
ES:DI = Pointer to first character of the string.

Registers Altered: AH, CX, SI, DI, BP, DS, ES

Example: Search for the name of the ACME__INT routine as index 1000H.

```
MOV     BP,      V__SYSTEM              ; HP vector 12H
        MOV      AH, F__STR__GET__STRING ; Function 46H
        MOV      BX, 1000H              ; Index of ACME__INT name string
        PUSH     DS                     ; EX-BIOS destroys DS
        INT      HP__ENTRY              ; Int 6FH for EX-BIOS
;
; Write the string to the screen:
;
        MOV      AX, F10__WRS__00       ; Call the write string function.
        MOV      BP, SI                 ; Offset of string address
        PUSH     DS                     ; Segment of string address
        POP      ES                     ; CX is already set
        MOV      DX, 0                  ; Cursor position at (0,0)
        MOV      BH, 0                  ; Video page 0
        MOV      BL, 7                  ; Character attribute
        INT      INT__VIDEO             ; Video interrupt 10
        POP      DS                     ; Recover old DS
```

F__STR__GET__INDEX (AH = 48H)

Given a pointer to a string it returns the index of the string if it is in the bucket header list.

On Entry: BP = V__SYSTEM (12H)
AH = F__STR__GET__INDEX (48H)
ES:DI = Pointer to first character of the zero terminated string.

On Exit: AH = RS__SUCCESSFUL if index was found.
BX = Index found for the given string.

Registers Altered: AH, BX, BP, DS

Example: Get the index of the ACME_NAME string.

```
MOV     BP, V_SYSTEM              ; HP vector 12H
MOV     AH, F_STR_GET_INDEX       ; function 48H
MOV     DI, seg ACME_NAME         ; Move segment of string
MOV     ES, DI                    ; into ES
MOV     DI, offset ACME_NAME
PUSH    DS                        ; EX-BIOS destroys DS
INT     HP_ENTRY                  ; Int 6FH for EX-BIOS
POP     DS
MOV     ACME_NAME_INDEX, BX       ; Save the index.
```

SECTION 10

TABLE OF CONTENTS

SECTION 10. SYSTEM PROCESSES

10.1 Overview

This section describes system processes contained in the ROM BIOS. System processes are different from drivers in that they are not readily accessible to application programs and they perform larger tasks than a typical driver function. The ROM BIOS has five main system processes; reset, power-on self test (POST), system generation (SYSGEN), booting (BOOT), and return from protected mode.

10.2 Reset

The 80286 is reset through a hardware reset signal. This signal sets the CS and IP registers to begin execution at memory location 0F000:0FFF0H. The system can be reset by either a hardware reset to the 80286, or by any software routine that jumps to memory location 0F000:0FFF0H. There are four events that initiate a system reset:

- Power-on. This reset occurs when power is applied to the system. The power supply resets the 80286 through its reset signal when the system is turned on. POST is initiated and performs a full memory test.

- Hard Reset. This reset is initiated by the <CTRL>-<Alt>-<Sys req> key sequence. This sequence generates a scancode that is interpreted by the HP-HIL controller as a system reset. The HP-HIL controller asserts the Non-Maskable Interrupt (NMI) line when this scancode sequence is detected. The default interrupt service routine for the NMI interrupt (02H) in turn jumps to the reset memory location. This reset is a superset of the industry standard. POST is initiated and performs a full memory test.

- Soft Reset. This reset is initiated by the <CTRL>-<Alt>-<DEL> key sequence. This sequence is interpreted by the INT 09H keyboard interrupt service routine as a reset command. POST is initiated. A full memory test is not performed.

- Programmatic Reset. The final reset source is a software initiated hardware reset. A command is sent to the 8041 controller to pulse the 80286 hardware reset line. Once the 80286 has been placed in the Protected Mode, a hardware reset is the only method available to return to the Real Mode. POST may or may not be performed depending upon the shut down status byte in CMOS.

Once a reset operation has been initiated by one of the four possible sources, the system must determine if it is a power-on reset. If it is a power-on reset, bit 2 in the 8041 controller's status port is cleared. POST is performed. A command is sent to the 8041 to set bit 2. If it is not a power-on reset, bit 2 in the 8041 controller status port is already set. The CMOS shutdown status byte determines whether POST is performed.

If it is not a power-on reset, the system looks at the shut down status byte (CMOS address 0FH) to determine whether to perform POST or return from protected mode. If the shut down status byte is set to one of the values that indicates the system is returning from protected mode, the reset process will initiate the return from protected mode process. This process is described next. All other values of the shut down status byte are interpreted as reset commands, and the reset process will initiate the power-on self test process. The reset process has completed its tasks when one of these two processes has been invoked.

10.3 Protected Mode Support

The 80286 processor has two modes of operation. Protected mode provides memory protection, virtual memory addressing, and a 16 MB physical address space. Real mode provides a 1 MB address space and an 8086 compatible mode. The normal mode of operation of the system is real mode. However, a few programs use protected mode, for example, VDISC.SYS, the DOS virtual disc device driver.

The system provides some support to the programmer for use of the protected mode features. The INT 15H driver provides two functions that support system operation in protected mode. One of these functions enables data to be moved to and from extended memory. This function enters protected mode to perform this task, and returns to real mode. The second function provides a method for programmers to switch into protected mode. These functions are described in Section 9 of this manual.

10.3.1 Shut Down Status Byte

The shut down status byte is used by the system to determine what action should be taken on reset. Table 10.1 shows how the shut down status byte is interpreted. Note that any value that does not indicate a return from protected mode is interpreted by the system as a reset, and will cause the reset process to invoke POST.

Table 10.1

Shut Down Status Byte

Value	Definition
00-04H	Perform power-on reset sequence.
05H	Flush keyboard and jump via double word stored at 0040:0067H.
06-07H	Perform power-on reset sequence.
08H	Return from test of extended memory.
09H	Return from INT 15H block move function.
0AH	Jump via double word stored at 0040:0067H.
0BH-FFH	Perform power-on reset sequence.

The values 08H and 09H are used internally by the ROM BIOS. If the return from protected mode process detects either of these values, it will branch to their respective routines. Values 05H and 0AH should be used by all other programs returning from protected mode.

10.4 Power-On Self Test (POST)

Each time the system is powered-on, or a reset is performed, the POST process is executed. The purpose of the POST process is to verify the basic functionality of the system components and to initialize certain system parameters. The POST process performs the following tasks:

- Initialize the video display for diagnostic messages.
- Test the operation of the 80286.
- Test the system ROM.
- Test and initialize 8254 timer/counter and start the refresh counter.
- Test and initialize DMA controllers and DMA page registers.
- Test the first 64KB of system RAM.
- Test and initialize the 8259A interrupt controllers.
- Test the 8041 controller.
- Test the HP-HIL controller and link.
- Test CMOS RAM for integrity.
- Determine if manufacturing electronic tool is present, if so, run manufacturing test.
- Test the remaining base system RAM (RAM above the first 64KB.
- Test the extended RAM above memory address 100000H. (protected mode RAM.)
- Test the real-time clock portion of the RTC/CMOS chip.
- Test the flexible disc controller subsystem.
- Test the 80287 co-processor if present.

The power on self test performs tests on various sub-systems in the hardware when power is switched on or when the system is reset. If a problem is detected, a 4 digit hex error code is displayed. (In order for the code to be displayed, the video display adapter must be a multimode, a monochrome, or a color adaptor.) These codes are listed in table 10.2.

POST then compares the configuration information stored in the CMOS memory with the actual system. If a discrepancy is found, a message will be displayed instructing the user to run the SETUP utility. For example, if the CMOS memory indicates two flexible disc drives present, but the system contains only one, the message will be displayed.

If the POST process is initiated by a soft reset, the RAM tests are not executed. This portion of POST determines the amount of system memory and performs a test of that memory. In all other aspects, POST executes the same for power-on, hard reset, and soft reset.

10.5 System Generation (SYSGEN)

When the POST code module has completed its tasks, it initiates the system generation (SYSGEN) process. The SYSGEN process initializes the system software, then initiates the boot process. In general, the system data structures are initialized by the SYSGEN process, whereas the system hardware is initialized by the POST process. For example, the STD-BIOS and EX-BIOS data areas are initialized by the SYSGEN process. SYSGEN initializes the following items:

- Interrupt vectors
- STD-BIOS data area
- EX-BIOS data area

The interrupt vectors are initialized to their default values. Processor interrupt vectors are initialized to their appropriate service routines. Hardware interrupt vectors are initialized to their service routines, or a null routine if they are unused. The interrupt vectors used to access the STD-BIOS drivers are initialized to their respective driver entry points.

The STD-BIOS data area fields are initialized to their default values. Configuration dependent fields such as the base I/O address of the serial and parallel ports, current video mode, etc. are initialized at this time.

The EX-BIOS data area is set up next in the SYSGEN process. Initializing the EX-BIOS data area consists of several distinct steps as outlined below.

Table 10.2

Diagnostic Error Codes Displayed by POST

Error Code	Test	Description
0001 to 000FH	80286 chip	80286 chip failed.
0010	ROM checksum	ROM 0 fails checksum test.
0011	ROM checksum	ROM 1 fails checksum test.
0110 to 012FH	RTC test	Real-time clock failed.
0200 to 02FFH	CMOS test	Real-time clock failed.
0300 to 037FH	8041 test	8041 keyboard controller failed.
0401	System error.	Could not set A20 line.
1000 to 12FFH	Timer chip test	Timer chip failed
2110 to 211FH	DMA test	DMA chip 1 failed.
2120 to 212FH	DMA test	DMA chip 2 failed.
2131H	DMA test	DMA chip 1 failed.
2132H	DMA test	DMA chip 2 failed.
2210 to 2217H	DMA test	Page register failed.
3000 to 30FFH	HP-HIL controller	HP-HIL controller failed.
4000 to 400FH	RAM test	128k bank 0 d0-d3
4010 to 40F0H	RAM test	128k bank 0 d4-d7
4100 to 410FH	RAM test	128k bank 0 d8-d11
4110 to 41F0H	RAM test	128k bank 0 d12-d15
4200 to 420FH	RAM test	128k bank 1 d0-d3
4210 to 42F0H	RAM test	128k bank 1 d4-d7
4300 to 430FH	RAM test	128k bank 1 d8-d11
4310 to 43F0H	RAM test	128k bank 1 d12-d15
4400 to 440FH	RAM test	128k bank 2 d0-d3
4410 to 44F0H	RAM test	128k bank 2 d4-d7
4500 to 450FH	RAM test	128k bank 2 d8-d11
4510 to 45F0H	RAM test	128k bank 2 d12-d15
4600 to 460FH	RAM test	128k bank 3 d0-d3
4610 to 46F0H	RAM test	128k bank 3 d4-d7
4700 to 470FH	RAM test	128k bank 3 d8-d11
4710 to 47F0H	RAM test	128k bank 3 d12-d15

Error Code	Test	Description
4800 to 480FH	RAM test	128k bank 4 d0-d3
4810 to 48F0H	RAM test	128k bank 4 d4-d7
4900 to 490FH	RAM test	128k bank 4 d8-d11
4910 to 49F0H	RAM test	128k bank 4 d12-d15
5000 to 5FFFH	Reserved for Manufacturing test.	
6100 to 6113H	RAM test	Address line defined by the last 2 digits failed. (Hex) i.e. 6111 = address line 11h = a17 failed.
7100 to 71FFH	8259 test	Master 8259 failed.
7200 to 72FFH	8259 test	Industry Standard (STD) slave failed.
7300 to 73FFH	8259 test	HP slave failed.
7400H	8259 test	Master 8259 failed.
7500H	8259 test	Industry Standard (STD) slave failed.
7500H	8259 test	HP slave failed.
8000 to 82FFH	Reserved for manufacturing test.	
8300 to 83FFH	Hard disc	Controller/drive failed.
8400 to 8FFFH	Reserved for manufacturing test.	
9001 to 91FFH	Flexible Disc	Flexible disc controller problem.
9200 to 9FFFH	Reserved for manufacturing test.	
A002 to A00FH	80287 co-proc.	Internal problem with 287.
B001 to B007H	Multimode	Video adapter problem.
B008H	Multimode	Video adapter RAM problem.
C000 to CFFFH	Extended RAM	Extended RAM failure.
Where:	0C000 to 0C0FFH = > even byte is bad 0C100 to 0C1FFH = > odd byte is bad. xx00 to xxFEH = > bad RAM at address 00*10000H to 0FE*10000H	

example:	if error = 0C124H then:
	1 = > odd byte is bad.
	24 = > error is in 128K bank starting at address:
	024H*10000H = 0240 000H
	
	if error = 0C0F1H then:
	0 = > even byte is bad.
	F1 = > error is in 128K bank starting at address:
	0F1H*10000H = 0F10 000H	

0D000 to 0FFFFH	Reserved for manufacturing test.	

10.5.1 Memory Allocation

The first step in the process is to allocate system memory for the EX-BIOS data area. This memory allocation algorithm has two important features. First, by taking the memory size stored in CMOS memory into consideration, it allows large driver data areas to be allocated in the EX-BIOS data area. This method of expanding the EX-BIOS data area is explained in Section 9. Second, it prevents invalid CMOS memory size data from preventing the system from booting. If the CMOS memory size is set (using the SETUP utility or writing directly to the CMOS memory) such that there is insufficient room for the EX-BIOS data area, this algorithm will adjust the value and write the new value to CMOS memory. The EX-BIOS data area is required to support the EX-BIOS extended features.

There are three important variables in this calculation.

- RAM__SIZE—This is the top of actual system memory. It is usually 256, 512, or 640 KB and will always be an even multiple of 64 KB.
- EX-BIOS__SIZE—This variable is the size of the EX-BIOS data area, which is 4 KB in its default configuration.
- CMOS__SIZE—This is the memory size stored in CMOS.

The CMOS__SIZE is checked for validity. If it is between 4 KB and 64 KB from RAM__SIZE, this value is used as the base of the EX-BIOS data area. If CMOS__SIZE is more than 64 KB from RAM__SIZE, the base of the EX-BIOS data area is located 64 KB below the top of actual system memory. Finally, if CMOS__SIZE is less than 4 KB from the top of RAM__SIZE (or greater than the top of actual memory), the base of the EX-BIOS data area is located 4 KB from the top of system memory. The following formulas show this relationship:

```
If (RAM__SIZE—CMOS__SIZE) > 4KB and < 64KB
then EX-BIOS__SIZE = (RAM__SIZE—CMOS__SIZE).

If (RAM__SIZE—CMOS__SIZE) > 64KB
then EX-BIOS__SIZE = 64KB.

If (RAM__SIZE—CMOS__SIZE) < 4KB
then EX-BIOS__SIZE = 4KB.
```

The following examples illustrate this relationship:

> In a 640 KB system, if CMOS__SIZE is 512 KB then the EX-BIOS__SIZE data area starts at 600 KB. This leaves an 88 KB free area between the EX-BIOS__SIZE data area and the memory allocated to DOS.

In a 640 KB system if CMOS__SIZE is 620 KB then the EX-BIOS__SIZE data area starts at 620 KB. In this case the EX-BIOS__SIZE data area occupies all the area between the top of RAM and the memory allocated to DOS.

10.5.2 HP__VECTOR__TABLE Initialization

Once the EX-BIOS data area has been allocated, and its base address determined, the HP__VECTOR__TABLE is constructed. An image of the default HP__VECTOR__TABLE is stored in the system ROM. This image is transferred from ROM to the base of the EX-BIOS data area. All free and reserved vectors are initialized to point at V__DOLITTLE, a null routine. Some of these vectors will be initialized to other drivers later in the SYSGEN process.

10.5.3 EX-BIOS Driver Initialization

The next step in the SYSGEN process is the initialization of the EX-BIOS drivers. Each driver is called with the SF__INIT subfunction. Some of the EX-BIOS drivers add vectors to the table when called to initialize. For example, the V__HPHIL driver initializes the vector addresses reserved for the HP-HIL physical device drivers. The HP__VECTOR__TABLE is fully initialized to its default state when each driver has been called in this manner. Additional drivers may be added or substituted by application programs or system software utilizing the vector maintenance functions of V__SYSTEM (refer to Section 9 for a description of these functions).

10.5.4 Option ROM Module Integration

The ROM BIOS architecture allows code modules residing on adapter cards to be integrated into the system. These ROM modules must be in the system address range of 0C0000H—0DFFFFH. (Note that only video adapter cards can have base address in the range of 0C0000H through 0C7FFFH). In addition to ROM modules located on adapter cards, the processor extension card contains sockets for additional ROMs. These ROMs are addressed from 0E0000H—0EFFFFH. ROM modules located on adapter cards or on the processor extension card are integrated into the system in the same manner.

All ROM modules contain a header and checksum byte. The header format is shown below:

Byte 0—55H
Byte 1—0AAH
Byte 2—Length of ROM module in 512 byte blocks.
Byte 3—Initialization entry point.

Bytes 0 and 1 are signature bytes. All ROM modules must contain this signature at the start of the header in order to be identified by the SYSGEN process.

Byte 2 of the header contains the number of 512 byte blocks in the ROM module, except the ROM module located on the processor extension card (memory address 0E0000H). Byte 2 in that ROM module header is reserved.

During the boot process, the address range from 0C8000H to 0DFFFFH is scanned in 2 KB blocks looking for valid option ROM headers. In addition, memory location 0E0000H is also examined for a valid header. Since the scan does not proceed past 0E0000H, only one ROM module can reside in the address range 0E0000H to 0EFFFFH. The processor extension card will accept two different size ROMs; 32 KB or 64 KB. If a 32 KB part is installed, the ROM will appear in the system address space starting at location 0E8000H instead of 0E0000H. Therefore, the 32 KB ROM will not be integrated into the system by SYSGEN.

If a valid ROM header is found, a checksum is computed for the ROM module. This is done by summing each byte in the ROM module. The sum of all the bytes in the ROM, including the checksum byte, must equal 0. For ROM modules located from 0C0000H to 0DFFFFH, the checksum is computed for the number of bytes indicated in the length field of the header. For a ROM module located from 0E0000H to 0EFFFFH this checksum is calculated on the entire 64 KB of address space.

If the checksum is valid, a FAR call to byte 3 of the module is is performed. The ROM module should perform any initialization required and then execute a RETF instruction.

This integration process allows option ROMs to install vectors in either the HP__VECTOR__TABLE or the low memory interrupt vectors. This re-vectoring process is the typical method used to integrate ROM modules into the system.

10.6 Boot Process (INT 19H)

The boot process loads the operating system. The ROM BIOS INT 19H loads the boot sector from drive ''A:'' or ''C:''. This sector must contain the bootstrap loader for the operating system. Control is then passed to the code loaded from the boot sector. This code is responsible for loading the operating system. Refer to the appropriate operating system reference documentation for additional information on its boot process.

10.6.1 Booting From a Flexible Disc

The INT 19H driver attempts to read the boot sector from Drive ''A:'' (disc 0). It will retry the read four times before failing. The boot sector on flexible discs is located on Side 0, Track 0, Sector 1. Table 10.3 contains a description of the contents of a valid boot sector. If drive ''A:'' contains a disc that does not have a valid boot sector, then the system will report the error message:

> Non-System disc or disc error
> Replace and strike any key when ready.

If a valid boot sector is found, it is read into memory starting at location 07C0H:0000H (07C00H) and control is transferred through a FAR JUMP to location 07C0H:0000H. It is the responsibility of this code to load the rest of the operating system into memory.

10.6.2 Booting From a Hard Disc

If the flexible disc drive does not contain a disc, the system will attempt to boot from the hard disc. Booting from a hard disc is a two step process. First, the active partition must be determined, then the boot record is read from the active partition.

The hard disc can be divided into as many as four partitions. Each partition contains an operating system, programs, and data. Only one of the partitions can be active at any time. Partitions are added, deleted, activated, and deactivated using utilities provided with the respective operating systems. Partitions occupy a specified number of cylinders on the disc. For example, the optional 20 MB hard disc drive has 606 cylinders. One partition might occupy cylinders 0 through 303, while the second partition occupied cylinders 304 through 605.

Table 10.3

Boot Record

Offset	Size	Description
0000H	3 Bytes	Near JUMP instruction to boot code.
0003H	8 Bytes	OEM name and version number.
000BH	1 Word	Bytes per sector.
000DH	1 Byte	Sectors per allocation unit.
000EH	1 Word	Reserved sectors.
0011H	1 Byte	Number of File Allocation Tables (FATs).
0012H	1 Word	Number of root directory entries.
0014H	1 Word	Number of sectors in logical image.
0016H	1 Byte	Media descriptor.
0017H	1 Word	Number of FAT sectors.
0019H	1 Word	Sectors per track.
001BH	1 Word	Number of heads.
001DH	1 Word	Number of hidden sectors.
001FH	478 Bytes	Boot code.
01FEH	1 Word	55AAH signature word.

The first physical sector (cylinder 0, head 0, sector 1) of the hard disc contains the master boot record. The master boot record contains a code module and the disc partition table. The disc partition table contains the starting and ending cylinder of each of the disc partitions, as well as a flag that indicates whether the partition is active or not. Table 10.4 contains a description of the master boot record.

Table 10.4

Hard Disc Master Boot Record

Offset	Size	Description
0000H	446 Bytes	Master boot code.
01BEH	16 Bytes	Partition table entry #1.
01CEH	16 Bytes	Partition table entry #2.
01DEH	16 Bytes	Partition table entry #3.
01EEH	16 Bytes	Partition table entry #4.
01FEH	1 Word	0AA55H signature word.

A partition entry consists of 16 bytes. It contains information specifying the location of the partition, type of operating system, and a flag to indicate if the partition is active. Table 10.5 details the partition table entry.

Table 10.5

Partition Table Entry Record

Size	Description
1 Byte	Boot indicator.
1 Byte	Starting head number.
1 Byte	Starting sector number.
1 Byte	Starting cylinder number.*
1 Byte	System indicator.**
1 Byte	Ending head number.
1 Byte	Ending sector number.
1 Byte	Ending cylinder number.*
2 Words	Number of sectors in preceding partitions.
2 Words	Total number of sectors in partition.

* The actual cylinder number is a ten bit value composed of the cylinder byte plus the two most significant bits of the associated sector byte. These two bits are the most significant bits of the ten bit number.

** System indicators are:
00H = Unknown operating system
01H = DOS (12 bit FAT)
04H = DOS (16 bit FAT)

The INT 19H code will load the code module contained in the master boot record into memory, then transfer control to it. This code scans the data in the disc partition table to determine the active partition, and its starting cylinder. The first sector of the active partition becomes the logical boot sector of the partition, and it contains a boot record. The boot record has the same format as the boot record contained on a flexible disc, except that some of the parameters are adjusted for the increased capacity of the hard disc partition. Refer to table 10.3 for the format of a typical boot record.

APPENDICES

TABLE OF CONTENTS

APPENDIX A

A. BIOS INTERRUPTS

This appendix contains three tables. The first lists the interrupt vector assignments. The second lists each of the STD-BIOS interrupts with supported functions. The third lists the EX-BIOS drivers; their vector addresses, functions and subfunctions.

A.1 Interrupt Vector Assignments

Table A.1

Interrupt Vector Assignments

Address	Int	Function	Type*	Service	Routine**
000-003H	0	Divide by Zero	PI	STD-BIOS	(UI)
004-007H	1	Single Step	PI	STD-BIOS	(UI)
008-00BH	2	Nonmaskable Interrupt	PI	STD-BIOS	
00C-00FH	3	Breakpoint	PI	STD-BIOS	(UI)
010-013H	4	Arithmetic Overflow	PI	STD-BIOS	(UI)
014-017H	5	Print Screen	SW	STD-BIOS	(DRVR)
018-01BH	6	Invalid Opcode	PI	STD-BIOS	(UI)
01C-01FH	7	Reserved	PI	STD-BIOS	(UI)
020-023H	8	Timer Interrupt (IRQ 0)	HW	STD-BIOS	
024-027H	9	Keyboard ISR (IRQ 1)	HW	STD-BIOS	

Address	Int	Function		Type*	Service	Routine**
028-02BH	A	Reserved	(IRQ 2)	HW	STD-BIOS	
02C-02FH	B	Serial Port 1 ISR	(IRQ 3)	HW	STD-BIOS	(UI)
030-033H	C	Serial Port 0 ISR	(IRQ 4)	HW	STD-BIOS	(UI)
034-037H	D	Printer Port 1 ISR	(IRQ 5)	HW	STD-BIOS	(UI)
038-03BH	E	Diskette ISR	(IRQ 6)	HW	STD-BIOS	
03C-03FH	F	Printer Port 0 ISR	(IRQ 7)	HW	STD-BIOS	(UI)
040-043H	10	Video		SW	STD-BIOS	(DRVR)
044-047H	11	Equipment Check		SW	STD-BIOS	(DRVR)
048-04BH	12	Memory Size		SW	STD-BIOS	(DRVR)
04C-04FH	13	Diskette/Hard Disc		SW	STD-BIOS	(DRVR)
050-053H	14	Serial		SW	STD-BIOS	(DRVR)
054-057H	15	System Functions		SW	STD-BIOS	(DRVR)
058-05BH	16	Keyboard		SW	STD-BIOS	(DRVR)
05C-05FH	17	Printer		SW	STD-BIOS	(DRVR)
060-063H	18	Reserved		SW	N/A	(IRET)
064-067H	19	Boot		SW	STD-BIOS	(DRVR)
068-06BH	1A	Time and Date		SW	STD-BIOS	(DRVR)
06C-06FH	1B	Keyboard Break		SW	STD-BIOS	(IRET)
070-073H	1C	Timer Tick		SW	STD-BIOS	(IRET)
074-077H	1D	Video Parameter Table		PT	STD-BIOS	
078-07BH	1E	Diskette Parameter Table		PT	STD-BIOS	
07C-07FH	1F	Graphics Character Table		PT	STD-BIOS	
080-083H	20	Program Terminate		SW	DOS	
084-087H	21	DOS Function Calls		SW	DOS	
088-08BH	22	DOS Terminate Address		PT	DOS	
08C-08FH	23	DOS <CTRL>-<Break> Address		SW	DOS	
090-093H	24	DOS Critical Error		SW	DOS	
094-097H	25	DOS Absolute Disc Read		SW	DOS	
098-09BH	26	DOS Absolute Disc Write		SW	DOS	
09C-09FH	27	DOS Terminate Stay Resident		SW	DOS	
0A0-0CBH	28-32	Reserved for DOS		SW	DOS	
0CC-0CFH	33	HP Mouse Service		SW	EX-BIOS	(DRVR)
0D0-0FFH	34-3F	Reserved for DOS		SW	DOS	
100-103H	40	Alternate Diskette		SW	STD-BIOS	
104-107H	41	Hard Disc Parameter Table (0)		PT	STD-BIOS	

Address	Int	Function	Type*	Service	Routine**
108-117H	42-45	Reserved	SW	STD-BIOS	
118-11BH	46	Hard Disc Parameter Table (1)	PT	STD-BIOS	
11C-17FH	47-5F	Reserved	SW	STD-BIOS	
180-19FH	60-67	Reserved for User Programs	SW	N/A	
1A0-1A3H	68	8041 Service Request ISR	HW	EX-BIOS	
1A4-1A7H	69	Keyboard OBF ISR	HW	EX-BIOS	
1A8-1ABH	6A	Reserved	HW	EX-BIOS	
1AC-1AFH	6B	Reserved	HW	EX-BIOS	
1B0-1B3H	6C	HP-HIL Controller ISR	HW	EX-BIOS	
1B4-1B7H	6D	Reserved	HW	EX-BIOS	
1B8-1BBH	6E	Reserved	HW	EX-BIOS	
1BC-1BFH	6F	EX-BIOS Entry Point	SW	EX-BIOS	(DRVR)
1C0-1C3H	70	Real-time Clock ISR (IRQ 8)	HW	STD-BIOS	
1C4-1C7H	71	SW Redirected (IRQ 9)	HW	STD-BIOS	
1C8-1CBH	72	Reserved (IRQ 10)	HW	STD-BIOS	(UI)
1CC-1CFH	73	Reserved (IRQ 11)	HW	STD-BIOS	(UI)
1D0-1D3H	74	Reserved (IRQ 12)	HW	STD-BIOS	(UI)
1D4-1D7H	75	Coprocessor (IRQ 13)	HW	STD-BIOS	
1D8-1DBH	76	Hard Disc ISR (IRQ 14)	HW	STD-BIOS	(UI)
1DC-1DFH	77	Reserved (IRQ 15)	HW	STD-BIOS	(UI)
1E0-1FFH	78-7F	Not Used	SW	N/A	
200-3C3H	80-F0	Reserved	SW	N/A	
3C4-3FFH	F1-FF	Not Used	SW	N/A	

* PI —Processor interrupt
HW —Hardware interrupt
SW —Software interrupt
PT —Interrupt vector used as pointer to data.
N/A —Not applicable

**UI —Unused Interrupt ISR
IRET —Interrupt Returned
DRVR—Application callable Entry Point

The following table lists the STD-BIOS interrupt vectors, their usage and, where appropriate, their functions.

A.2 STD-BIOS Interrupts and Functions

Table A.2

STD-BIOS Interrupts and Functions

INT Hex	Function Equate	Function Value	Definition
00H			Divide by zero
01H			Single step
02H			Nonmaskable interrupt
03H			Breakpoint
04H			Arithmetic overflow
05H			Print screen
06H			Invalid opcode
07H			Reserved
08H			Timer interrupt
09H			Keyboard ISR
0AH			Reserved
0BH			Serial port 1 ISR
0CH			Serial port 0 ISR
0DH			Printer port 1 ISR
0EH			Diskette ISR
0FH			Printer port 0 ISR
10H	INT_VIDEO		Video
	F10_SET_MODE	00H	Set video mode
	F10_SET_CURSIZE	01H	Set cursor size
	F10_SET_CURPOS	02H	Set cursor position
	F10_RD_CURPOS	03H	Read cursor position
	F10_RD_PENPOS	04H	Read light-pen position
	F10_SET_PAGE	05H	Set active display page
	F10_SCROLL_UP	06H	Scroll rectangle up
	F10_SCROLL_DN	07H	Scroll rectangle down
	F10_RD_CHARATR	08H	Read character and attribute at cursor position
	F10_WR_CHARATR	09H	Write character and attribute at cursor position
	F10_WR_CHARCUR	0AH	Write character at cursor position
	F10_SET_PALLET	0BH	Set color pallet
	F10_WR_PIXEL	0CH	Write pixel

INT Hex	Function Equate	Function Value	Definition
	F10__RD__PIXEL	0DH	Read pixel
	F10__WR__CHARTEL	0EH	Write teletype character
	F10__GET__STMODE	0FH	Get video state and mode
		10H-12H	Reserved
			Write string functions
	F10__WRS__00	1300H	global attribute
	F10__WRS__01	1301H	global attribute, move cursor
	F10__WRS__02	1302H	individual attributes
	F10__WRS__03	1303H	individual attributes, move cursor
	F10__INQUIRE	6F00H	EX-BIOS present
	F10__GET__INFO	6F01H	Get video parameters
	F10__SET__INFO	6F02H	Set video parameters
	F10__MOD__INFO	6F03H	Modifies video parameters
	F10__GET__RES	6F04H	Report video resolution
	F10__XSET__MODE	6F05H	Set video resolution
11H	INT__EQUIPMENT		Equipment check
12H	INT__MEM__SIZE		Memory Size
	*** Note that both hard disc and *** diskette share interrupt 13H		
13H	INT__DISC		Disc Functions
	F13__RESET__DISC	00H	Reset Disc
	F13__RD__LSTATUS	01H	Read status of last operation
	F13__RD__SECTORS	02H	Read sectors
	F13__WR__SECTORS	03H	Write sectors
	F13__VR__SECTORS	04H	Verify sectors
	F13__FORMAT__FLEX	05H	Format flexible disc track
		06H	Reserved
	F13__FORMAT__HDISC	07H	Format hard disc
	F13__GET__HPARMS	08H	Get hard disc parameters
		09H-0BH	Reserved
	F13__TRACK__SEEK	0CH	Seek to track
	F13__ALT__RESET	0DH	Alternate hard disc reset
		0EH-014H	Reserved
	F13__GET__DASD	15H	Read disc type (DASD)
	F13__CHG__STATUS	16H	Get disc change line status
	F13__SET__DASD	17H	Set disc type for formatting (DASD)

INT Hex	Function Equate	Function Value	Definition
14H	INT__SERIAL		Serial
	F14__INIT	00H	Initialize serial port parameters
	F14__XMIT	01H	Send out one character
	F14__RECV	02H	Receive one character
	F14__STATUS	03H	Get serial port status
	F14__INQUIRE	6F00H	EX-BIOS present
	F14__EXINIT	6F01H	Initializes serial port (19.2 Kbaud)
	F14__PUT__BUFFER	6F02H	Write a buffer of data
	F14__GET__BUFFER	6F03H	Read a buffer of data
	F14__TRM__BUFFER	6F04H	Read a buffer of data, terminate on specified condition
15H	INT__SYSTEM		System functions
		00H	Unsupported (turn on cassette motor)
		01H	Unsupported (turn off cassette motor)
		02H	Unsupported (read data blocks)
		03H	Unsupported (write data blocks)
	F15__DEVICE__OPEN	80H	Device open
	F15__DEVICE__CLOSE	81H	Device close
	F15__PROG__TERM	82H	Program termination
	F15__WAIT__EVENT	83H	Event wait
	F15__JOYSTICK	84H	Joystick support
	F15__SYS__REQ	85H	System request key pressed
	F15__WAIT	86H	Wait fixed amount of time
	F15__BLOCK__MOVE	87H	Extended memory transfer
	F15__GET__XMEM__SIZE	88H	Get extended memory size
	F15__ENTER__PROT	89H	Switch to protected mode
	F15__DEV__BUSY	91H	Device busy hook
	F15__INT__COMPLETE	8BH	Set Interrupt Completed Flag
16H	INT__KBD		Keyboard
	F16__GET__KEY	00H	Read keycode from keyboard buffer
	F16__STATUS	01H	Report status of keyboard buffer
	F16__KEY__STATE	02H	Get key modifier status
	F16__INQUIRE	6F00H	EX-BIOS present
	F16__DEF__ATTR	6F01H	Report default typematic values
	F16__GET__ATTR	6F02H	Report typematic values
	F16__SET__ATTR	6F03H	Set typematic values
	F16__DEF__MAPPING	6F04H	Report default translator assignments

INT Hex	Function Equate	Function Value	Definition
	F16_GET_MAPPING	6F05H	Report translator assignments
	F16_SET_MAPPING	6F06H	Set translator assignments
	F16_SET_XLATORS	6F07H	Set CCP and softkey pads
	F16_KBD	6F08H	Report keyboard information
	F16_KBD_RESET	6F09H	Reset keyboard to defaults
17H	INT_PRINTER		Printer
	F17_PUT_CHAR	00H	Send printer one byte
	F17_INIT	01H	Initialize printer port
	F17_STATUS	02H	Get printer port status
	F17_INQUIRE	6F00H	EX-BIOS present
		6F01H	Reserved
	F17_PUT_BUFFER	6F02H	Write a buffer to printer port
		6F03H	Reserved
		6F04H	Reserved
18H			Reserved
19H	INT_BOOT		Boot
1AH	INT_CLOCK		Time and date
	F1A_RD_CLK_CNT	00H	Read current clock count
	F1A_SET_CLK_CNT	01H	Set current clock count
	F1A_GET_RTC	02H	Read real-time clock
	F1A_SET_RTC	03H	Set real-time clock
	F1A_GET_DATE	04H	Read date from real-time clock
	F1A_SET_DATE	05H	Set date in real-time clock
	F1A_SET_ALARM	06H	Set alarm
	F1A_RESET_ALARM	07H	Reset alarm
1BH			Keyboard break
1CH			Timer tick
1DH			Video parameter table
1EH			Diskette parameter table
1FH			Graphics character table
20H			Program terminate
21H			DOS function calls
22H			DOS terminate address
23H			DOS <CTRL>-<Break> address
24H			DOS critical error
25H			DOS absolute disc read
26H			DOS absolute disc write
27H			DOS terminate stay resident
28H-32H			Reserved for DOS

INT Hex	Function Equate	Function Value	Definition
33H	INT_HPMOUSE		HP Mouse service
	F33_INSTALL	00H	Mouse installed flag
	F33_ENABLE	01H	Put cursor on screen
	F33_DISABLE	02H	Turn off cursor
	F33_REPORT_DATA	03H	Get position/button information
	F33_PUT_CURSOR	04H	Position the cursor
	F33_REPORT_PRESS	05H	Report button press status
	F33_REPORT_RELEASE	06H	Report button release status
	F33_SET_HORIZ	07H	Set min/max horizontal values
	F33_SET_VERT	08H	Set min/max vertical values
	F33_GRAPH_CURSOR	09H	Define graphics cursor
	F33_TEXT_CURSOR	0AH	Define text cursor
	F33_MOTION	0BH	Report motion counters
	F33_SET_USR	0CH	Define user subroutine
	F33_ENABLE_LIGHT	0DH	Unsupported
	F33_DISABLE_LIGHT	0EH	Unsupported
	F33_RATIO	0FH	Set pixel movement ratio
	F33_COND_OFF	10H	Define conditional off area
	F33_RESERVED	11H	Reserved
	F33_XTEND_GCSR	12H	Extended sprite graphics entry point
	F33_SPEED	13H	Sets mouse movement doubling
	F33_INQUIRE	6F00H	EX-BIOS mouse driver present
34H-3FH			Reserved for DOS
40H			Alternate Diskette
41H			Hard Disc Parameter Table (0)
42H-45H			Reserved
46H			Hard Disc Parameter Table (1)
47H-5FH			Reserved
60H-67H			Reserved for User Programs
68H			8041 Service Request ISR
69H			Keyboard OBF ISR
6AH			Reserved
6BH			Reserved
6CH			HP-HIL Controller ISR
6DH			Reserved
6EH			Reserved
6FH	HP_ENTRY		EX-BIOS Entry Point
70H			Real-time Clock ISR (IRQ 8)
71H			SW redirected (IRQ 9)
72H			Reserved (IRQ 10)
73H			Reserved (IRQ 11)

INT Hex	Function Equate	Function Value	Definition	
74H			Reserved	(IRQ 12)
75H			Coprocessor	(IRQ 13)
76H			Hard Disc ISR	(IRQ 14)
77H			Reserved	(IRQ 15)
78H-7FH			Not Used	
80H-F0H			Reserved	
F1H-FFH			Not Used	

A.3 EX-BIOS Drivers and Functions

Many additional features of the HP system can be accessed through the software interrupt INT 6FH. To call the EX-BIOS extensions, the BP register must contain the vector address of the desired driver, the AH register must contain the function code, and the AL register must contain the subfunction code. The rest of the registers are available for passing data and returning data to and from the routine.

In general the AX, BP and DS registers are not preserved. They must be preserved by the calling routine if it needs them. See Section 2 for an example showing how EX-BIOS drivers are called.

Table A.3

EX-BIOS Drivers and Functions

Vector Address	Func. Value	Function Equate	Definition
0000H		V__SCOPY	Copyright Notice Routine
0006H		V__DOLITTLE	NOP Routine (IRET)
000CH		V__PNULL	Null device driver
0012H		V__SYSTEM	System Management Functions
0012H	00	F__ISR	Interrupt service routine (unsupported)
0012H	02	F__SYSTEM	Standard driver functions
0012H	02/00	SF__INIT	System initialization

Vector Address	Func. Value	Function Equate	Definition
0012H	04	F_INS_BASEHPVT	Returns HP_VECTOR_TABLE segment
0012H	06	F_INS_XCHGFIX	Exchanges fixed table entries
0012H	08	F_INS_XCHGRSVD	Sets next "reserved" entry in table
0012H	0A	F_INS_XCHGFREE	Sets next "free" entry in table
0012H	0C	F_INS_FIXOWNDS	Install fixed vector, user supplied DS
0012H	0E	F_INS_FIXGETDS	Install fixed vector, system supplies DS
0012H	10	F_INS_FIXGLBDS	Install fixed vector, DS set to global data area
0012H	12	F_INS_FREEOWNDS	Install next free vector, user supplies DS
0012H	14	F_INS_FREEGETDS	Install next free vector, system supplies DS
0012H	16	F_INS_FREEGLBDS	Install next free vector, DS set to global data area
0012H	18	F_INS_FIND	Search for matching device header
0012H	1A		Reserved*
0012H	1C		Reserved*
0012H	1E	F_RAM_GET	Get EX-BIOS memory pool address and size
0012H	20	F_RAM_RET	Set memory pool address and size
0012H	22	F_CMOS_GET	Read and verify CMOS memory
0012H	24	F_CMOS_RET	Write to CMOS memory
0012H	26		Reserved*
0012H	28		Reserved*
0012H	2A	F_YIELD	Just returns
0012H	2C		Reserved*
0012H	2E		Reserved*
0012H	30	F_SND_CLICK_ENABLE	Enable keyclick
0012H	32	F_SND_CLICK_DISABLE	Disable keyclick (Default)
0012H	34	F_SND_CLICK	Execute keyclick if enabled
0012H	36	F_SND_BEEP_ENABLE	Enables beep
0012H	38	F_SND_BEEP_DISABLE	Disables beep
0012H	3A	F_SND_BEEP	Beeps if enabled
0012H	3C	F_SND_SET_BEEP	Sets beep frequency
0012H	3E	F_SND_TONE	Produce tone, user suppled duration and frequency
0012H	40	F_STR_GET_FREE_INDEX	Return next free string index
0012H	42	F_STR_DEL_BUCKET	Delete bucket string list
0012H	44	F_STR_PUT_BUCKET	Add bucket to current string list

Vector Address	Func. Value	Function Equate	Definition
0012H	46	F_STR_GET_STRING	Search the list for index, return string
0012H	48	F_STR_GET_INDEX	Search list for a string, return index
0018H			Reserved*
001EH		V_S8259	8259 interrupt controller support
001EH	00	F_ISR	Unsupported
001EH	02	F_SYSTEM	System functions
001EH	02/00	SF_INIT	Initialize HP slave 8259A
001EH	02/02	SF_START	Enable HP slave 8259A interrupts
001EH	02/06	SF_VERSION_DESC	Report HP version number
001EH	04	F_IO_CONTROL	Entry point to I/O control functions
001EH	04/00	SF_ENABLE_SVC	Unmask svc/8041 interrupt
001EH	04/02	SF_DISABLE_SVC	Mask svc/8041 interrupt
001EH	04/04	SF_ENABLE_KBD	Unmask keyboard INT 9 interrupt
001EH	04/06	SF_DISABLE_KBD	Mask keyboard INT 9 interrupt
001EH	04/08	SF_ENABLE_HPHIL	Unmask HP-HIL interrupt
001EH	04/0A	SF_DISABLE_HPHIL	Mask HP-HIL interrupt
0024H			Reserved*
002AH		V_SINPUT	Inquire Commands
002AH	00	F_ISR	Pass ISR Event Record to physical driver
002AH	02	F_SYSTEM	System Functions
002AH	02/00	SF_INIT	Supported
002AH	04	F_IO_CONTROL	Entry point to I/O control functions
002AH	04/00	SF_DEF_LINKS	Set header link fields to system defaults
002AH	04/02	SF_GET_LINKS	Return device header link field entries
002AH	04/04	SF_SET_LINKS	Set device header link field entries
002AH	06	F_INQUIRE	Return describe record for an HP-HIL device
002AH	08	F_INQUIRE_ALL	Return device IDs for all HP-HIL devices present
002AH	0A	F_INQUIRE_FIRST	Return vector address of first HP-HIL device driver
002AH	0C	F_REPORT_ENTRY	Report entry point of PGID

Vector Address	Func. Value	Function Equate	Definition
0030H			Reserved*
0036H		V__QWERTY	QWERTY keypad translator
0036H	00	F__ISR	Translates to PC scan code.
0036H	02	F__SYSTEM	System functions
0036H	02/06	SF__VERSION__DESC	Reports HP version number
003CH		V__SOFTKEY	Physical HP softkey translator
003CH	00	F__ISR	Translates to PC scan code
003CH	02	F__SYSTEM	System functions
003CH	02/00	SF__INIT	Driver initialization
003CH	02/06	SF__VERSION__DESC	Report HP version number
0042H		V__FUNCTION	Industry standard function key translator
0042H	00	F__ISR	Logical Interrupt
0042H	02	F__SYSTEM	System functions
0042H	02/06	SF__VERSION__DESC	Report HP version number
0048H		V__NUMPAD	Ind. standard numeric Key Pad Translator
0048H	00	F__ISR	Logical Interrupt
0048H	02	F__SYSTEM	System functions
0048H	02/06	SF__VERSION__DESC	Reports HP version number
004EH		V__CCP	Cursor Control Key Pad Translator
004EH	00	F__ISR	Logical Interrupt
004EH	02	F__SYSTEM	System functions
004EH	02/06	SF__VERSION__DESC	Reports HP version number
0054H		V__SVIDEO	Video Functions
0054H	00	F__ISR	Interrupt service routine
0054H	02	F__SYSTEM	Standard driver functions
0054H	04	F__IO__CONTROL	Driver dependent control functions
0054H	04/00	SF__VID__ID__HP	Returns the value "HP" in BX register
0054H	04/02	SF__VID__GET__INFO	Return video display adapter information
0054H	04/04	SF__VID__SET__INFO	Set info. on Extended Control Register of the Multimode Video Adapter
0054H	04/06	SF__VID__MOD__INFO	Modify Extended Control Register of Multimode Video Adapter
0054H	04/08	SF__VID__GET__RES	Get the resolution of active video adaptor
0054H	04/0A	SF__VID__SET__MODE	Set video mode of active Display adapter

Vector Address	Func. Value	Function Equate	Definition
005AH		V_STRACK	Sprite control
005AH	00	F_ISR	Update sprite
005AH	02	F_SYSTEM	System functions
005AH	02/00	SF_INIT	Initialize driver
005AH	02/02	SF_START	Start driver
005AH	04	F_TRACK_INIT	Sets tracking to default state
005AH	06	F_TRACK_ON	Enables tracking
005AH	08	F_TRACK_OFF	Disables tracking
005AH	0A	F_DEF_MASKS	Define sprite masks
005AH	0C	F_SET_LIMITS_X	Set max/min horizontal values
005AH	0E	F_SET_LIMITS_Y	Set max/min vertical values
005AH	10	F_PUT_SPRITE	Display sprite
005AH	12	F_REMOVE_SPRITE	Remove sprite from display
0060H		V_EVENT_TOUCH	Application access to touch events
0066H		V_EVENT_TABLET	Application access to tablet events
006CH		V_EVENT_POINTER	Application access to pointer events
0072H-84H			Reserved*
008AH		V_CCPCUR	Cursor control pad translator
008AH	00	F_ISR	Logical Interrupt
008AH	02	F_SYSTEM	System functions
008AH	02/06	SF_VERSION_DESC	Returns HP version number
0090H		V_RAW	Return untranslated CCP data
0090H	00	F_ISR	Logical Interrupt
0090H	02	F_SYSTEM	System functions
0090H	02/06	SF_VERSION_DESC	Returns HP version number
0096H		V_CCPNUM	Translate scancodes from Numeric Pad
0096H	00	F_ISR	Logical Interrupt
0096H	02	F_SYSTEM	System functions
0096H	02/06	SF_VERSION_DESC	Returns HP version number
009CH		V_OFF	Discards CCP and HP softkey scancodes
009CH	00	F_ISR	Logical Interrupt.
009CH	02	F_SYSTEM	System functions
009CH	02/06	SF_VERSION_DESC	Returns HP version number
00A2H		V_CCPGID	Translates CCP data to T_REL16 data
00A8H		V_SKEY2FKEY	HP softkeys to function key translator

Vector	Func.	Function	
Address	Value	Equate	Definition
00A8H	00	F_ISR	Logical Interrupt
00A8H	02	F_SYSTEM	System functions
00A8H	02/06	SF_VERSION_DESC	Returns HP version number
00AEH		V_8041	8041/keyboard interface. provides HP extensions to INT 16H
00AEH	00	F_ISR	Processes ISR event record
00AEH	02	F_SYSTEM	System functions
00AEH	02/00	SF_INIT	Initializes driver
00AEH	02/02	SF_START	Driver Start-up
00AEH	02/06	SF_VERSION_DESC	Reports HP version number
00AEH	04	F_IO_CONTROL	Driver Dependant Functions
00AEH	04/00	through 04/08	Reserved*
00AEH	04/0A	SF_CREAT_INTR	Create interval entry
00AEH	04/0C	SF_DELET_INTR	Delete interval entry
00AEH	04/0E	SF_ENABL_INTR	Enable interval
00AEH	04/10	SF_DISBL_INTR	Disable interval
00AEH	04/12	SF_SET_RAMSW	Set RAM switch to one (1)
00AEH	04/14	SF_CLR_RAMSW	Set RAM switch to zero (0)
00AEH	04/16	SF_SET_CRTSW	Set CRT switch to one (1)
00AEH	04/18	SF_CLR_CRTSW	Set CRT switch to zero (0)
00AEH	04/1A	SF_PASS_THRU	Pass data byte to 8041
00AEH	04/1C	through 04/2E	Reserved*
00B4H		V_PGID_CCP	Translate GID info to cursor control pad format
00BAH		V_LTABLET	Application interface to Tablet
00BAH	00	F_ISR	Logical Interrupt
00BAH	02	F_SYSTEM	System functions
00BAH	02/00	SF_INIT	Initialize the driver data area
00BAH	02/02	SF_START	Start driver
00BAH	02/04	SF_REPORT_STATE	Report state of device
00BAH	02/06	SF_VERSION_DESC	Report driver version number
00BAH	02/08	SF_DEF_ATTR	Set default logical scaling attributes
00BAH	02/0A	SF_GET_ATTR	Get scaling attributes
00BAH	02/0C	SF_SET_ATTR	Set scaling attributes
00BAH	04	F_IO_CONTROL	I/O Control Functions
00BAH	04/00	SF_LOCK	Unsupported
00BAH	04/02	SF_UNLOCK	Unsupported
00BAH	04/04	SF_TRACK_ON	Turns cursor track on
00BAH	04/06	SF_TRACK_OFF	Turns cursor track off
00BAH	04/08	SF_CREATE_EVENT	Establish a new routine to be called on logical device events
00BAH	04/0A	SF_EVENT_ON	Enable event call to parent driver

Vector Address	Func. Value	Function Equate	Definition
00BAH	04/0C	SF_EVENT_OFF	Disable event call to parent driver
00BAH	04/0E	SF_CLIPPING_ON	Enable logical device clipping
00BAH	04/10	SF_CLIPPING_OFF	Disable logical device clipping
00BAH	06	F_SAMPLE	Report absolute position of GID
00C0H		V_LPOINTER	Application interface to Pointer/ Mouse
00C0H	00	F_ISR	Logical Interrupt
00C0H	02	F_SYSTEM	System functions
00C0H	02/00	SF_INIT	Initialize the driver data area
00C0H	02/02	SF_START	Start driver
00C0H	02/04	SF_REPORT_STATE	Report state of device
00C0H	02/06	SF_VERSION_DESC	Report driver version number
00C0H	02/08	SF_DEF_ATTR	Set default logical scaling attributes
00C0H	02/0A	SF_GET_ATTR	Get scaling attributes
00C0H	02/0C	SF_SET_ATTR	Set scaling attributes
00C0H	04	F_IO_CONTROL	I/O Control Functions
00C0H	04/00	SF_LOCK	Unsupported
00C0H	04/02	SF_UNLOCK	Unsupported
00C0H	04/04	SF_TRACK_ON	Turn cursor track on
00C0H	04/06	SF_TRACK_OFF	Turn cursor track off
00C0H	04/08	SF_CREATE_EVENT	Establish a new routine to be called on logical device events
00C0H	04/0A	SF_EVENT_ON	Enable event call to parent driver
00C0H	04/0C	SF_EVENT_OFF	Disable event call to parent driver
00C0H	04/0E	SF_CLIPPING_ON	Enable logical device clipping
00C0H	04/10	SF_CLIPPING_OFF	Disable logical device clipping
00C0H	06	F_SAMPLE	Report absolute position GID
00C6H		V_LTOUCH	Application interface to Touch Screen
00C6H	00	F_ISR	Logical Interrupt
00C6H	02	F_SYSTEM	System functions
00C6H	02/00	SF_INIT	Initialize the driver data area
00C6H	02/02	SF_START	Start driver
00C6H	02/04	SF_REPORT_STATE	Report state of device
00C6H	02/06	SF_VERSION_DESC	Report driver version number
00C6H	02/08	SF_DEF_ATTR	Set default logical scaling attributes
00C6H	02/0A	SF_GET_ATTR	Get scaling attributes
00C6H	02/0C	SF_SET_ATTR	Set scaling attributes
00C6H	04	F_IO_CONTROL	I/O Control functions
00C6H	04/00	SF_LOCK	Unsupported

Vector Address	Func. Value	Function Equate	Definition
00C6H	04/02	SF_UNLOCK	Unsupported
00C6H	04/04	SF_TRACK_ON	Turn cursor track on
00C6H	04/06	SF_TRACK_OFF	Turn cursor track off
00C6H	04/08	SF_CREATE_EVENT	Establish a new routine to be called on logical device events
00C6H	04/0A	SF_EVENT_ON	Enable event call to parent driver
00C6H	04/0C	SF_EVENT_OFF	Disable event call to parent driver
00C6H	04/0E	SF_CLIPPING_ON	Enable logical device clipping
00C6H	04/10	SF_CLIPPING_OFF	Disable logical device clipping
00C6H	06	F_SAMPLE	Report absolute position of GID
00CCH		V_LHPMOUSE	Interface to Microsoft Mouse driver
00CCH	00	F_ISR	Logical Interrupt
00CCH	02	F_SYSTEM	System Functions
00CCH	02/00	SF_INIT	Initializes driver
00CCH	02/02	SF_START	Starts driver
00CCH	04	F_IO_CONTROL	I/O control driver functions
00CCH	04/00	SF_MOUSE_COM	BIOS mouse install function
00CCH	04/02	SF_MOUSE_OVERRIDE	Set speed factor
0108H		V_NULL	No driver
010EH			Reserved*
0114H		V_HPHIL	Setup HP-HIL to INPUT driver linkage
0114H	00	F_ISR	Logical Interrupt
0114H	02	F_SYSTEM	System Functions
0114H	02/00	SF_INIT	Initializes the driver data area
0114H	02/04	SF_REPORT_STATE	Reports state of device
0114H	02/06	SF_VERSION_DESC	Reports driver version number
0114H	02/0E	SF_OPEN	Put driver in open state
0114H	02/10	SF_CLOSE	Put driver in closed state
0114H	04	F_IO_CONTROL	I/O control to driver
0114H	04/06	SF_CRV_RECONFIGURE	Forces HP-HIL to reconfigure all devices
0114H	04/08	SF_CRV_WR_PROMPTS	Write a prompt to a device
0114H	04/0A	SF_CRV_WR_ACK	Write an acknowledge to a device
0114H	04/0C	SF_CRV_REPEAT	Sets either 30Hz or 60Hz repeat rate
0114H	04/0E	SF_CRV_DISABLE_REPEAT	Cancel keyboard repeat rate
0114H	04/10	SF_CRV_SELF_TEST	Issue self-test command to physical device

Vector Address	Func. Value	Function Equate	Definition
0114H	04/12	SF_CRV_REPORT_STATUS	Get status from any HP-HIL device that needs to report
0114H	04/14	SF_CRV_REPORT_NAME	Returns the ASCII name for a device
0114H	04/16	SF_KEYBOARD_REPEAT	Set typematic values
0114H	04/18	SF_KEYBOARD_LED	Sets keyboard LED states
0114H	06	F_PUT_BYTE	Write one byte to specified HP-HIL device
0114H	08	F_GET_BYTE	Read one byte from specified HP-HIL device
0114H	0A	F_PUT_BUFFER	Write a string of bytes to HP-HIL device
011AH-1C2H			Reserved*
1C8H-228H			Vectors available (16)
xxxH**		Keyboard Driver	Processes scancodes form HP-HIL driver
	00	F_ISR	Logical Interrupt
	02	F_SYSTEM	System Functions
	02/00	SF_INIT	Driver initialization
	02/06	SF_VERSION_DESC	Reports HP version number
xxxH**		HP-HIL driver vectors 1 thru 7	Physical HP-HIL driver vectors
	00	F_ISR	Logical Interrupt
	02	F_SYSTEM	System functions
	02/00	SF_INIT	Initialize driver
	02/02	SF_START	Start driver
	02/04	SF_REPORT_STATE	Unsupported
	02/06	SF_VERSION_DESC	Report HP version number
xxxH**		Available Vectors	Inquiry on availability of free vector in HP_VECTOR_TABLE

*Vectors marked reserved should not be used.

**Vectors with addresses xxxH do not have a fixed location. Their location is determined at power-on depending on the system's configuration.

APPENDIX B

B. MEMORY MAP

B.1 System Memory Map

The system maintains ROM and RAM entry point compatibility with the industry standard. Table B.1 provides an outline of the first megabyte of memory.

Table B.1

Memory Map

Description	Starting Address	Absolute Begin	End
Interrupt Vectors	0000:0000H	00000H	003FFH
STD-BIOS Data Area	0040:0000H	00400H	0051DH
Scratch	0050:001EH	0051EH	005FFH
Bios Stack	0060:0000H	00600H	006FFH
DOS	0070:0000H	00700H	
Application	0C00:0050H	0C050H	nF800H
EX-BIOS System RAM		nF800H	nFFFFH
n is dependent upon the amount of memory installed. The EX-BIOS takes a minimum of 800 hex bytes.			
Max RAM Equal 256KB		00000H	3FFFFH
Max RAM Equal 640KB		00000H	9FFFFH
Boot Address	07C0:0000H	07C00H	

Description	Starting Address	Absolute Begin	End
Reserved Video Buffer	A000:0000H	A0000H	
Monochrome Video Buffer	B000:0000H	B0000H	B7FFFH
Color Video Buffer	B800:0000H	B8000H	BFFFFH
Video ROM Space	C000:0000H	C0000H	C7FFFH
IHV ROM	C800:0000H	C8000H	
SPU IHV ROM Space	E000:0000H	E0000H	
BIOS ROM	F000:0000H	F0000H	
BIOS ROM	F800:0000H	F8000H	
RESET Vector	FFFF:0000H	FFFF0H	

B.2 STD-BIOS Data Structures

The data area for the STD-BIOS is in absolute memory locations 00400H through 005FFH, which conforms to the industry standard. Table B.2 summarizes the assignments within this block of memory. A detailed description of these data fields follows the summary.

Table B.2

STD-BIOS Data Area

Address	Function
400H-407H	RS-232 Communication Port Addresses
408H-40FH	Parallel Printer Port Addresses
410H-416H	Equipment Flag
417H-43DH	Keyboard Data Area
43Eh-448H	Flexible Disc Data Area
449H-466H	Video Display Data Area
467H-46BH	Option ROM Data Area
46CH-470H	Timer Data Area
471H-473H	System Data Flags
474H-477H	Hard Disc Data Area

Address	Function
478H-47FH	Printer Timeout Counters
480H-483H	Keyboard Buffer Pointers
484H-488H	Enhanced Graphics Adapter (EGA) Data Area
489H-48AH	Reserved
48BH-48BH	Flexible Disc Data Rate Area
48CH-48FH	Extended Hard Disc Data Area
490H-496H	Extended Flexible Disc Data Area
497H-497H	Keyboard Mode Indicator/LED Data Area
498H-4A0H	Real-Time Clock Data Area
4A1H-4A7H	Reserved
4A8H-4ABH	Pointer to EGA Data Area
4ACH-4EFH	Reserved
4F0H-4FFH	Intra-application Communication Area
500H-500H	Print Screen Status
501H-503H	Reserved
504H-504H	DOS Data Area
505H-5FFH	Reserved

B.2.1 RS-232 Communication Port Addresses

The I/O port addresses of up to four serial communication adapter ports are stored in these four words.

40:000H	02	S40__RS232__PORT1__ADR	Address of serial port 1
40:002H	02	S40__RS232__PORT2__ADR	Address of serial port 2
40:004H	02	S40__RS232__PORT3__ADR	Address of serial port 3
40:006H	02	S40__RS232__PORT4__ADR	Address of serial port 4

B.2.2 Parallel Printer Port Addresses

The I/O port addresses of up to four parallel printer adapter ports are stored in these four words.

40:008H	02	S40__PRINT__PORT1__ADR	Address of parallel port 1
40:00AH	02	S40__PRINT__PORT2__ADR	Address of parallel port 2
40:00CH	02	S40__PRINT__PORT3__ADR	Address of parallel port 3
40:00EH	02	S40__PRINT__PORT4__ADR	Address of parallel port 4

B.2.3 Equipment Byte Data Area

This data area contains several words describing some of the optional hardware installed in the system.

40:010H	02	S40__EQUIPMENT__FLAG	Installed devices word (see table B.3)
40:012H	01	S40__MFG__INIT	Manufacturing initialization/test byte
40:013H	02	S40__MEMORY__SIZE	Memory size in 1k bytes
40:015H	01	S40__MFG__ERR__FLAG1	Manufacturing scratchpad
40:016H	01	S40__MFG__ERR__FLAG2	Manufacturing error codes

Table B.3

Equipment Flag (40:010H)

Bit	Value	Definition
0FH-0EH	0	no printers installed
	1	one printer installed
	2	two printers installed
	3	three printers installed
0DH-0CH	——	reserved
0BH-09H	0	no RS-232 ports installed
	1	one RS-232 port installed
	2	two RS-232 ports installed
	3	three RS-232 ports installed
	4	four RS-232 ports installed
08H	——	reserved
07H-06H	0	1 flexible disc drive installed, if bit 0 = 1
	1	2 flexible disc drives installed, if bit 0 = 1
05H-04H	0	video adapter is not monochrome or color
	1	initial video mode of 40-column color
	2	initial video mode of 80-column color
	3	initial video mode of 80-column monochrome
03H-02H	——	reserved
01H	0	math co-processor not present
	1	math co-processor present
00H	0	no disc drives present
	1	some number of flexible disc drives present, see bits 7-6

B.2.4 Keyboard Data Area

This area is used by the keyboard driver to store keyboard states, scancodes and keycodes.

40:017H	01	S40__KBD__STATE1	State of special keys: shift, caps, etc. (see table B.4).
40:018H	01	S40__KBD__STATE2	Secondary state of special keys (see table B.5).

40:019H	01	S40__ALT__INPUT__ACCUM	Accumulator for alt/numpad entry
40:01AH	02	S40__KBD__BUF__HEAD	Keyboard buffer head pointer
40:01CH	02	S40__KBD__BUF__TAIL	Keyboard buffer tail pointer
40:01EH	20	S40__KBD__BUFFER	Keyboard buffer, room for 15 entries + overrun

Table B.4

Keyboard State Mask Byte1 (40:17H)

Bit	Data	Definition
07H	0	Insert state inactive
	1	Insert state active
06H	0	Caps lock state inactive
	1	Caps lock state active
05H	0	Num lock state inactive
	1	Num lock state active
04H	0	Scroll lock state inactive
	1	Scroll lock state active
03H	0	<Alt> key not depressed (inactive)
	1	<Alt> key depressed (active)
02H	0	<CTRL> key not depressed (inactive)
	1	<CTRL> key depressed (active)
01H	0	Left <Shift> key not depressed (inactive)
	1	Left <Shift> key depressed (active)
00H	0	Right <Shift> key not depressed (inactive)
	1	Right <Shift> key depressed (active)

Table B.5

Keyboard State Mask Byte2 (40:18H)

Bit	Data	Definition
07H	0	<Ins> key not depressed
	1	<Ins> key depressed
06H	0	<Caps lock> key not depressed
	1	<Caps lock> key depressed
05H	0	<Num lock> key not depressed
	1	<Num lock> key depressed
04H	0	<ScrLck> key not depressed
	1	<ScrLck> key depressed
03H	0	Pause state (<CTRL>-<Num lock>) inactive
	1	Pause state active
02H	0	<Sys req> key not depressed
	1	<Sys req> key depressed
01H-00H	——	Reserved

B.2.5 Flexible Disc Data Area

This area is used by the flexible disc driver to store information about current drive activity.

40:03EH	01	S40__FLOPPY__SEEK__STAT	Drive recalibration status (see table B.6)
40:03FH	01	S40__FLOPPY__MOTOR__STAT	Drive motor status (see table B.7)
40:040H	01	S40__FLOPPY__TIME__OUT	Drive timeout counter (see table B.8)
40:041H	01	S40__FLOPPY__RETURN__STAT	Drive return code/error status
40:042H	07	S40__FLOPPY__CONTRL__STAT	Controller status/hard disc command/ parm port copies

Table B.6

Flexible Disc Seek Status Byte (40:03EH)

Bit	Data	Definition
07H	1	Disc hardware interrupt occurred
06H-02H	——	Reserved
01H	0	Indicates drive 1 needs recalibration before next seek
	1	Indicates drive 1 does not need recalibration before next seek
00H	0	Indicates drive 0 needs recalibration before next seek
	1	Indicates drive 0 does not need recalibration before next seek

Table B.7

Flexible Disc Motor Status Byte (40:03FH)

Bit	Data	Definition
07H	0	Current operation is not a write
	1	Current operation is a write
06H	——	Reserved
05H	0	Drive one is not selected
	1	Drive one is selected
04H	0	Drive zero is not selected
	1	Drive zero is selected
03H-02H	——	Reserved
01H	0	Drive one motor is not running
	1	Drive one motor is running
00H	0	Drive zero motor is not running
	1	Drive zero motor is running

Table B.8

Flexible Disc Drive Error Status (40:041H)

Bit	Data	Definition
07H	1	Timeout error; disc failed to respond in time
06H	1	Seek error; seek to track failed
05H	1	Controller error; disc controller chip failed
04H-00H	1	Bad command; invalid command request
	2	Address error; address mark on disc not found
	3	Write protect error
	4	Sector not found; unable to locate sector, disc damaged or unformatted
	6	Media changed; the drive door was opened on a 1.2MB disc drive
	8	DMA error; DMA failed to respond in time
	9	Segment wrap; attempt to perform DMA across a segment boundary
	10H	CRC error; crc check on data failed

B.2.6 Video Display Data Area

This area is used by the video driver to store current screen parameters and cursor positions.

40:049H	01	S40__CRT__MODE	Current video mode
40:04AH	02	S40__CRT__WIDTH	Current # of screen columns
40:04CH	02	S40__CRT__LENGTH	Current length of screen in bytes
40:04EH	02	S40__CRT__PAGE__ADR	Starting address of current display page
40:050H	10	S40__CRT__CURSOR__POS	Cursor coordinates (row, column) up to 8 pages
40:060H	02	S40__CRT__CURSOR__MODE	Current cursor mode setting
40:062H	01	S40__CRT__DISPLAY__PAGE	Current display page

40:063H	02	S40_CRT_PORT_ADR	Base I/O port address for active video controller
40:065H	01	S40_CRT_MODE_SEL_REG	Mode select register copy
40:066H	01	S40_CRT_PALETTE	Color palette register copy

B.2.7 Option ROM Data Area

This area is used by the POST (SYSGEN) routine.

40:067H	02	S40_XROM_INIT_ADR	Offset address for optional I/O rom init routine
40:069H	02	S40_XROM_SEGMENT	Segment address for optional I/O rom
40:06BH	01	S40_XROM_INT_FLAG	Flags last interrupt that occurred

B.2.8 Timer Data Area

This area stores the current timer count and flags.

40:06CH	02	S40_TIMR_LOW	Least significant word of timer count
40:06EH	02	S40_TIMR_HIGH	Most significant word of timer count
40:070H	01	S40_TIMR_OVR_FLOW	24-hour timer tick rollover counter

B.2.9 System Data Flags

This area used by the system to flag <CTRL>-<Break> and <CTRL>-<Alt>-<DEL> requests.

40:071H	01	S40_SYS_BREAK_FLAG	System break request flag
40:072H	02	S40_SYS_RESET_FLAG	System reset flag

B.2.10 Hard Disc Data Area

This area is used by the INT 13H fixed disc driver to store current information about the fixed disc controller and status.

40:074H	01	S40_FD_STATUS	Hard disc status of last Int 13H operation
40:075H	01	S40_FD_COUNT	Number of hard discs present
40:076H	01	S40_FD_CONTROL	Copy of hard disc controller register
40:077H	01	S40_FD_PORT_OFFSET	Hard disc port offset

B.2.11 Printer Timeout Counters

These tables contain timeout counts for the parallel and serial ports. The default value for the parallel printer port is 14H while the serial port is 01H.

40:078H	01	S40_PRINT_TIMEOUT1	Parallel port 1 timeout count
40:079H	01	S40_PRINT_TIMEOUT2	Parallel port 2 timeout count
40:07AH	01	S40_PRINT_TIMEOUT3	Parallel port 3 timeout count
40:07BH	01	S40_PRINT_TIMEOUT4	Parallel port 4 timeout count
40:07CH	01	S40_RS232_TIMEOUT1	Serial port 1 timeout count
40:07DH	01	S40_RS232_TIMEOUT2	Serial port 2 timeout count
40:07EH	01	S40_RS232_TIMEOUT3	Serial port 3 timeout count
40:07FH	01	S40_RS232_TIMEOUT4	Serial port 4 timeout count

B.2.12 Keyboard Buffer Pointers

These pointers indicate where in memory the keyboard buffer is as opposed to the current access points to the buffer stored in the pointers above. This allows an application to move and enlarge the keyboard buffer.

40:080H	02	S40__KBD__BUF__START	Pointer to physical start of keyboard buffer
40:082H	02	S40__KBD__BUF__END	Pointer to physical end of keyboard buffer

B.2.13 Enhanced Graphics Adapter (EGA) Data Area

This data area is used by the optional EGA driver when present.

40:084H	01	S40__EGA__CRT__ROW__CNT	Number of crt rows minus one
40:085H	02	S40__EGA__CHAR__SIZE	Number of bytes per character in font table
40:087H	01	S40__EGA__INFO1	EGA miscellaneous information
40:088H	01	S40__EGA__INFO2	EGA miscellaneous information
40:089H	02		Reserved

B.2.14 Flexible Disc Rate Area

This data area is used by the flexible disc driver to optimize performance on the 1.2mb drives by keeping track of the last data rate selected for disc access.

40:08BH	01	S40__FLOPPY__LAST__RATE	Last data rate selected

B.2.15 Extended Hard Disc Data Area

40:08CH	01	S40_AFD_STATUS_REG	Hard disc status reg. copy
40:08DH	01	S40_AFD_ERROR_REG	Hard disc error reg. copy
40:08EH	01	S40_AFD_INTR_FLAG	Hard disc interrupt flag
40:08FH	01	S40_AFD_CTRL_FLAG	Hard disc controller flag

B.2.16 Extended Flexible Disc Data Area

This data area is used by the flexible disc driver to store information about the current media in the drives and what operations are being performed on it.

40:090H	01	S40_AFLOPPY_MEDIA0	Drive 0 media state (see table B.9)
40:091H	01	S40_AFLOPPY_MEDIA1	Drive 1 media state
40:092H	01	S40_AFLOPPY_OPER0	Drive 0 operation state
40:093H	01	S40_AFLOPPY_OPER1	Drive 1 operation state
40:094H	01	S40_AFLOPPY_TRACK0	Drive 0 current track
40:095H	01	S40_AFLOPPY_TRACK1	Drive 1 current track
40:096H	01	S40_AFLOPPY_RESERVED	Flexible disc reserved byte

Table B.9

Flexible Disc Media Byte (40:090H)

Bit	Data	Definition
07H-06H	0	Data transfer rate is 500kb/sec
	1	Data transfer rate is 300kb/sec
	2	Data transfer rate is 250kb/sec
05H	0	Single step all seeks
	1	Double step all seeks
04H	0	Type of disc in drive unknown
	1	Type of disc in drive known
03H	——	Reserved
02H-00H	0	Attempting 360k disc in 360k drive
	1	Attempting 360k disc in 1.2mb drive
	2	Attempting 1.2mb disc in 1.2mb drive
	3	Determined 360k disc in 360k drive
	4	Determined 360k disc in 1.2mb drive
	5	Determined 1.2mb disc in 1.2mb drive

B.2.17 Keyboard Mode Indicator

This byte is used by the keyboard driver to store the current state of the keyboard LED's.

40:097H	01	S40__KBD__LED__FLAGS	Keyboard LED flags (see table B.10)

Table B.10

Keyboard LED Flag Byte (40:97H)

Bit	Data	Definition
07H-03H	——	Reserved
02H	0	<Caps lock> LED is off
	1	<Caps lock> LED is on
01H	0	<Num lock> LED is off
	1	<Num lock> LED is on
00H	0	<Scroll lock> LED is off
	1	<Scroll lock> LED is on

B.2.18 Real-time Clock Data Area

This area is used by the RTC driver to store information needed to interrupt an application waiting on an RTC event.

40:098H	02	S40__RTC__WAIT__OFFSET	Offset address of user wait flag
40:09AH	02	S40__RTC__WAIT__SEGMENT	Segment address of user wait flag
40:09CH	02	S40__RTC__WAIT__CNT__LOW	Low word of wait count
40:09EH	02	S40__RTC__WAIT__CNT__HIGH	High word of wait count
40:0A0H	01	S40__RTC__WAIT__ACTV__FLG	Wait active flag
40:0A1H	07		Reserved

B.2.19 Pointer to EGA Data Area

40:0A8H	04	S40_EGA_TBL_PTR	Pointer to table of EGA pointers
40:0ACH	2C		Reserved

B.2.20 Intra-application Communications Area

Used by applications to communicate with each other and with themselves from one work session to another.

40:0F0	10	S40_INTRA_APPL	Available to any application

B.2.21 Print Screen Status

40:100H	01	S40_PSCRN_STATUS	Flag for print screen in progress (see table B.11)
40:101H	03		Reserved

Table B.11

Print Screen Status Byte (40:100H)

Bit	Data	Definition
07H-00H	0 1 FFH	Print not in progress Print in progress Error during print

B.2.22 DOS Data Area

The following data areas are used by DOS to provide status information on single-drive systems.

40:104H	01	S40__SINGLE__DRV__STAT	Status of flexible disc for single drive systems, ie currently drive A: or B:
40:105H	1A		Reserved

B.2.23 Reserved Data Areas

The following areas are reserved and should not be used under any circumstances:

40:089H	02
40:0A1H	07
40:0ACH	2C
40:101H	03
40:105H	1A

B.3 EX-BIOS Data Area Map

Figure B.1 shows the EX-BIOS RAM space which contains the HP__VECTOR__TABLE, the EX-BIOS memory pool, and the EX-BIOS global data area.

The following notes correspond to the letters in figure B.1.

a. This address is the segment (CS) value stored in the second word of the HP__ENTRY interrupt vector 06FH, the HP__VECTOR__TABLE is at offset zero. This value may also be obtained from the V__SYSTEM driver, using function F__INS__BASEHPVT.

EX-BIOS Data Area Layout

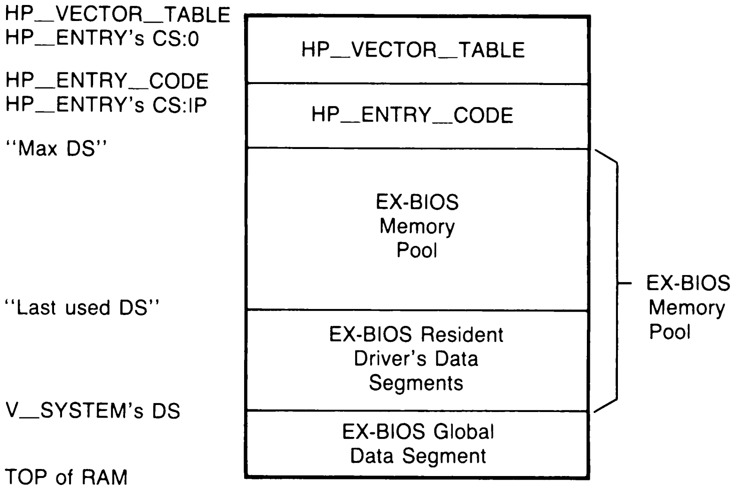


Figure B.1

b. This address is the offset (IP) value stored in the first word of the HP__ENTRY interrupt vector 06FH. This address (CS:IP) represents the end of the HP__VECTOR__TABLE and points to the EX-BIOS's HP__ENTRY__CODE.

c. This address represents the last allocatable data segment ("MAX DS") value available from the EX-BIOS memory pool. This address may be obtained as well as allocated from the EX-BIOS V__SYSTEM driver, see F__RAM__GET and F__RAM__RET in Section 9.

d. This address is passed to drivers requesting memory from the EX-BIOS memory pool. Drivers must first subtract the size of their data segment from the "last used DS" value to get an addressable data area. The new "last used DS" is returned to the EX-BIOS using the F__RAM__RET function.

e. This address represents the EX-BIOS global data area used by drivers and services that share data. This address is the DS value stored in the HP__VECTOR__TABLE for the V__SYSTEM driver.

f. Top of RAM is the last address in memory. In a 256KB system this value is 3FFFFH while in a 640KB system this value is 9FFFFH. The data region between Top of RAM and the base of HP__VECTOR__TABLE is not directly available to applications. In the base system this region is 4KB long. However, different system configurations may require that this region be lengthened.

B.3.1 Option ROM Data Segments

An option ROM which does not have available on board RAM can get memory in the manner described above. However, the problem arises as to how the option ROM is to 'remember' the data segment if it doesn't have any RAM to save the segment in. This problem usually can be solved since most option ROMs are accessed through the software interrupt mechanism. The option ROM adapter simply directs its entry point software interrupt vector to its EX-BIOS RAM data segment which in turn jumps to the option ROM's entry point. That is,

80286 INT nn → EX-BIOS data segment → option ROM

```
    PUSH CS
    POP DS                          ; Load option ROM DS
    JMP FAR ROM__ENTRY__POINT
```

B.3.2 EX-BIOS Global Data Area

The EX-BIOS global data area is shared between several EX-BIOS drivers. It contains temporary and permanent variables required by the EX-BIOS to function properly. Some of these variables can be modified by application programs. As with the STD-BIOS data area, care should be taken when modifying the EX-BIOS data area.

The EX-BIOS global data area can be found by calling the V__SYSTEM driver, with the function F__INS__BASEHPVT. The EX-BIOS global data area segment will be returned in the DS register. Table B.12 defines the contents of this area.

Table B.12

Global Data Area

Byte	Offset	Type	Definition
00-013H	Reserved	Word	
14	T__SND__FLAG	Byte	Sound driver status
	Bit **Definition** 7 '1' Click enabled 6 '1' Beep enabled 5-0 Reserved		
15	T__SND__CLICK__COUNT	Byte	Contains the number of pending key clicks. Maximum of four.
16	T__SND__CLICK__DURA	Byte	Contains the current tick duration scaler.
17	T__SND__CLICK__VOLUME	Byte	Contains the current key click volume.
18	T__SND__BEEP__CYCLE	Word	Contains the current beep period in ten microsecond increments.
1A	T__SND__BEEP__DURA	Word	Contains the current duration of the beep in 10 microsecond increments.
1C	T__SND__BEEP__COUNT	Byte	Contains the number of pending beep functions. Maximum of four.
1D	Reserved		
1E	T__STR__NEXT__INDEX	Word	The next unused string index number.
20 and up	Reserved*		

B.4 ROM BIOS Memory Map

Table B.13 lists the compatible ROM entry points. The programmer is encouraged not to access these entry points directly.

Table B.13

Rom Entry Points

Int	Rom Entry	Type	Function
2	F000:E2C3	code	Nonmaskable interrupt
5	F000:FF54	code	Print screen
10	F000:F065	code	Video
11	F000:F84D	code	Equipment check
12	F000:F841	code	Memory size
13	F000:EC59	code	Diskette/hard disc
14	F000:E739	code	Serial
15	F000:F859	code	System functions
16	F000:E82E	code	Keyboard
17	F000:EFD2	code	Printer
18	F000:4B86	code	Reserved
19	F000:E6F2	code	Boot
1A	F000:FE6E	code	Time and date
1B	F000:FF53	code	Keyboard break
1C	F000:FF53	code	Timer tick
1D	F000:F0A4	data	Video parameter table
1E	0000:0522	data	Diskette parameter table
1F	F000:0000	data	Graphics character table

B.5 Product Identification

Table B.14

Product Identification Strings

ROM version independent information			
0F000:00F8H	DB	‘H’	HP Vectra PC ID
	DB	‘P’	
	DB	00H	
	DB	00H	
ROM version dependent information			
0F000:00FCH	DB	Revision__Code__Secondary	Secondary code revision
	DB	Revision__Code__Primary	Primary code revision
	DB	Date__Code__Year	ROM Release year—1960 stored in BCD
	DB	Date__Code__Week	Week of the year stored in BCD
Industry Standard PC ID			
0F000:FFFEH	DB	0FCH	IBM-AT Compatible PC

APPENDIX C

C. CMOS MEMORY LAYOUT AND REAL-TIME CLOCK

The real-time clock chip contains 64 bytes of non-volatile memory. Values saved in this memory area are not destroyed when the system is powered off. Table C.1 defines the use of the CMOS memory area.

Table C.1

CMOS Memory and Real-time Clock

CMOS Address	Application
00H	*RTC current second
01H	*RTC second alarm value
02H	*RTC current minute
03H	*RTC minute alarm value
04H	*RTC current hour
05H	*RTC hour alarm value
06H	*RTC current day of the week
07H	*RTC current day of the month
08H	*RTC current month
09H	*RTC current year
0AH	*RTC status register A
0BH	*RTC status register B
0CH	*RTC status register C
0DH	*RTC status register D
0EH	*Diagnostic status byte

CMOS Address	Application
0FH	*Shut down status byte
10H	Flexible disc drive type (A and B)
11H	Reserved
12H	Fixed disc type (C and D)
13H	Reserved
14H	Equipment byte
15H	Low base memory
16H	High base memory
17H	Extended memory size (low byte)
18H	Extended memory size (high byte)
19-20H	Reserved
21-27H	*Reserved
28H	*HP checksum for bytes 29, 2A, 2B, 2C
29-2BH	%*Reserved
2CH	%*Reserved
2DH	*Reserved
2E-2FH	*2-byte industry standard CMOS checksum for bytes 10H to 20H
30H	*Extended memory size (low byte, defined by POST)
31H	*Extended memory size (high byte)
32H	*Date century byte
33H	*Information flags
34-3FH	*Reserved

Notes:

*These bytes are not included in the industry standard CMOS checksum

%These bytes are included in HP's checksum

C.1 Real-Time Clock/CMOS Access

Port 70H and port 71H provide the interface to the real-time clock and CMOS memory controller. Port 70H is used to specify the byte address to read or write. Port 71H is used to pass the data. For example, to read the equipment byte, the programmer would write 14H to port 70H, then read the data byte from port 71H. A read or write to port 71H must always be preceeded by a write to port 70H.

C.2 Real-Time Clock (CMOS Address 00H-0DH)

The real-time clock (RTC) chip maintains the current date and time, even when the system is powered off. Four registers are initialized by the SETUP program when the user sets the current date and time. These are detailed in tables C.2, C.3, C.4 and C.5.

Table C.2

CMOS__RTC__REGA (CMOS Address 0AH)

Bit	Data	Definition
7	0	The current date and time is available to read
	1	The current date and time are not available to read because an update of these values is in progress
6-4	——	Time divider selection bits to indicate what time-base frequency is being used. This field is set to 2H to indicate that a 32,768 hertz crystal is providing the time-base.
3-0	——	Rate selection bits to specify output square wave frequency. This field is set to 06H to select a square wave frequency of 1.024K Hertz or a periodic interrupt rate of 976.562 microseconds.

Table C.3

CMOS__RTC__REGB (CMOS Address 0BH)

Bit	Data	Definition
7	0	Update clock normally (default)
	1	Suspend clock updates
6	0	Disable periodic interrupts (default)
	1	Enable periodic interrupts
5	0	Disable alarm interrupts (default)
	1	Enable alarm interrupts
4	0	Do not generate an interrupt when the current update cycle completes (default)
	1	Generate an interrupt each time a clock update completes
3	0	Disable square wave output (default)
	1	Enable square wave output

Bit	Data	Definition
2	0	Store date and time in BCD (Binary Coded Decimal) (default)
	1	Store date and time as binary integers
1	0	Places hours byte in 12 hour mode
	1	Places hours byte in 24 hour mode (default)
0	0	Disable daylight savings (default)
	1	Enable daylight savings

Table C.4

CMOS__RTC__REGC (CMOS Address 0CH)

Bit	Value	Definition
7	0	No interrupts are currently asserted
	1	The RTC is asserting an interrupt due to either the alarm, periodic interrupt, or update ended.
6	0	No periodic interrupt has occurred since the last read of this bit.
	1	A periodic interrupt has occurred, read only and cleared by read.
5	0	No alarm interrupt has occurred since the last read of this bit.
	1	An alarm interrupt has occurred, read only and cleared by read.
4	0	No update ended interrupt has occurred since the last read of the bit.
	1	An update ended interrupt has occurred, read only and cleared by read.
3-0	——	Reserved

Table C.5

CMOS__RTC__REGD (CMOS Address 0DH)

Bit	Value	Definition
7	0	Power was lost to the RTC chip since the last read of this bit.
	1	The RTC chip has not lost power since the last read of this bit. Read only, set to 1 after read.
6-0	——	Reserved

C.3 Diagnostic Status Byte (CMOS Address 0EH)

This byte is set by the POST routine to flag errors detected during power on. The contents of this byte are detailed in table C.6.

Table C.6

CMOS__DIAGNOSTIC__STATUS (CMOS Address 0EH)

Bit	Data	Definition
7	1	Power to RTC failed
6	1	Bad industry standard CMOS checksum
5	1	Configuration inconsistency
4	1	Memory size does not match
3	1	Hard disc failed initialization
2	1	Invalid CMOS
1-0	——	Reserved

C.4 System Shutdown Byte (CMOS Address 0FH)

This byte is used by the system power-on sequence to determine what action is to be taken upon return from protected mode. The details of this byte are shown in table C.7.

Table C.7

CMOS__SHUTDOWN__BYTE (CMOS Address 0FH)

Bit	Value	Definition
7-0	0-3	Perform power-on reset sequence
	4	INT 19H (reboot)
	5	Flush keyboard and jump indirect via double word 40:67H
	6-7	Reserved
	8	Used by POST during test of protected mode RAM
	9	Used for INT 15H support (block move)
	A	Jump indirect via double word at 40:67H
	B-FF	(same as values 0-3)

C.5 Diskette Descriptor Byte (CMOS Address 10H)

This byte is initialized by SETUP and indicates what types of flexible disc drives are installed. The details of this byte are shown in table C.8.

Table C.8

CMOS__FDC__TYPE (CMOS Address 10H)

Bit	Value	Definition
7-4	0	No drive installed as drive A
	1	360KB drive installed as drive A
	2	1.2MB drive installed as drive A
3-0	0	No drive installed as drive B
	1	360KB drive installed as drive B
	2	1.2MB drive installed as drive B

C.6 CMOS Fixed Disc Type (CMOS Address 12H)

CMOS__FIXED__DISC__TYPE, (CMOS Address 12H), is reserved for the hard disc.

C.7 Equipment Byte (CMOS Address 14H)

This byte is used to initialize STD-BIOS RAM location 40:0010H. This is the value returned by the STD-BIOS interrupt INT 11 (get current equipment). The details of this byte are shown in table C.9.

Table C.9

CMOS__EQ__BYTE (CMOS Address 14H)

Bit	Value	Definition
7-6	0	One drive installed
	1	Two drives installed
5-4	1	Primary display is 40 column color
	2	Primary display is 80 column color
	3	Primary display is 80 column monochrome
3-2	——	Reserved
1	1	80287 installed
0	1	At least one flexible disc installed

C.8 System Base RAM Size (CMOS Address 15H—16H)

This value represents the amount of base (DOS addressable) memory installed in the system minus the amount of RAM used by the EX-BIOS data area. Three base memory configurations are valid:

0100H	256K of base memory installed
0200H	512K of base memory installed
0280H	640K of base memory installed

The actual stored value will be adjusted to leave space for the EX-BIOS data area. For example, the value may be 00FCH instead of 0100H, indicating that there is 256K of base RAM installed but the EX-BIOS data area is using 4K of it.

CMOS_BASE_MEMORY_LSB (CMOS Address = 15H)

CMOS_BASE_MEMORY_MSB (CMOS Address = 16H)

C.9 System Extended Memory Size (CMOS Address 17H—18H)

These values are initialized by the SETUP program to the user specified extended memory size from zero to 15Mb in 512Kb increments. For example:

0200	512K of extended memory (0.5Mb)
0400	1024K of extended memory (1.0Mb)
0600	1536K of extended memory (1.5Mb)
	through
3A00	14848K of extended memory (14.5Mb)
3C00	15360K of extended memory (15.0Mb)

Note that extended memory is memory above one megabyte.

CMOS_EXT_MEMORY_LSB (CMOS Address = 17H)

CMOS_EXT_MEMORY_MSB (CMOS Address = 18H)

C.10 EX-BIOS Checksum Byte (CMOS Address 28H)

This byte contains the checksum which is used to verify the contents of the EX-BIOS CMOS data locations. This checksum is computed each time one of these locations is modified using an EX-BIOS CMOS function.

If bit 7 of byte 29 is 1 then

CMOS__EX__BIOS__CRC =
[29] + [2A] + [2B] + [2C] *: 8 bit carryout*

If bit 7 of byte 29 is 0 then

CMOS__EX__BIOS__CRC =
[29] + [2A] + [2B] *: 8 bit carryout*

C.11 EX-BIOS Reserved Bytes (CMOS Address 29H—2CH)

These bytes are reserved by EX-BIOS. They are included in the EX-BIOS checksum byte at CMOS address 28H.

Table C.10

CMOS__HPCONFIG (CMOS Address 29H)

Bit	Data	Definition
7	0	Do not include byte 2C in checksum (default). Note: this bit is not reset during a <CTRL>-<Alt>-<Sys req> reset sequence
	1	Include byte 2C in checksum
6	0	Select the first ROM video adapter as primary (default)
	1	Select the second ROM video adapter as primary
4-1	——	Reserved
0	0	Manufacturing test disabled
	1	Manufacturing test enabled

C.12 STD-BIOS Checksum Word (CMOS Address 2EH-2FH)

This word contains the checksum which is used to verify the contents of the STD-BIOS CMOS data locations. This checksum is computed each time one of these locations is modified using an EX-BIOS CMOS function. If the EX-BIOS is not used for CMOS update then it is the programmer's responsibility to calculate and replace the STD-BIOS checksum.

CMOS__STD__BIOS__CRC =
[10]+[11]+[12]+...+[20] *: 16 bit carryout*

C.13 Test Information Byte (CMOS Address 33H)

Bit seven of this byte is initialized by the boot process to indicate that 640K of base memory is installed. The details of this byte are shown in table C.11.

Table C.11

CMOS__TEST__INFO (CMOS Address 33H)

Bit	Data	Definition
7	1	128kb expansion RAM installed
6-0	——	Reserved

APPENDIX D

D. I/O PORT MAP

Appendix D describes the I/O map of the system. Table D.1 lists the I/O map of all devices integrated in the System Processing Unit (SPU). Table D.2 lists the recommended I/O port assignments for devices in adapter cards. Subsequent sections in the appendix describe the SPU built-in devices individually. I/O devices in adapter cards are described fully in the *Vectra Technical Reference Manual, Volume I.*

Table D.1

SPU I/O Map

I/O Address	Description
000-01FH	First DMA Controller (8237A)
020-03FH	Master Interrupt Controller (8259A)
040-05FH	Timer Controller (8254)
060H	Keyboard Buffer Full port
061H	SPU Control port
064H	Keyboard Output Buffer Full (OBF) port
068H	Keyboard Extended Command port
069H	SVC Service Request read data port
06AH	Keyboard Handshake port
06C-06FH	HP-HIL Controller ports
070H	RTC address / NMI disable port
071H	RTC data
078H	Hard Reset NMI enable port
07CH	HP-Slave Interrupt Controller (8259A) port 0
07DH	HP-Slave Interrupt Controller (8259A) port 1
080-09FH	DMA Page Registers ports
0A0-0BFH	Industry Standard (STD) Slave Interrupt Controller (8259A)
0C0-0DFH	Second DMA Controller (8237A)
0F0H	Clear 80287 Coprocessor port
0F1H	Reset " " "
0F8-0FFH	80287 Math Coprocessor

Table D.2

Adapter I/O Map

I/O Address	Description
1F0-1F3H	Hard Disc controller
200-207H	Game I/O adapter
278-27FH	Parallel port 2
2E8-2EFH	Serial port 3
2F8-2FFH	Serial port 2
300-307H	Prototype adapter card
320-323H	Reserved
378-37FH	Parallel port 1
380-38FH	SDLC, bisynch 2
3A0-3AFH	Bisynch 1
3B0-3B7H	Monochrome display adapter
3BC-3BFH	Monochrome display/Parallel adapter
3D0-3DFH	Color Graphics adapter
3E8-3EFH	Serial port 4
3F0-3F7H	Flexible Disc controller
3F8-3FFH	Serial port 1

D.1 DMA Channel Controller

The SPU supports seven DMA channels using two Intel 8237A DMA controllers in cascade mode. For each DMA channel there is a page register used to extend the addressing range of the channel to 16 MB. The only type of DMA transfer allowed is "I/O to memory". No "I/O to I/O" or "memory to memory" transfers are allowed due to the way the hardware is configured. For more detailed information on the 8237A DMA controllers see Intel's *The 8086 Family User's Manual*.

Table D.3 summarizes how the DMA channels are allocated.

Table D.3

DMA Channel Allocation

First DMA controller (used for 8 bit transfers):	
channel 0	—Spare.
channel 1	—Usually datacomm.
channel 2	—Flexible disc I/O.
channel 3	—Spare.

Second DMA controller (used for 16 bit transfers):	
channel 4	—Cascade from first DMA controller.
channel 5	—Spare.
channel 6	—Spare.
channel 7	—Spare.

D.1.1 I/O Port Addresses for DMA Controllers

Table D.4 shows the I/O port addresses for the page register and DMA controllers used when writing the DMA addresses.

Table D.4

I/O Port Addresses and Address Lines

DMA page register I/O Ports

Channel	I/O port	Address Lines	
0	087H	A23-A16	
1	083H	A23-A16	byte transfers
2	081H	A23-A16	
3	082H	A23-A16	
4	Not connected		
5	08BH	A23-A17	
6	089H	A23-A17	word transfers
7	08AH	A23-A17	
X	08FH	Used for RAM refresh	

DMA Controller I/O Ports

Channel	I/O port		
0	000H	address register (A15-A0)	byte transfers
	001H	byte count register	
1	002H	address register (A15-A0)	
	003H	byte count register	
2	004H	address register (A15-A0)	
	005H	byte count register	
3	006H	address register (A15-A0)	
	007H	byte count register	
4	0C0H	address register (A16-A1)	word transfer
	0C2H	word count register	
5	0C4H	address register (A16-A1)	
	0C6H	word count register	
6	0C8H	address register (A16-A1)	
	0CAH	word count register	
7	0CCH	address register (A16-A1)	
	0CEH	word count register	

Notes:

Channel 4 (first channel on the second DMA controller) is used to cascade the first DMA controller and it must not be programmed for DMA transfers.

Channels 5 thru 7 are word-wide channels so the address lines used are A1 thru A23. Address line A0 is always forced to zero. The address register on these channels provides address lines A16 thru A1 and address lines A23 through A17 come from bits 7 through 1 of the page register. Bit 0 of the page register is not used. Care should be taken in making sure that the counts and addresses are in words rather than bytes.

Table D.5 lists I/O ports used for writing commands to the DMA controllers.

Table D.5

Controller Command I/O Ports

Controller			
1	**2**	**I/O Write**	**I/O Read**
0D0H	008H	Command Register	Status Register
0D2H	009H	Request Register	illegal
0D4H	00AH	Single Mask Register	illegal
0D6H	00BH	Mode Register	illegal
0D8H	00CH	Clear Byte Pointer Flag	illegal
0DAH	00DH	Master Clear Command	Temporary Register
0DCH	00EH	Clear Mask Command	illegal
0DEH	00FH	Multi-Mask Register	illegal

D.2 8259A Interrupt Controllers

The system has three 8259A interrupt controllers. They are arranged as a master interrupt controller and two slaves that are cascaded through the master. Table D.6 shows the I/O ports for these interrupt controllers and how they are cascaded.

Table D.6

8259A Controller I/O Ports

Register Name	Master	HP-Slave	STD-Slave
Command Register	20H	7CH	0A0H
Interrupt Mask Register	21H	7DH	0A1H

Table D.7 shows how the master and slave controllers are connected. The Interrupt Requests (IRQ) are numbered sequentially starting with the Master 8259 controller and followed by the industry standard (IS) Slave and HP-Slave.

Table D.7

8259A Master to Slave Connections.

Master's IRQ		Interrupt Request Description
IRQ0(08H)		Timer
IRQ1(09H)	<—[HP-Slave IRQ]	Keyboard
	IRQ16(68H)	8041
	IRQ17(69H)	Keyboard OBF
	IRQ18(6AH)	Reserved
	IRQ19(6BH)	Reserved
	IRQ20(6CH)	HP-HIL Controller
	IRQ21(6DH)	Reserved
	IRQ22(6EH)	Reserved
	IRQ23(6FH)	Reserved
IRQ2(0AH)	<—[STD-Slave IRQ]	Reserved
	IRQ08(70H)	Real Time Clock
	IRQ09(71H)	SW Redirected
	IRQ10(72H)	Reserved
	IRQ11(73H)	Reserved
	IRQ12(74H)	Reserved
	IRQ13(75H)	80287 Coprocessor
	IRQ14(76H)	Hard Disc
	IRQ15(77H)	Reserved

Master's IRQ	Interrupt Request Description
IRQ3(0BH)	Serial Port 1
IRQ4(0CH)	Serial Port 0
IRQ5(0DH)	Printer Port 1
IRQ6(0EH)	Diskette
IRQ7(0FH)	Printer Port 0

Note: The numbers in parentheses are the interrupt vector numbers generated by the IRQs.

The following example shows how the 8259A controllers are programmed on initialization:

```
                CLI                                     ; Disable interrupts
PROGRAM__MASTER:
                MOV     AL, 11H                         ; ICW1: Initialization Command
                OUT     20H,AL
                JMP     $+2
                MOV     AL,08H                          ; Interrupt Vector Base at 08H
                OUT     21H,AL
                JMP     $+2
                MOV     AL,06H                          ; Define master with two slaves
                OUT     21H,AL                          ; one at IRQ1 and one at IRQ2
                JMP     $+2
                MOV     AL,01H                          ; 8086/88 Mode
                OUT     21H,AL
                JMP     $+2
PROGRAM__HP__SLAVE:
                MOV     AL, 11H                         ; ICW1: Initialization Command
                OUT     7CH,AL
                JMP     $+2
                MOV     AL,68H                          ; Interrupt Vector Base at 68H
                OUT     7DH,AL
                JMP     $+2
                MOV     AL,01H                          ; Slave ID number
                OUT     7DH,AL
                JMP     $+2
                MOV     AL,01H                          ; 8086/88 Mode
                OUT     7DH,AL
                JMP     $+2
                MOV     AL,68H                          ; Place HP slave on special
                OUT     7CH                             ; mask mode.
                JMP     $+2
PROGRAM__STD__SLAVE:
                MOV     AL, 11H                         ; ICW1: Initialization Command
```

```
        OUT     0A0H,AL
        JMP     $+2
        MOV     AL,70H                          ; Interrupt Vector Base at 70H
        OUT     0A1H,AL
        JMP     $+2
        MOV     AL,02H                          ; Slave ID number
        OUT     0A1H,AL
        JMP     $+2
        MOV     AL,01H                          ; 8086/88 Mode
        OUT     0A1H,AL
        JMP     $+2
        STI                                     ; Reenable interrupts
```

The following example shows how an interrupt generated from the HP-Slave is serviced. This provides an example of what commands to send the 8259 controllers to handle an IRQ request. See Section 4 for more details.

```
;
; Interrupt handler example for handling an IRQ16 which is an 8041
; keyboard controller service request:
;
INTERRUPT__HANDLER:
        PUSH    AX                              ; Save registers
        IN      AL,7DH                          ; Get enable mask from HP-Slave
        JMP     $+2
        OR      AL,01H                          ; disable IRQ16 interrupt
        OUT     7DH,AL
        JMP     $+2
        MOV     AL,20H                          ; Send an EOI to master 8259
        OUT     20H,AL                          ; so that other interrupts can
        JMP     $+2                             ; get thru
        .
        .                                       ; 8041 Service processing here
        .
        IN      AL,7DH                          ; Get enable mask from HP-Slave
        JMP     $+2
        AND     AL,0FEH                         ; Enable IRQ16 again
        OUT     7DH,AL
        JMP     $+2
        MOV     AL,60H                          ; Send the HP-Slave a specific
        OUT     7CH,AL                          ; EOI command.
        JMP     $+2
        POP     AX                              ; Restore registers
        IRET
```

D.3 8254 Timer Controller (I/O Ports 40H through 43H)

The system contains an Intel Programmable Interval Timer 8254. The timer controller consists of three separate timer channels; timer channels 0, 1 and 2. Channel 0 provides the BIOS with a programmable time interval. Channel 1 provides the memory refresh signal of the dynamic RAMs in the system. Channel 2 generates a fixed frequency envelope to the sound generation circuit.

WARNING!

> Timer channel 1 should not be used. Writing to this channel may cause loss of data in system memory.

The timer chip interfaces to the 80286 via 4 I/O ports:

I/O Port	Function
040H	Counter data for timer 0.
041H	Counter data for timer 1.
042H	Counter data for timer 2.
043H	The control register for all three timers.

See Intel's *8086 Family User's Manual* for more details of the 8254 timer controller.

D.4 Keyboard Data Buffer (60H)

The keyboard data buffer is read by the 80286 when the keyboard asserts the OBF interrupt. The OBF signal is automatically cleared when the data buffer is read. See Section 5 for more information about the keyboard data buffer.

D.5 SPU Control Port (61H)

The SPU Control Port (61H) is a bidirectional buffer which latches an assortment of unrelated signals. Table D.8 describes the bit values contained in this buffer.

Table D.8

PORT 61H Bit Values

When 80286 reads port 61H:

Bit	Data	Definition
7	1	Parity error in on-board system ram
6	1	I/O channel check error has occurred
5		Output from timer channel 2
4		Refresh detect; toggles once per refresh cycle
3		Status of I/O channel check NMI latch (See Fig D.2)
	1	Disabled.
	0	Enabled
2		Status of SPU RAM parity error latch (See Fig D.2)
	1	Disabled
	0	Enabled
1	1	Speaker data from timer channel 2 is enabled to drive speaker circuit.
0	1	Gate to timer channel 2 is enabled

When 80286 writes port 61H:

Bit	Data	Description
7-4		Not used
3	1	Disable and clear I/O channel check.
2	1	Disable and clear on-board parity NMI
1	1	Enable the data from timer channel 2 to drive speaker circuit.
0	1	Enable gate to timer channel 2. (speaker data)

D.6 Speaker Control

Figure D.1 shows the relationship of the timer channel 2 and the rest of the speaker circuit.

Speaker Control Circuit

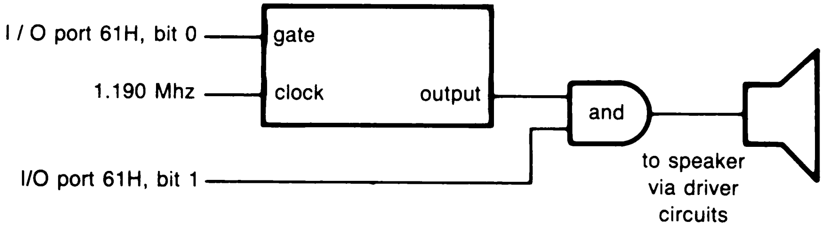


Figure D.1

The sound related EX-BIOS functions are the recommended method of accessing the speaker functions (see Section 9).

D.7 Keyboard I/O Ports

Keyboard Command Port (64H): Used to write commands to the 8041 keyboard controller.

Keyboard Extended Command Port (68H): provides a data/command path to the 8041 not conflicting with the industry standard I/O Ports 60H and 64H.

KBD Request Port (69H): Allows communications between the 8041 and the EX-BIOS service request (SVC) routines.

Keyboard Handshake (6AH): The single bit write only port provides a mechanism for the 8041 keyboard controller to indicate the last service request (SVC) has been handled.

HP-HIL Controller (6CH thru 6FH): This set of I/O Ports provides the communication mechanism to the HP-HIL controller.

D.8 Real Time Clock Ports

Real Time Clock and CMOS RAM ports 70H and 71H provide the interface to the MC146818 real time clock (RTC). See Appendix C for further details.

D.9 Hard Reset Enable Port

Hard Reset Enable Register (Port 78H): This write only port enables the hard reset line from the HP-HIL controller. Table D.9 shows the bit definitions for this port.

Table D.9

Hard Reset Enable Register

Bit	Data	Description
7	1	Enable hard reset from HP-HIL controller chip.
	0	Disable and clear hard reset from HP-HIL controller chip.
6-0		Reserved.

D.10 NMI Sources and Involved I/O Ports

The non-maskable interrupt (NMI) of the 80286 is attached to circuitry which allows multiple sources to cause an NMI. Each of these sources can be disabled individually as well as collectively.

Figure D.2 is a block diagram of the NMI circuit.

NMI Circuit

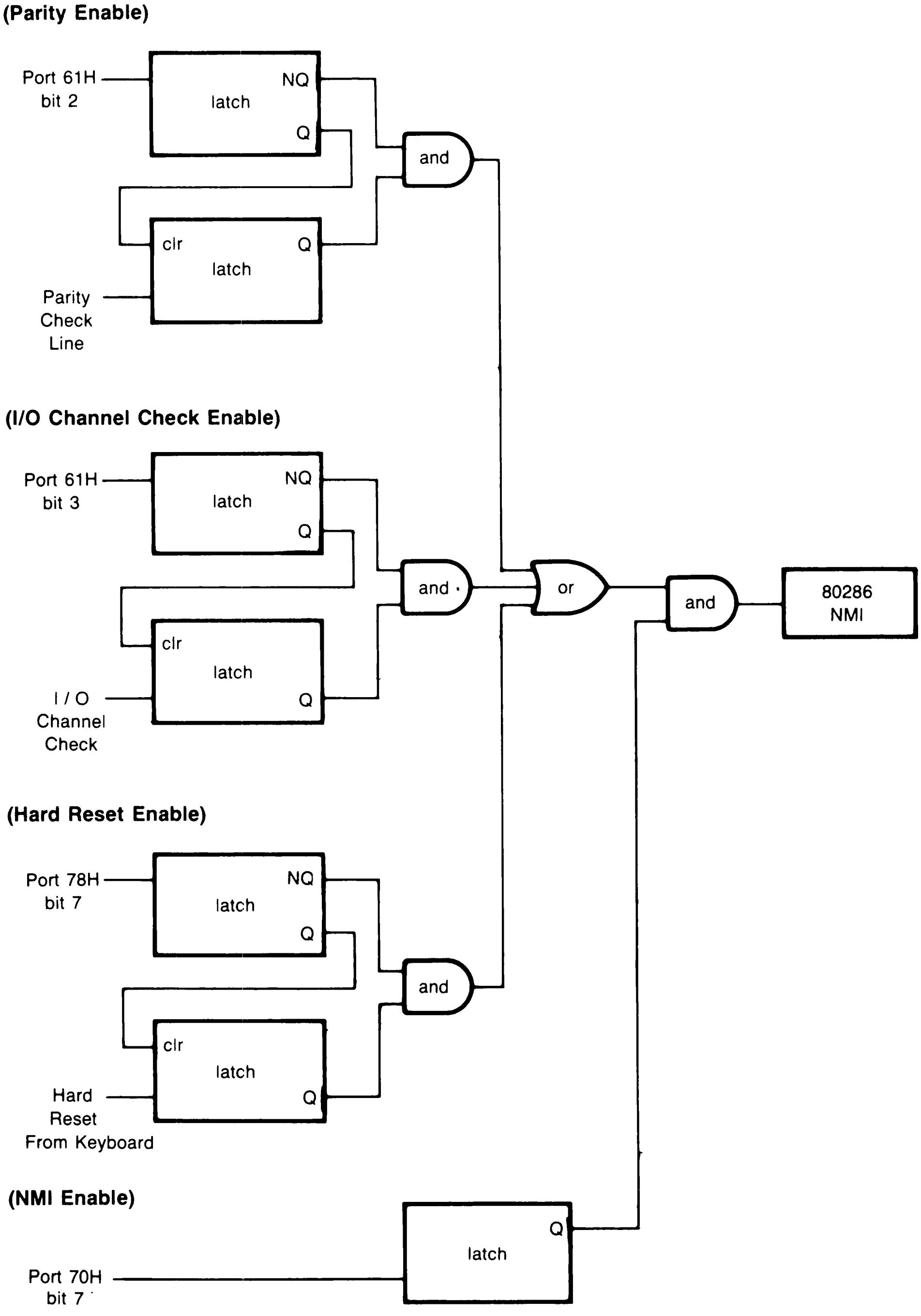


Figure D.2

APPENDIX E

E. SYSTEM EQUATE FILE

This appendix contains the Macro Assembler (MASM) listing of the system equate file, EQUATES.ASM.

Equates are assembly language (MASM) directives. The term equate as used here includes the MASM directives: EQU, ' = ', STRUC, RECORD, and MACRO. They allow the programmer to assign ASCII strings (names) to numeric constants, data structures, data records and macros. The name can then be used in programs to define data structures, code structures, or record structures. When the program is assembled, MASM substitutes the value associated with the name for every occurance of the name in the source code.

The MASM directive 'INCLUDE' is used by programs to define constants, data structures or code structures commonly used by different programs. When a particular equate or group of equates is needed in a program, the programmer does not have to either define a new equate name for the variable or type it into the program. The programmer can use the 'INCLUDE' statement to define the equate. At assembly time the INCLUDE directive causes the assembler to read a specified file and process it as if its contents were actually in the orignal source code file. See the HP Vectra MS-DOS Macro Assembler manual for more information on include files.

E.1 The Equate File

The equate file supports programmer's access to both the STD-BIOS and EX-BIOS. Support is provided for software interrupt numbers, interrupt function and subfunction codes, and data structures associated with the various functions. Commonly used MACROS are also defined.

Equates File

```
                        page    59,132
                        .286c
                                        NAME     EQUATES
                        ;************************************************************************
                        ; This file contains equates, data structures and definitions needed to
                        ; access both the Standard BIOS (STD-BIOS) and the Extended BIOS (EX-BIOS)
                        ; of Vectra using MASM 3.0. Depending on what part of the BIOS you are
                        ; accessing you might not need all the equates.  The equates are organized
                        ; as follows:
                        ;
                        ;   o 80286 Support Macros.
                        ;   o EX-BIOS Equates.
                        ;       1) Generic Structures and equates used by all drivers.
                        ;       2) Equates for Vector Addresses.
                        ;       3) Function and Subfunction Equates common to all drivers
                        ;       4) Function and Subfunction Equates individual to drivers. These
                        ;          are ordered by vector number
                        ;   o MS-DOS Macros and Equates.
                        ;   o Industry Standard (STD-BIOS) Interrupt numbers and function equates.
                        ;   o Industry Standard (STD-BIOS) Data Area
                        ;   o Bit definitions for Industry Standard (STD-BIOS) data area entries
                        ;
                        ; The programmer can extract only those equates that he needs to create
                        ; a tailored equate file.
                        ;************************************************************************
                        page

                        ;************************************************************************
                        ; 80286 Support macros and equates
                        ;************************************************************************
                        ;************************************************************************
                        ; The following macro is used to compensate for a bug in the 80286
                        ; hardware interrupt system.  During a normal POPF instruction
                        ; cycle interrupts are always enabled regardless of the state of
                        ; the interrupt enable flag prior to the pop or after the pop.
                        ;************************************************************************
                        POPPF       macro
                                    jmp    $+3
                                    iret
                                    push  cs
                                    call  $-2
                                    endm

                        ;************************************************************************
                        ; EX-BIOS support macros and equates.
                        ;************************************************************************

                        ; Equates for EX-BIOS interrupt number and vector address.
 = 006F                 HP_ENTRY                equ      6FH

                        ;************************************************************************
                        ;                       SYSCALL [vector address]
                        ;************************************************************************
                        syscall                 macro    vector
                         ifnb                    <vector>
                                                mov      bp,vector
                         endif
                                                int      HP_ENTRY
                                                endm

                        ;************************************************************************
                        ; HP_VECTOR_TABLE Entry
                        ;************************************************************************
                        HP_TABLE_ENTRY          struc   ;<1,2,3>
 0000  0000             HP_ENTRY_IP             dw      0
 0002  0000             HP_ENTRY_CS             dw      0
 0004  0000             HP_ENTRY_DS             dw      0
 0006                   HP_TABLE_ENTRY          ends
                        page
                        ;********************************************************************
                        ; Structure of Data Header for HP's vectors.
                        ;********************************************************************
                        HP_SHEADER              STRUC   ;<1,2,3,4,5,6,7,8,9,0>
 0000  0000             DH_ATR                  dw  0    ; Attribute
 0002  0000             DH_NAME_INDEX           dw  0    ; Name index of driver.
 0004  0000             DH_V_DEFAULT            dw  0    ; Driver vector position in HPtable
 0006  0000             DH_P_CLASS              dw  0    ; Parent class
 0008  0000             DH_C_CLASS              dw  0    ; Child class
 000A  0000             DH_V_PARENT             dw  0    ; Vector used when the driver cannot handle
                                                         ; an F_ISR function call
 000C  0000             DH_V_CHILD              dw  0    ; Vector used when the driver cannot handle
                                                         ; a regular function call
 000E  00               DH_MAJOR                db  0    ; Driver's major address if any.
 000F  00               DH_MINOR                db  0    ; Driver's minor address if any.
 0010                   HP_SHEADER              ENDS

                        ;********************************************************************
                        ; DH_ATR bit record
                        ;********************************************************************
 = 8000                 ATR_HP            equ  1000000000000000B ; 1- The Rest of header is valid
 = 4000                 ATR_DEVCFG        equ  0100000000000000B ; 1- Present in DEVCONFG
 = 2000                 ATR_ISR           equ  0010000000000000B ; 1- Replace My ISR   (Child)
```

Equates File (continued)

```
= 1000                 ATR_ENTRY          equ  0001000000000000B ; 1- Replace My ENTRY (Parent)
= 0000                 ATR_RSVD           equ  0000000000000000B ; 0 - Available on allocation from HP
= 0200                 ATR_FREE           equ  0000001000000000B ; 1 - Available Vector
= 0400                 ATR_SRVC           equ  0000010000000000B ; 2 - Service Vector
= 0600                 ATR_LOG            equ  0000011000000000B ; 3 - Logical Device Start DEVCONFG CHA
= 0800                 ATR_IND            equ  0000100000000000B ; 4 - Filter, show driver (options)
= 0A00                 ATR_BOT            equ  0000101000000000B ; 5 - End of Chain
= 0C00                 ATR_INP            equ  0000110000000000B ; 6 - Mappable input driver
= 0E00                 ATR_TYPE7          equ  0000111000000000B ; 7 - Available
= 0E00                 ATR_TYPE_MASK      equ  0000111000000000B
= 0100                 ATR_STRING         equ  0000000100000000B
= 0080                 ATR_MAP_CALL       equ  0000000010000000B
= 0060                 ATR_SUBADD         equ  0000000001100000B
= 0000                 ATR_NOADDR         equ  0000000000000000B
= 0020                 ATR_MAJOR          equ  0000000000100000B
= 0040                 ATR_MINOR          equ  0000000001000000B
= 0060                 ATR_MID            equ  0000000001100000B
= 0010                 ATR_PSHARE         equ  0000000000010000B
= 0008                 ATR_CSHARE         equ  0000000000001000B
= 0004                 ATR_ROM            equ  0000000000000100B
= 0002                 ATR_YIELD          equ  0000000000000010B
= 0001                 ATR_0              equ  0000000000000001B
                       page
                       ;*****************************************************************
                       ;   DH Class
                       ;*****************************************************************
= 8000                 CL_KBDFC            equ  1000000000000000B      ; 1 - HP Softkey Transaltor
= 4000                 CL_KBD              equ  0100000000000000B      ; 1 - Keyboard
= 2000                 CL_CCP              equ  0010000000000000B      ; 1 - Cursor Control pad
= 1000                 CL_CON              equ  0001000000000000B      ; 1 - Console Device
= 0800                 CL_BYTE             equ  0000100000000000B      ; 1 - PRN device
= 0400                 CL_COMM             equ  0000010000000000B      ; 1 - COMM device
= 0200                 CL_INTERFACE        equ  0000001000000000B      ; 1 - interface, HP-HIL, HPIB
= 0100                 CL_FILT             equ  0000000100000000B      ; 1 - charachter filter
= 0080                 CL_BLK              equ  0000000010000000B      ; 1 - block device
= 0040                 CL_BOOT             equ  0000000001000000B      ; 1 - boot block device
= 0020                 CL_LGID             equ  0000000000100000B      ; 1 - logical physical gid
                                                                       ; e g  ccp to gid translator
= 0010                 CL_PGID             equ  0000000000010000B      ; 1 - physical gid
= 0008                 CL_GID              equ  0000000000001000B      ; 1 - any graphics input device
= 0004                 CL_PTS              equ  0000000000000100B      ; 1 - physical touch screen
= 0002                 CL_ASCII            equ  0000000000000010B      ; 1 - ascii input device
= 0001                 CL_EXTEND           equ  0000000000000001B      ; 0 - set of classes above
                                                                       ; 1 - alternate class set
= FFFF                 CL_ALL              equ  1111111111111111B      ; Member of all classes
= 0000                 CL_NULL             equ  0000000000000000B      ; Member No Classes

                       ;*********************************************************************
                       ; Vector Addresses
                       ;*********************************************************************
= 0000                 V_SCOPY                   equ      0000H        ; Copyright Notice
= 0006                 V_DOLITTLE                equ      0006H        ; Nop Routine
= 000C                 V_PNULL                   equ      000CH        ; No Device
= 0012                 V_SYSTEM                  equ      0012H        ; System Intrinsics
= 001E                 V_S8259                   equ      001EH        ; 8259 Interface
= 002A                 V_SINPUT                  equ      002AH        ; Input Inquire routines
= 0036                 V_QWERTY                  equ      0036H        ; Qwerty KBD Translator
= 003C                 V_SOFTKEY                 equ      003CH        ; HP f1-f8 Translator
= 0042                 V_FUNCTION                equ      0042H        ; IBM F1-F10 Translator
= 0048                 V_NUMPAD                  equ      0048H        ; Numeric Pad Translator
= 004E                 V_CCP                     equ      004EH        ; CCP Translator Driver
= 0054                 V_SVIDEO                  equ      0054H        ; Video Intrinsics
= 005A                 V_STRACK                  equ      005AH        ; Common cursor control funcs.
= 0060                 V_EVENT_TOUCH             equ      0060H        ; Touch Event Intercept
= 0066                 V_EVENT_TABLET            equ      0066H        ; Tablet Event Intercept
= 006C                 V_EVENT_POINTER           equ      006CH        ; Pointer Event Intercept
= 008A                 V_CCPCUR                  equ      008AH        ; CCP to Cursor Always Filter (Default)
= 0090                 V_RAW                     equ      0090H        ; CCP+Softkey RAW Mode Filter
= 0096                 V_CCPNUM                  equ      0096H        ; CCP to Numeric Pad Filter
= 009C                 V_OFF                     equ      009CH        ; CCP+Softkey Off Filter
= 00A2                 V_CCPGID                  equ      00A2H        ; CCP to GID Filter ( Not Implemented)
= 00A8                 V_SKEY2FKEY               equ      00A8H        ; Softkey (f1-f8) to Function
                                                                       ;  key (F1-F8) Filter (Default)
= 00AE                 V_8041                    equ      00AEH        ; 8041 Interface
= 00B4                 V_PGID_CCP                equ      00B4H        ; Graphic to CCP Filter
= 00BA                 V_LTABLET                 equ      00BAH        ; Tablet driver
= 00C0                 V_LPOINTER                equ      00C0H        ; Pointer driver
= 00C6                 V_LTOUCH                  equ      00C6H        ; Touch driver
= 00CC                 V_LHPMOUSE                equ      00CCH        ; Microsoft/Mouse System's
                                                                       ;   Compatible Driver
= 0108                 V_LNULL                   equ      0108H        ; No Driver
= 0114                 V_HPHIL                   equ      0114H        ; HP-HIL Driver

                       ;****************************************************************
                       ; Common Function Codes for HP Routines
                       ;****************************************************************
= 0000                 F_ISR                     equ      00H*2        ; Interrupt service call
= 0002                 F_SYSTEM                  equ      01H*2        ; System func. call,
                                                                       ;   Subfunction required
= 0004                 F_IO_CONTROL              equ      02H*2        ; Device/Driver Dependent
                                                                       ;   Functions
= 0006                 F_PUT_BYTE                equ      03H*2        ; Write one byte of data:
                                                                       ;   Byte is in AL
= 0008                 F_GET_BYTE                equ      04H*2        ; Read a byte of data:
                                                                       ;   Byte returned in AL
```

Equates File (continued)

```
= 000A        F_PUT_BUFFER          equu   05H*2            ; Write a buffer of data,
                                                           ;    ES,DI pointer, CX count
= 000C        F_GET_BUFFER          equ    06H*2            ; Read a buffer if data,
                                                           ;    ES,DI pointer, CX count
= 000E        F_PUT_WORD            equ    07H*2            ; Write word of data, DX data
= 0010        F_GET_WORD            equ    08H*2            ; Read word of data, DX data
=             F_PUT_BLOCK           equ    F_PUT_BUFFER     ;used for disk applications
=             F_GET_BLOCK           equ    F_GET_BUFFER

              ;*****************************************************************
              ; Common Subfunction codes of the F_SYSTEM Function.
              ;*****************************************************************
= 0000        SF_INIT               equ    00H*2   ;Initialize command
= 0002        SF_START              equ    01H*2   ;Secondary Init--initialize dependent
                                                   ; upon other drivers/data structures
= 0004        SF_REPORT_STATE       equ    02H*2   ;Reports state of driver
= 0006        SF_VERSION_DESC       equ    03H*2   ;Report version and option describe
                                                   ;   record
= 0008        SF_DEF_ATTR           equ    04H*2   ;Reports default Configuration
                                                   ;    (Baud Rate)
= 000A        SF_GET_ATTR           equ    05H*2   ;Reports Current Configuration
                                                   ;    ES,DI pointer
= 000C        SF_SET_ATTR           equ    06H*2   ;Sets Next Configuration ES,DI, CX
= 000E        SF_OPEN               equ    07H*2   ;Reserve Driver for exclusive access
= 0010        SF_CLOSE              equ    08H*2   ;Release  "   from      "       "
= 0012        SF_TIMEOUT            equ    09H*2   ;Notify Driver Timeout Occurred
= 0014        SF_INTERVAL           equ    0AH*2   ;Notify Driver Interval Occurred
= 0016        SF_TEST               equ    0BH*2   ;Test Function

              ;*****************************************************************
              . Common Subfunction Codes for the F_IO_CONTROL function.
              ;*****************************************************************
= 0000        SF_LOCK               equ    00H*2   ; Lock Device for exclusive access
= 0002        SF_UNLOCK             equ    01H*2   ; Unlock Device for exclusivce access
              page
              ;*****************************************************************
              ; HP Routines Return Status. Sucessful codes are positive and failure
              ; are negative.
              ;*****************************************************************

= 000C        RS_BREAK        equ     00CH             Break -- IFC
= 000A        RS_DATA_NREADY  equ     00AH           ; RS232 Data Not Ready=>Retry Operation
= 0008        RS_OVERRUN      equ     008H           ; RS232 Port Data Overrun =>Retry Operation
= 0006        RS_DONE         equ     006H           ; indicates all done return child
= 0004        RS_NOT_SERVICED equ     004H           ; indicates a chained ISR--not handled
= 0002        RS_UNSUPPORTED  equ     002H           ; indicates function is NOPed/not valid
                                                     ; for this driver
= 0000        RS_SUCCESSFUL   equ     000H           ; indicates executed just fine

              ;*******************************************************************

= 00FE        RS_FAIL         equ     0FEH           ; To be used with hardware failure
= 00FC        RS_TIMEOUT      equ     0FCH           ; to indicate remote device timeout
= 00FA        RS_BAD_PARAMETER equ    0FAH           ; to indicate a bad parameter
= 00F8        RS_BUSY         equ     0F8H           ; to indicate driver/device is busy
= 00F6        RS_NO_VECTOR    equ     0F6H           ; out of hp_VT vectors
= 00F4        RS_OFFLINE      equ     0F4H           ; device is offline
= 00F2        RS_OUT_OF_PAPER equ     0F2H           ; out of paper on printer device
= 00F0        RS_PARITY       equ     0F0H           ; parity error in transmission
= 00EE        RS_FRAME        equ     0EEH           ; framing error
              page
              ;***********************************************************************
              ; Function Number Equates for the EX-BIOS routines and its Data Structures.
              ;***********************************************************************

              ;*****************************************************************
              ; V_SYSTEM (0012H) function codes.
              ;*****************************************************************
              ; F_ISR and F_SYSTEM are functions common to all drivers.

= 0004        F_INS_BASEHPVT         equ     0004H
= 0006        F_INS_XCHGFIX          equ     0006H
= 0008        F_INS_XCHGRSVD         equ     0008H
= 000A        F_INS_XCHGFREE         equ     000AH
= 000C        F_INS_FIXOWNDS         equ     000CH
= 000E        F_INS_FIXGETDS         equ     000EH
= 0010        F_INS_FIXGLBDS         equ     0010H
= 0012        F_INS_FREEOWNDS        equ     0012H
= 0014        F_INS_FREEGETDS        equ     0014H
= 0016        F_INS_FREEGLBDS        equ     0016H
= 0018        F_INS_FIND             equ     0018H
= 001E        F_RAM_GET              equ     001EH
= 0020        F_RAM_RET              equ     0020H
= 0022        F_CMOS_GET             equ     0022H
= 0024        F_CMOS_RET             equ     0024H
= 002A        F_YIELD                equ     002AH
= 0030        F_SND_CLICK_ENABLE     equ     0030H
= 0032        F_SND_CLICK_DISABLE    equ     0032H
= 0034        F_SND_CLICK            equ     0034H
= 0036        F_SND_BEEP_ENABLE      equ     0036H
= 0038        F_SND_BEEP_DISABLE     equ     0038H
= 003A        F_SND_BEEP             equ     003AH
= 003C        F_SND_SET_BEEP         equ     003CH
= 003E        F_SND_TONE             equ     003EH
```

Equates File (continued)

```
      = 0040                      F_STR_GET_FREE_INDEX     equ      0040H
      = 0042                      F_STR_DEL_BUCKET         equ      0042H
      = 0044                      F_STR_PUT_BUCKET         equ      0044H
      = 0046                      F_STR_GET_STRING         equ      0046H
      = 0048                      F_STR_GET_INDEX          equ      0048H
                                  page
                                  ;*****************************************************************
                                  ;  String Bucket Header  This structure is useful if using the
                                  ; following V_SYSTEM functions  F_STR_DEL_BUCKET and F_STR_PUT_BUCKET
                                  ;*****************************************************************
                                  STR_HEADER               STRUC
      0000  ????????              STR_NXT_HDR              dd       (?)
      0004  ????                  STR_UPPER_BOUND          dw       (?)
      0006  ????                  STR_LOWER_BOUND          dw       (?)
      0008  ????????              STR_LIST_PTR             dd       (?)
      000C  ????                  STR_SEGMENT              dw       (?)
      000E                        STR_HEADER               ENDS

                                  ;*****************************************************************
                                  ;  V_SYSTEM Global Data Segment
                                  ;*****************************************************************
                                  HP_GLB_HEADER            STRUC
      0000  ????                  T_HP_HEADER              dw       (?)
      0002     06 [               T_USED_AND_RESERVED      dw       6 dup (?)
                       ????
                              ]

      000E  ????                  T_HP_LAST_DS             dw       (?)
      0010  ????                  T_HP_MAX_DS              dw       (?)
      0012  ????                  T_HP_NXT_VCTR            dw       (?)
      0014  ??                    T_SND_FLAG               db       (?) ;
      0015  ??                    T_SND_CLICK_COUNT        db       (?) ;
      0016  ??                    T_SND_CLICK_DURA         db       (?) ;
      0017  ??                    T_SND_CLICK_VOLUME       db       (?) ;
      0018  ????                  T_SND_BEEP_CYCLE         dw       (?) ;
      001A  ????                  T_SND_BEEP_DURA          dw       (?) ;
      001C  ??                    T_SND_BEEP_COUNT         db       (?) ;
      001D  ??                                             db       (?) ;  1 reserved byte for volume
      001E  ????                  T_STR_NEXT_INDEX         dw       (?)
      0020  ????????              T_STR_ROOT               dd       (?)
      0024     0E [               T_STR_VCT_HDR            db       size STR_HEADER dup (?) ; Area vector's name and
                       ??
                            ]

      0032     0E [               T_STR_MSG_HDR            db       size STR_HEADER dup (?) ; ROM message strings
                       ??
                            ]

      0040  ??                    T_8259_FLAGS             db       (?)
      0041     1F [                                        db       31 dup (?) ; reserve 2 paragraph
                       ??
                            ]

      0060                        HP_GLB_HEADER            ENDS
                                  page
                                  ;*****************************************************************
                                  ; V_S8259 (1EH) 8259 interrupt controller support
                                  ;*****************************************************************
                                  ; F_ISR and F_SYSTEM are functions common to all drivers.

                                  ;
                                  ; Driver specific F_IO CONTROL subfunctions.

      = 0000                      SF_ENABLE_SVC             equ      00H      ; unmask svc/8041 interrupt
      = 0002                      SF_DISABLE_SVC            equ      02H      ; mask svc/8041 interrupt
      = 0004                      SF_ENABLE_KBD             equ      04H      ; unmask keyboard INT 9 interrupt
      = 0006                      SF_DISABLE_KBD            equ      06H      ; mask keyboard INT 9 interrupt
      = 0008                      SF_ENABLE_HPHIL           equ      08H      ; unmask HP-HIL interrupt
      = 000A                      SF_DISABLE_HPHIL          equ      0AH      ; mask HP-HIL interrupt

                                  ;*****************************************************************
                                  ; V_SINPUT (2AH) Function and subfunction codes
                                  ;*****************************************************************
                                  ; F_ISR and F_SYSTEM are functions common to all drivers.
                                  ;
                                  ; Driver Specific F_IO_CONTROL subfunctions.

      = 0000                      SF_DEF_LINKS              equ      0000H ; Sets default cofiguration
      = 0002                      SF_GET_LINKS              equ      0002H ; Reports current configuration
      = 0004                      SF_SET_LINKS              equ      0004H ; Sets Next Configuation


      = 0006                      F_INQUIRE                 equ      0006H
      = 0008                      F_INQUIRE_ALL             equ      0008H ; Reports ID's of devices
      = 000A                      F_INQUIRE_FIRST           equ      000AH ; Reports base HP-HIL address vector
      = 000C                      F_REPORT_ENTRY            equ      000CH ; Reports entry point of (V_PGID)

                                  page
                                  ;*****************************************************************
                                  ;  V_SVIDEO (54H) subfunction codes   Use these subfunctions when
                                  ;  calling the video driver directly.
                                  ;*****************************************************************
                                  ; F_ISR and F_SYSTEM are functions common to all drivers.
                                  ;
                                  ; Driver Specific F_IO_CONTROL subfunctions.
```

Equates File (continued)

```
                                             ;
 = 0000                                      SF_VID_ID_HP               equ     0
 = 0002                                      SF_VID_GET_INFO            equ     2
 = 0004                                      SF_VID_SET_INFO            equ     4
 = 0006                                      SF_VID_MOD_INFO            equ     6
 = 0008                                      SF_VID_GET_RES             equ     8
 = 000A                                      SF_VID_SET_MODE            equ     10

                                             ;*********************************************************************
                                             ; V_SVIDEO (54H) EX-BIOS Data Structures.
                                             ;*********************************************************************
 = 0027                                      VID_BLOCK_SIZE             equ     0027H
                                             VIDEO_DATA                 struc
 0000  ????                                  VID_ATR                    dw      ?
 0002  ????                                  VID_NAME_INDEX             dw      ?
 0004  ????                                  VID_V_DEFAULT              dw      ?
 0006  ??                                    VID_PRIMARY                db      ?        ; Which display is primary:
 0007  ??                                    VID_SECONDARY              db      ?        ; Which adapter is seconday (see above).
 0008  ??                                    VID_FOUND_ROM              db      ?        ; Flag set to true if rom found and init'ed
 0009      04 [                              VID_IDS                    db      4 DUP (?) ; Room for four possible ID's
                  ??
                        ]

 000D      04 [                              VID_STATUS                 db      4 DUP (?) ; Room for all status registers.
                  ??
                        ]

 0011      04 [                              VID_EXT_STATUS             db      4 DUP (?) ; Room for extended status registers.
                  ??
                        ]

 0015      27 [                              VID_PARM_BLOCK             db      VID_BLOCK_SIZE DUP (?)  ; Place to save parameters
                  ??
                        ]

 003C  ??                                    VID_LAST_IBM_MODE          db      (?)
 003D  ??                                    VID_EXT_MODE               db      (?)
 003E      02 [                              VID_PADDING                db      2H DUP (?)
                  ??
                        ]

 0040                                        VIDEO_DATA                 ends

                                             page
                                             ;*********************************************************************
                                             ; V_STRACK (5AH) called by logical GID drivers for cursor movement
                                             ;*********************************************************************
                                             ; F_ISR and F_SYSTEM are functions common to all drivers.

 = 0004                                      F_TRACK_INIT               equ     04H      ; sets tracking to default state
 = 0006                                      F_TRACK_ON                 equ     06H      ; enables tracking
 = 0008                                      F_TRACK_OFF                equ     08H      ; disables tracking
 = 000A                                      F_DEF_MASKS                equ     0AH      ; define sprite masks
 = 000C                                      F_SET_LIMITS_X             equ     0CH      ; set max/min horizontal values
 = 000E                                      F_SET_LIMITS_Y             equ     0EH      ; set max/min vertical values
 = 0010                                      F_PUT_SPRITE               equ     10H      ; display sprite at initial position
 = 0012                                      F_REMOVE_SPRITE            equ     12H      ; remove sprite from display

                                             ;*********************************************************************
                                             ; V_8041 (00AEH) Function and subfunction codes.
                                             ;*********************************************************************
                                             ; F_ISR and F_SYSTEM are functions common to all drivers.
                                             ;
                                             ; Driver Specific F_IO_CONTROL subfunctions.
                                             ;
                                                        ; Subfunction codes 0H, 2H, 4H, 6H and 8H are reserved
 = 000A                                      SF_CREAT_INTR              equ     000AH    ; Create Interval Entry.
 = 000C                                      SF_DELET_INTR              equ     000CH    ; Delete Interval Entry.
 = 000E                                      SF_ENABL_INTR              equ     000EH    ; Enable Interval.
 = 0010                                      SF_DISBL_INTR              equ     0010H    ; Disable Interval.
 = 0012                                      SF_SET_RAMSW               equ     0012H    ; Set RAM Switch to 1.
 = 0014                                      SF_CLR_RAMSW               equ     0014H    ; Clear RAM Switch to 0.
 = 0016                                      SF_SET_CRTSW               equ     0016H    ; Set CRT Switch to 1.
 = 0018                                      SF_CLR_CRTSW               equ     0018H    ; Clear CRT Switch to 0.
 = 001A                                      SF_PASS_THRU               equ     001AH    ; Pass Data Byte to HP8041.
                                                        ; Subfunction codes 1CH, 1EH, 20H, 22H, 24H, 26H, 28H, 2AH,
                                                        ; 2CH and 2EH are reserved.
                                             page
                                             ;*********************************************************************
                                             ; Physical Graphics Input Device (GID) Function Codes.
                                             ;*********************************************************************
                                             ; F_ISR and F_SYSTEM are functions common to all drivers.

                                             ;*********************************************************************
                                             ; Logical Graphics Input Device (GID) Function Codes. This is a common
                                             ; driver for: V_LTABLET, V_LPOINTER and V_LTOUCH.
                                             ;*********************************************************************
                                             ; F_ISR and F_SYSTEM are functions common to all drivers


                                             ;
                                             ; Driver specific F_IO_CONTROL subfunctions

 = 0004                                      SF_TRACK_ON                equ     4
 = 0006                                      SF_TRACK_OFF               equ     6
 = 0008                                      SF_CREATE_EVENT            equ     8
```

Equates File (continued)

```
      = 000A                    SF_EVENT_ON             equ     0Ah
      = 000C                    SF_EVENT_OFF            equ     0Ch
      = 000E                    SF_CLIPPING_ON          equ     0Eh
      = 0010                    SF_CLIPPING_OFF         equ     10h

      = 0006                    F_SAMPLE                equ     06

                                ;*****************************************************************
                                ; GID Data Structures
                                ;*****************************************************************

                                ;****************************************************************
                                ; Physical GID Describe Record
                                ;****************************************************************
                                DESCRIBE       STRUC
      0000      10 [                           db      size HP_SHEADER dup (?) ; this data is always offset by
                      ??
                          ]

      0010  ??                  D_SOURCE       db      ?       ; 7-4 (high nibble) contains the GID type
                                                               ; 3-0 (low nibble) is the address of the device
      0011  ??                  D_HPHIL_ID     db      ?       ; device id byte returned by an HP-HIL device
      0012  ??                  D_DESC_MASK    db      ?       ; describe header from HP-HIL device
      0013  ??                  D_IO_MASK      db      ?       ; I/O descriptor byte from device
      0014  ??                  D_XDESC_MASK   db      ?       ; extended describe byte from device
      0015  ??                  D_MAX_AXIS     db      ?       ; maximum number of axes reported
      0016  ??                  D_CLASS        db      ?       ; device class
                                                               ; 7-4 (high nibble) contains current class
                                                               ; 3-0 (low nibble) contain the default class
      0017  ??                  D_PROMPTS      db      ?       ; number of buttons/prompts
                                                               ; 7-4 (high nibble) is the number of prompts
                                                               ; 3-0 (low nibble) is the number of buttons
      0018  ??                  D_RESERVED     db      ?       ; reserved for future
      0019  ??                  D_BURST_LEN    db      ?       ; maximum burst length output to a device
                                                               ; if devices supports more than 255 bytes then
                                                               ; 255 bytes is the default maximum
      001A  ??                  D_WR_REG       db      ?       ; number of write registers supported by a device
      001B  ??                  D_RD_REG       db      ?       ; number of read registers supported by a device
      001C  ??                  D_TRANSITION   db      ?       ; transitions reported per button
      001D  ??                  D_STATE        db      ?       ; current state of buttons
      001E  ????                D_RESOLUTION   dw      ?       ; counts / cm (m) returned by HP-HIL device
      0020  ????                D_SIZE_X       dw      ?       ; Maximum count of in units of resolution
      0022  ????                D_SIZE_Y       dw      ?       ;
      0024  ????                D_ABS_X        dw      ?       ; data reported from device
      0026  ????                D_ABS_Y        dw      ?       ; that reports absolute data
      0028  ????                D_REL_X        dw      ?       ; data reported from device
      002A  ????                D_REL_Y        dw      ?       ; that is relitive
      002C  ????                D_ACCUM_X      dw      ?       ; these are used to accumulate scaling
      002E  ????                D_ACCUM_Y      dw      ?       ; remainder
      0030                      DESCRIBE       ENDS

      = 0030                    DESCRIBE_SIZE           equ     size DESCRIBE
      = 001E                    D_CCP_STATE             equ     D_STATE + 1
      =                         D_SIZE                  equ     D_SIZE_X
      =                         D_SAMPLE_ABSOLUTE       equ     D_ABS_X
      =                         D_SAMPLE_RELATIVE       equ     D_REL_X
      =                         D_REMAINDER_ACCUM       equ     D_ACCUM_X
      =                         D_BUFFER                equ     D_SIZE_X        ; offset where buffer begins
      = 00F0                    D_CLASS_CURRENT         equ     0F0H
      = 000F                    D_CLASS_DEFAULT         equ     00FH

                                ; The field D_SOURCE uses the following masks to access the defined nibbles
      = 000F                    D_ADDR_MASK           equ     00FH
      = 00F0                    D_TYPE_MASK           equ     0F0H

                                ;****************************************************************
                                ; Logical GID Describe Record
                                ;****************************************************************
                                LDESCRIBE      STRUC
      0000      10 [                           db      size HP_SHEADER dup (?) ; this data is always offset by
                      ??
                          ]

      0010  ??                  LD_SOURCE      db      ?       ; 7-4 (high nibble) contains the GID type
                                                               ; 3-0 reserved
      0011  ??                  LD_HPHIL_ID    db      ?       ; device id byte returned by an HP-HIL device
      0012  ????                LD_DEVICE_STATE dw     ?       ; status bits for logical device
      0014  ??                  LD_INDEX       db      ?       ; vector index of invoking driver
      0015  ??                  LD_MAX_AXIS    db      ?       ; maximum number of axis reported
      0016  ??                  LD_CLASS       db      ?       ; device class
                                                               ; 7-4 (high nibble) contains current class
                                                               ; 3-0 (low nibble) contain the default class
      0017  ??                  LD_PROMPTS     db      ?       ; number of buttons/prompts
                                                               ; 7-4 (high nibble) is the number of prompts
                                                               ; 3-0 (low nibble) is the number of buttons
      0018  ??                  LD_RESERVED    db      ?       ; reserved for future
      0019  ??                  LD_RES2        db      ?
      001A  ??                  LD_RES3        db      ?
      001B  ??                  LD_RES4        db      ?
      001C  ??                  LD_TRANSITION  db      ?       ; transitions reported per button
      001D  ??                  LD_STATE       db      ?       ; current state of buttons
      001E  ????                LD_RESOLUTION  dw      ?       ; counts / cm (m) returned by HP-HIL device
      0020  ????                LD_SIZE_X      dw      ?       ; Maximum count of in units of resolution
      0022  ????                LD_SIZE_Y      dw      ?       ;
```

Equates File (continued)

```
 0024  ????             LD_ABS_X        dw      ?       ; data reported from device
 0026  ????             LD_ABS_Y        dw      ?       ; that reports absolute data
 0028  ????             LD_REL_X        dw      ?       ; data reported from device
 002A  ????             LD_REL_Y        dw      ?       ; that is relitive
 002C  ????             LD_ACCUM_X      dw      ?       ; these are used to accumulate scaling
 002E  ????             LD_ACCUM_Y      dw      ?       ; remainders
 0030                   LDESCRIBE       ENDS
 = 0030                 LDESCRIBE_SIZE          equ     size LDESCRIBE

 =                      LD_SIZE                 equ     LD_SIZE_X
 =                      LD_SAMPLE_ABSOLUTE      equ     LD_ABS_X
 =                      LD_SAMPLE_RELATIVE      equ     LD_REL_X
 =                      LD_REMAINDER_ACCUM      equ     LD_ACCUM_X
 =                      LD_BUFFER               equ     LD_RESOLUTION   ; offset where buffer begins

                        ; the following masks are used in the field LD_CLASS
 = 00F0                 LD_CLASS_CURRENT        equ     0F0H
 = 000F                 LD_CLASS_DEFAULT        equ     00FH

                        ; The field LD_SOURCE uses the following masks to access the defined nibbles
 = 000F                 LD_RES_MASK             equ     00FH
 = 00F0                 LD_TYPE_MASK            equ     0F0H

                        ;*************************************************************
                        ; Logical GID LD_DEVICE_STATE
                        ;*************************************************************
 = 0010                 EVENT_ENABLED           equ     10h
 = 0008                 TRACK_ENABLED           equ     08h
 = 0004                 CLIP_ENABLED            equ     04h
 = 0002                 BUTTON_ERROR            equ     02H
 = 0001                 ISR_IN_PROGRESS         equ     01H

                        ;*************************************************************
                        ; D_SOURCE GID types
                        ;*************************************************************
 = 0000                 GID_R08                 equ     00H
 = 0001                 GID_R16                 equ     01H
 = 0002                 GID_A08                 equ     02H
 = 0003                 GID_A16                 equ     03H

 = 000F                 GID_UNDEF               equ     0FH     ; this undefined type is used because
                                                                ; the character input devices at the
                                                                ; current time do not have graphics input
                                                                ; and this tells the HP-HIL driver that
                                                                ; the data type to return is determined
                                                                ; by the scancodes the device returns
                        page
                        ;*************************************************************
                        ; D_CLASS and LD_CLASS device types
                        ;*************************************************************
 = 0000                 CLASS_KBD       equ     00H
 = 0001                 CLASS_TS        equ     01H
 = 0002                 CLASS_ASCII     equ     02H
 = 0003                 CLASS_BINARY    equ     03H
 = 0004                 CLASS_MOUSE     equ     04H
 = 0005                 CLASS_GIDCCP    equ     05H
 = 0006                 CLASS_TABLET    equ     06H
 = 0007                 CLASS_JOY       equ     07H
 = 0008                 CLASS_UNDEF8    equ     08H
 = 0009                 CLASS_PADDLE    equ     09H
 = 000A                 CLASS_THUMB     equ     0AH
 = 000B                 CLASS_TRACKBALL equ     0BH
 = 000C                 CLASS_KEYPAD    equ     0CH
 = 000D                 CLASS_UNDEFD    equ     0DH
 = 000E                 CLASS_UNDEFE    equ     0EH
 = 000F                 CLASS_UNDEFF    equ     0FH

                        ;*************************************************************
                        ; Keyboard and GID Event Data Types
                        ;*************************************************************
 = 0000                 T_KC_R0                 equ     00H     ; No key (should never occur)
 = 0001                 T_KC_R1                 equ     01H     ; reserved (see HP-HIL Technical
                                                                ; Reference Manual)
 = 0002                 T_KC_ASCII              equ     02H     ; ascii data
 = 0003                 T_KC_R3                 equ     03H     ; reserved (see HP-HIL Technical
                                                                ; Reference Manual)
 = 0004                 T_KC_ITF                equ     04H     ; ITF scancode reported by HP150 ITF keyboard,
                                                                ; HP200 ITF keyboard, barcode reader in scancode
                                                                ; mode.  ( Button reports are in this set )
 = 0005                 T_KC_R5                 equ     05H     ; reserved (see HP-HIL Technical
                                                                ; Reference Manual)
 = 0006                 T_KC_WILD               equ     06H     ; wild card set, device dependent, Button Pad uses
 = 0007                 T_KC_HPHIL_ENVOY        equ     07H     ; reported by VECTRA Keyboard
 = 0008                 T_KC_IBM_AT             equ     08H
 = 0009                 T_KC_BUTTON             equ     09H     ; reported by the physical driver to the logical
                                                                ; driver PGID translates T_KC_ITF to T_KC_BUTTON
                                                                ; and removes any other scancode from data stream
 = 000A                 T_KC_IBM_PC             equ     0AH
 = 000B                 T_KC_HP_SOFTKEY         equ     0BH
 = 000C                 T_KC_IS_FUNCTION        equ     0CH
 = 000D                 T_KC_HP_CCP             equ     0DH
 = 000E                 T_KC_QWERTY             equ     0EH
 = 000F                 T_KC_NUMPAD             equ     0FH
```

Equates File (continued)

```
                                  ; Bit definitions for Keyboard Event Data Types
                                  ;
      = 0010                      T_STRING               equ      010H
                                  ;T_STRING  00x1 ttttB  indicates a string of data bytes of type defined by the
                                  ;                      the lower nibble 'tttt'.  The state information only
                                  ;                      applies to the first byte of data as it can be
                                  ;                      modified by each subsequent byte of data.

      = 0020                      T_STATE                equ      020H
                                  ;T_STATE   001x ttttB  indicates the character type indicated
                                  ;                      in 'tttt' field has the current logical state
                                  ;                      of the keyboard appended onto it.

      = 0040                      T_GID                  equ      40H      ; used to set and test for gid types
      = 0040                      T_REL08                equ      40H      ;  normal mouse type data
      = 0041                      T_REL16                equ      41H
      = 0042                      T_ABS08                equ      42H      ,  normal TOUCH SCREEN Data
      = 0043                      T_ABS16                equ      43H      ;  normal TABLET data type
      = 0044                      T_MOUSE                equ      44H      ;  specially formed data
      = 0045                      T_TS                   equ      45H      ;  Specially formed data (0..80 x 0..25 default)
      = 0046                      T_TABLET               equ      46H      ;  Specially windowed data (640 x 200 default)
      = 0047                      T_POINTER              equ      47H      ;  specially windowed data (640 x 200 default)
      = 004F                      T_UNKNOWN              equ      4FH      ;  Unknown data type.


                                  ;*********************************************************************
                                  ; V_LHPMOUSE (00CCH) Function and subfunction codes.
                                  ;*********************************************************************
                                  ; F_ISR and F_SYSTEM are functions common to all drivers.


                                  ;
                                  ; Driver specific F_IO_CONTROL subfunctions.
                                  ;
      = 0000                      SF_MOUSE_COM           equ      0000H    ; This function is used during the
                                                                           ; reinit call from DOS.  It is used
                                                                           ; to set up INT 33H.  This is done
                                                                           ; because DOS takes INT 33H when it
                                                                           ; is initialized.
      = 0002                      SF_MOUSE_OVERRIDE      equ      0002H    ; This function is used to force the
                                                                           ; V_LHPMOUSE driver to install INT 33
                                                                           ; even when the mouse is not present.
                                                                           ; This allows a programmer to map
                                                                           ; devices to the V_LHPMOUSE driver if
                                                                           ; a mouse is not present.

                                  page
                                  ;*********************************************************************
                                  ; V_HPHIL (0114H) Function and subfunction codes.
                                  ;*********************************************************************
                                  ; F_ISR and F_SYSTEM are functions common to all drivers.


                                  ;
                                  ; Driver specific F_IO_CONTROL subfunctions.
                                  ;
      = 0004                      SF_CRV_CRV_MAJ_MIN     equ      0004     ; This is used to set a default
                                                                           ; major and minor addresses
      = 0006                      SF_CRV_RECONFIGURE     equ      0006     ; Funtion id used to force the HP-HIL
                                                                           ; link to reconfigure the devices
      = 0008                      SF_CRV_WR_PROMPTS      equ      0008     ; Used to write a prompt to a device
      =                           SF_CRV_WR_ACK          equ      000A     ; Used to write an acknowledge to a
                                                                           ; device
      =                           SF_CRV_REPEAT          equ      000C     ; Function is used to set a 30 Hz or
                                                                           ; 60 Hz repeat for keyboards
      =                           SF_CRV_DISABLE_REPEAT  equ      000E     ; Used to cancel the repeat rates in
                                                                           ; keyboards
      = 000A                      SF_CRV_SELF_TEST       equ      0010     ; Used to issue a selftest command
                                                                           ; to a physical device
      = 000C                      SF_CRV_REPORT_STATUS   equ      0012     ; Used to get the status information
                                                                           ; that an HP-HIL device might wish to
                                                                           ; report.  For specific information
                                                                           ; on what is reported, see the specs
                                                                           ; for the device.
      = 000E                      SF_CRV_REPORT_NAME     equ      0014     ; This function is used to return the
                                                                           ; ascii name that a device has
      = 0010                      SF_KEYBOARD_REPEAT     equ      0016     ; Used to set the keyboard repeat
                                                                           ; and delay rates
      = 0012                      SF_KEYBOARD_LED        equ      0018     ; Used to set the keyboard LEDs

                                  ; The functions F_PUT_BYTE, F_GET_BYTE and F_PUT_BUFFER are also supported
                                  ; in this driver.

                                  ;*********************************************************************
                                  ; System String Indexes. These are the indexes for the strings in ROM
                                  ; for the BASE system. If you need a particular string see the function
                                  ; F_STR_GET_STRING in the V_SYSTEM routines.
                                  ;*********************************************************************
      = 0800                      INDX_DRIVE_A           equ      2048+0
      = 0801                      INDX_DRIVE_B           equ      2048+1
      = 0802                      INDX_DRIVE_C           equ      2048+2
      = 0803                      INDX_DRIVE_D           equ      2048+3
      = 0804                      INDX_DRIVE_E           equ      2048+4
      = 0805                      INDX_DRIVE_F           equ      2048+5
      = 0806                      INDX_DRIVE_G           equ      2048+6
      = 0807                      INDX_DRIVE_H           equ      2048+7
      = 0808                      INDX_DRIVE_I           equ      2048+8
```

```
= 0809                 INDX_DRIVE_J          equ     2048+9
= 080A                 INDX_DRIVE_K          equ     2048+10
= 080B                 INDX_DRIVE_L          equ     2048+11
= 080C                 INDX_DRIVE_M          equ     2048+12
= 080D                 INDX_DRIVE_N          equ     2048+13
= 080E                 INDX_DRIVE_O          equ     2048+14
= 080F                 INDX_DRIVE_P          equ     2048+15
= 0810                 INDX_DRIVE_Q          equ     2048+16
= 0811                 INDX_DRIVE_R          equ     2048+17
= 0812                 INDX_DRIVE_S          equ     2048+18
= 0813                 INDX_DRIVE_T          equ     2048+19
= 0814                 INDX_DRIVE_U          equ     2048+20
= 0815                 INDX_DRIVE_V          equ     2048+21
= 0816                 INDX_DRIVE_W          equ     2048+22
= 0817                 INDX_DRIVE_X          equ     2048+23
= 0818                 INDX_DRIVE_Y          equ     2048+24
= 0819                 INDX_DRIVE_Z          equ     2048+25
= 081A                 INDX_HP_COPYRIGHT     equ     2048+26
= 081B                 INDX_SETUP_MSG        equ     2048+27
= 081C                 INDX_RETRY_MSG        equ     2048+28
= 081D                 INDX_INVALID_ROM_MSG  equ     2048+29
= 081E                 INDX_KYB_LOCKED_MSG   equ     2048+30
= 081F                 INDX_STRIKE_F1_MSG    equ     2048+31
= 0820                 INDX_BOOT_ERROR_MSG   equ     2048+32
= 0821                 INDX_TOUCH            equ     2048+33
= 0822                 INDX_TABLET           equ     2048+34
= 0823                 INDX_MOUSE            equ     2048+35
= 0824                 INDX_KEYBOARD         equ     2048+36
= 0825                 INDX_BARCODE          equ     2048+37
= 0826                 INDX_KNOB             equ     2048+38


                       ;*****************************************************************************
                       ; Industry Standard  equates.
                       ;*****************************************************************************

= 0000                 INT_DIVIDE_ZERO       equ     00H
= 0001                 INT_SINGLE_STEP       equ     01H
= 0002                 INT_NMI               equ     02H
= 0003                 INT_BREAKPOINT        equ     03H
= 0004                 INT_OVERFLOW          equ     04H

                       ;*****************************************************************************
                       ; Print Screen Service
                       ;*****************************************************************************
= 0005                 INT_PRINT_SCREEN      equ     05H

                       ;*****************************************************************************
                       ; 8259 Master Interrupt Controller Hardware Interrupts
                       ;*****************************************************************************
= 0008                 INT_IRQ0_TIMER        equ     08H
= 0009                 INT_IRQ1_KBD_ISR      equ     09H
= 000A                 INT_IRQ2              equ     0AH
= 000B                 INT_IRQ3_SERIAL1      equ     0BH
= 000C                 INT_IRQ4_SERIAL0      equ     0CH
= 000D                 INT_IRQ5_PRN1         equ     0DH
= 000E                 INT_IRQ6_FLOPPY       equ     0EH
= 000F                 INT_IRQ7_PRN0         equ     0FH


                       ;*****************************************************************************
                       ;  Interrupt 10H,  Video Support Functions
                       ;*****************************************************************************
= 0010                 INT_VIDEO             equ     10H     ;Video Functions Interrupt

= 0000                 F10_SET_MODE          equ     00H     ;Set Video Mode
= 0001                 F10_SET_CURSIZE       equ     01H     ;Set Cursor Size
= 0002                 F10_SET_CURPOS        equ     02H     ;Set Cursor Position
= 0003                 F10_RD_CURPOS         equ     03H     ;Read Cursor Position
= 0004                 F10_RD_PENPOS         equ     04H     ;Read Light-Pen Position
= 0005                 F10_SET_PAGE          equ     05H     ;Set Active Display Page
= 0006                 F10_SCROLL_UP         equ     06H     ;Scroll Rectangle Up
= 0007                 F10_SCROLL_DN         equ     07H     ;Scroll Rectangle Down
= 0008                 F10_RD_CHARATR        equ     08H     ;Read Character and Attribute at
                                                             ;Cursor Position
= 0009                 F10_WR_CHARATR        equ     09H     ;Write Character and Attribute at
                                                             ;Cursor Position
= 000A                 F10_WR_CHARCUR        equ     0AH     ;Write Character at Cursor Position
= 000B                 F10_SET_PALLET        equ     0BH     ;Set Color Pallet
= 000C                 F10_WR_PIXEL          equ     0CH     ;Write Pixel Dot
= 000D                 F10_RD_PIXEL          equ     0DH     ;Read Pixel Dot
= 000E                 F10_WR_CHARTEL        equ     0EH     ;Teletype Character Write
= 000F                 F10_GET_STMODE        equ     0FH     ;Get Video State and Mode
                               ; Function codes 10H - 12H are reserved
```

Equates File (continued)

```
                        ; Write String Functions
 = 1300                 F10_WRS_00            equ     1300H     ;Global Attribute
 = 1301                 F10_WRS_01            equ     1301H     ;Global Attribute, Move Cursor
 = 1302                 F10_WRS_02            equ     1302H     ;Individual Attributes
 = 1303                 F10_WRS_03            equ     1303H     ;Individual Attributes, Move Cursor
 = 6F00                 F10_INQUIRE           equ     6F00H     ;EX-BIOS present
 = 6F01                 F10_GET_INFO          equ     6F01H     ;Get video parameters
 = 6F02                 F10_SET_INFO          equ     6F02H     ;Sets video parameter
 = 6F03                 F10_MOD_INFO          equ     6F03H     ;Modifies video parameters
 = 6F04                 F10_GET_RES           equ     6F04H     ;Reports video resoultion
 = 6F05                 F10_XSET_MODE         equ     6F05H     ;Sets video resolution

                        ;*********************************************************************
                        ; Interrupt 11H, Equipment Determination Function
                        ;*********************************************************************
 = 0011                 INT_EQUIPMENT         equ     11H       ;Equipment Determination Interrupt

                        ;*********************************************************************
                        ;  Interrupt 12H, Report System Memory Size
                        ;*********************************************************************
 = 0012                 INT_MEM_SIZE          equ     12H       ;Memory Size Interrupt

                        ;*********************************************************************
                        ; Interrupt 13H, Internal Disc Support Functions (Flexible and Hard discs)
                        ;*********************************************************************
 = 0013                 INT_DISC              equ     13H       ;Hard Disc Functions Interrupt

 = 0000                 F13_RESET_DISC        equ     00H       ;Reset Hard Disc
 = 0001                 F13_RD_LSTATUS        equ     01H       ;Read Status of Last Operation
 = 0002                 F13_RD_SECTORS        equ     02H       ;Read Sectors
 = 0003                 F13_WR_SECTORS        equ     03H       ;Write Sectors
 = 0004                 F13_VR_SECTORS        equ     04H       ;Verify Sectors
 = 0005                 F13_FORMAT_FLEX       equ     05H       ;Format Diskette Track
                                ; Function code 06H is reserved
 = 0007                 F13_FORMAT_HDISC      equ     07H       ;Format Hard Disc
 = 0008                 F13_GET_HPARMS        equ     08H       ;Get Hard Disc Parameters
                                ; Function codes 09H - 0BH are reserved
 = 000C                 F13_TRACK_SEEK        equ     0CH       ;Seek to Track
 = 000D                 F13_ALT_RESET         equ     0DH       ;Alternate Hard Disc Reset
                                ; Function codes 0EH - 014H are reserved
 = 0015                 F13_GET_DASD          equ     15H       ;Read Disc Type (DASD)
 = 0016                 F13_CHG_STATUS        equ     16H       ;Get Disc Change Line Status
 = 0017                 F13_SET_DASD          equ     17H       ;Set Disc Type for Flexible Disc
                                                                ;  Formatting(DASD)

                        ;*********************************************************************
                        ; Interrupt 14H, Serial Communications Functions
                        ;*********************************************************************
 = 0014                 INT_SERIAL            equ     14H       ;Serial Communications driver

 = 0000                 F14_INIT              equ     00H       ;Initialize Serial Port Parameters
 = 0001                 F14_XMIT              equ     01H       ;Send Out One Character
 = 0002                 F14_RECV              equ     02H       ;Receive One Character
 = 0003                 F14_STATUS            equ     03H       ;Get Serial Port Status

 = 6F00                 F14_INQUIRE           equ     6F00H     ;Reports if EX-BIOS Functions
                                                                ; are present
 = 6F01                 F14_EXINIT            equ     6F01H     ;Initializes serial port
                                                                ; (19.2 Kbaud)
 = 6F02                 F14_PUT_BUFFER        equ     6F02H     ;Writes a buffer of data
 = 6F03                 F14_GET_BUFFER        equ     6F03H     ;Reads a buffer of data
 = 6F04                 F14_TRM_BUFFER        equ     6F04H     ;Reads a buffer of data,
                                                                ;terminates on specified condition.

                        ;*********************************************************************
                        ; Interrupt 15H,  System Control Functions (Processor Mode Switch, Extended
                        ;                                           Memory Functions, ETC.)
                        ;*********************************************************************
 = 0015                 INT_SYSTEM            equ     15H       ;System/Cassette Functions Interrupt

                                ; function codes 0 - 3 for Cassette Handling are Unsupported
 = 0080                 F15_DEVICE_OPEN       equ     80H       ;Device Open
 = 0081                 F15_DEVICE_CLOSE      equ     81H       ;Device Close
 = 0082                 F15_PROG_TERM         equ     82H       ;Program Termination
 = 0083                 F15_WAIT_EVENT        equ     83H       ;Event Wait
 = 0084                 F15_JOYSTICK          equ     84H       ;Joystick Support
 = 0085                 F15_SYS_REQ           equ     85H       ;System Request Key Pressed
 = 0086                 F15_WAIT              equ     86H       ;Wait Fixed Amount of Time
 = 0087                 F15_BLOCK_MOVE        equ     87H       ;Move Block of Memory to/from
                                                                ;Extended Memory
 = 0088                 F15_GET_XMEM_SIZE     equ     88H       ;Get Extended Memory Size
 = 0089                 F15_ENTER_PROT        equ     89H       ;Switch to Protected Mode
 = 008A                 F15_DEV_BUSY          equ     8AH       ;Device Busy Loop
 = 008B                 F15_INT_COMPLETE      equ     8BH       ;Set Interrupt Completed Flag

                        ;*********************************************************************
                        ; Interrupt 16H, Keyboard Support Functions
                        ;*********************************************************************
 = 0016                 INT_KBD               equ     16H       ;Keyboard Driver

 = 0000                 F16_GET_KEY           equ     00H       ;Read keycode from Keyboard Buffer
 = 0001                 F16_STATUS            equ     01H       ;Report Status of keyboard buffer
 = 0002                 F16_KEY_STATE         equ     02H       ;Get Key Modifier Status
```

Equates File (continued)

```
= 6F00              F16_INQUIRE          equ     6F00H   ;Inquire EX-BIOS Functions present
= 6F01              F16_DEF_ATTR         equ     6F01H   ;Reports default values for repeat
                                                         ;rates and delay time before repeat.
= 6F02              F16_GET_ATTR         equ     6F02H   ;Reports current repeat rates and
                                                         ;delay time.
= 6F03              F16_SET_ATTR         equ     6F03H   ;Replaces current repeat rates and
                                                         ;delay time.
= 6F04              F16_DEF_MAPPING      equ     6F04H   ;Reports default HP-System vector
                                                         ;entries for keyboard translator
                                                         ;drivers.
= 6F05              F16_GET_MAPPING      equ     6F05H   ;Reports current HP-System vector
                                                         ; entries for keyboard translator
                                                         ; drivers.
= 6F06              F16_SET_MAPPING      equ     6F06H   ;Replaces current HP-System vector
                                                         ; entries for keyboard translator
                                                         ; drivers.
= 6F07              F16_SET_XLATORS      equ     6F07H   ;Switches either the cursor control
                                                         ; pad translator or the HP Softkey
                                                         ; translator functions.
= 6F08              F16_KBD              equ     6F08H   ;Reports keyboard HP-HIL address and
                                                         ; Identification.
= 6F09              F16_KBD_RESET        equ     6F09H   ;Resets logical keyboard structure
                                                         ; to defaults.

                    ;**************************************************************************
                    ; Interrupt 17H, Printer Support Functions
                    ;**************************************************************************
= 0017              INT_PRINTER          equ     17H     ;Printer Port Support Interrupt

= 0000              F17_PUT_CHAR         equ     00H     ;Send Printer One Byte
= 0001              F17_INIT             equ     01H     ;Initialize Printer
= 0002              F17_STATUS           equ     02H     ;Get Printer Status

= 6F00              F17_INQUIRE          equ     6F00H   ;Reports EX-BIOS functions exists
= 6F01              F17_READ_STATUS      equ     6F01H   ;Reports the status of a printer
                                                         ; port read buffer.
= 6F02              F17_PUT_BUFFER       equ     6F02H   ;Writes a buffer of data to the
                                                         ; printer port.
= 6F03              F17_GET_BUFFER       equ     6F03H   ;Reads a buffer of data.
= 6F04              F17_TRM_BUFFER       equ     6F04H   ;Reads a buffer of data from the
                                                         ; port, terminates on specified
                                                         ; condition

                    ;**************************************************************************
                    ; Reboot System
                    ;**************************************************************************
= 0019              INT_BOOT             equ     19H     ; Reboot System

                    ;**************************************************************************
                    ; Interrupt 1AH,  Real-Time Clock Support Functions
                    ;**************************************************************************
= 001A              INT_CLOCK            equ     1AH     ; Clock Functions Interrupt

= 0000              F1A_RD_CLK_CNT       equ     00H     ;Read Current Clock Count
= 0001              F1A_SET_CLK_CNT      equ     01H     ;Set Current Clock Count
= 0002              F1A_GET_RTC          equ     02H     ;Read Real-Time Clock
= 0003              F1A_SET_RTC          equ     03H     ;Set Real-Time Clock
= 0004              F1A_GET_DATE         equ     04H     ;Read Date from Real-Time Clock
= 0005              F1A_SET_DATE         equ     05H     ;Set Date in Real-Time Clock
= 0006              F1A_SET_ALARM        equ     06H     ;Set Alarm
= 0007              F1A_RESET_ALARM      equ     07H     ;Reset Alarm

                    ;**************************************************************************
                    ; Interrupt 1BH,  Break Key
                    ;**************************************************************************
= 001B              INT_BREAK_EVENT      equ     1BH

                    ;**************************************************************************
                    ;  Timer Tick Event
                    ;**************************************************************************
= 001C              INT_TIMER_TICK       equ     1CH

                    ;**************************************************************************
                    ; Video Parameters
                    ;**************************************************************************
= 001D              INT_VIDEO_PARMS      equ     1DH

                    ;**************************************************************************
                    ; Floppy Parameters
                    ;**************************************************************************
= 001E              INT_FLOPPY_PARMS     equ     1EH

                    ;**************************************************************************
                    ; Graphics Characters Table
                    ;**************************************************************************
= 001F              INT_GRAPHICS_CHAR    equ     1FH

                    ;**************************************************************************
                    ; DOS Function call interrupt
                    ;**************************************************************************
= 0021              INT_DOS              equ     21H

                    ;**************************************************************************
                    ; Interrupt 33H,  HP Mouse Support (MS-Mouse Emulation)
```

Equates File (continued)

```
                        ; Full AX register used for function code
                        ;***************************************************************************
= 0033                  INT_HPMOUSE              equ     33H
= 0000                  F33_INSTALL          equ       0000H   ; Mouse installed flag and reset hardware.
= 0001                  F33_ENABLE           equ       0001H   ; Put cursor on screen.
= 0002                  F33_DISABLE          equ       0002H   ; Turn off cursor.
= 0003                  F33_REPORT_DATA      equ       0003H   ; Get positional data and button info
= 0004                  F33_PUT_CURSOR       equ       0004H   ; Positon the cursor.
= 0005                  F33_REPORT_PRESS     equ       0005H   ; Report button press status
= 0006                  F33_REPORT_RELEASE   equ       0006H   ; Report button release information.
= 0007                  F33_SET_HORIZ        equ       0007H   ; Set minimum and maximum horizontal
                                                                 ; values.
= 0008                  F33_SET_VERT         equ       0008H   ; Set min and max vertical values.
= 0009                  F33_GRAPH_CURSOR     equ       0009H   ; Define the graphics cursor.
= 000A                  F33_TEXT_CURSOR      equ       000AH   ; Define the text cursor.
= 000B                  F33_MOTION           equ       000BH   ; Report motion counters.
= 000C                  F33_SET_USR          equ       000CH   ; Define user subroutine call.
= 000D                  F33_ENABLE_LIGHT     equ       000DH   ; Enable light pen emulation mode.
= 000E                  F33_DISABLE_LIGHT    equ       000EH   ; Disable  light pen emulation.
= 000F                  F33_RATIO            equ       000FH   ; Set pixel movement ratio.
= 0010                  F33_COND_OFF         equ       0010H   ; Define area to conditionally turn
                                                                 ; display off.  Not used.
= 0012                  F33_XTEND_GCSR       equ       0012H   ; Extended sprite graphics entry point.
= 0013                  F33_SPEED            equ       0013H   ; Set mouse doubling speed factor.

= 6F00                  F33_INQUIRE          equ       6F00H   ; Returns "HP" in bx if HPMOUSE driver
                                                               ; is being used.

                        ;***************************************************************************
                        ; When there is a fixed disc installed, direct entrypoint to floppy
                        ;***************************************************************************
= 0040                  INT_FLOPPY_DIRECT        equ     40H

                        ;***************************************************************************
                        ; Fixed Disc Parameters
                        ;***************************************************************************
= 0041                  INT_HDISC_PARMS0         equ     41H

                        ;***************************************************************************
                        ; Fixed Disc Parameters
                        ;***************************************************************************
= 0046                  INT_HDISC_PARMS1         equ     46H

                        ;***************************************************************************
                        ; Real Time Clock Event vector
                        ;***************************************************************************
= 004A                  INT_RTC_EVENT            equ     4AH

                        ;***************************************************************************
                        ; 8259 EX-BIOS Slave Interrupt Controller
                        ;***************************************************************************
= 0068                  INT_SVC_REQUEST          equ     68H
= 0069                  INT_8041_OBF             equ     69H
= 006C                  INT_HPHIL                equ     6CH

                        ;***************************************************************************
                        ; 8259 Slave Interrupt Controller Interrupts
                        ;***************************************************************************

= 0070                  IRQ8_RTC                 equ     70H
= 0071                  IRQ9_REDIRECT            equ     71H
= 0072                  IRQ10                    equ     72H
= 0073                  IRQ11                    equ     73H
= 0074                  IRQ12                    equ     74H
= 0075                  IRQ13_287                equ     75H
= 0076                  IRQ14_HDISC              equ     76H
= 0077                  IRQ15                    equ     77H

                        ;***************************************************************************
                        ; MS-DOS Installable Device Driver Equates and Structures
                        ;***************************************************************************
= 0081                  MSD_ERR_STATUS  equ      10000001B        ;used as upper byte in status wrd
= 0001                  MSD_DONE_STATUS equ      00000001B        ;  bit 15=err bit 8=done
                        ; Equates for standard MSDOS errors.
= 0000                  MSD_WRITE_PROTECT        equ     0
= 0001                  MSD_UNKNOWN_UNIT         equ     01H
= 0002                  MSD_NOT_READY            equ     02H
= 0003                  MSD_UNKNOWN_CMD          equ     03H
= 0004                  MSD_CRC_ERROR            equ     04H
= 0005                  MSD_BAD_LENGTH           equ     05H      ;bad request length error
= 0006                  MSD_SEEK_ERROR           equ     06H
= 0007                  MSD_UNKNOWN_MEDIA        equ     07H
= 0008                  MSD_SEC_NOTFND           equ     08H      ;sector not found
= 0009                  MSD_PAPER_OUT            equ     09H
= 000A                  MSD_WRITE_FAULT          equ     0AH
= 000B                  MSD_READ_FAULT           equ     0BH
= 000C                  MSD_GEN_FAILURE          equ     0CH
= 000D                  MSD_BAD_DCHG             equ     0DH      ;bad disk change error

                        ;******************************************************************************
                        ; Command Equates => the following list are the commands defined for MS-DOS
                        ;    installable device drivers.
                        ;******************************************************************************
= 0000                  MSD_INIT                 equ     0
= 0001                  MSD_MEDIA_CHK            equ     01H              ;media check command
```

Equates File (continued)

```
= 0002                   MSD_BLD_BPB          equ     02H            ;build Bios Parameter Block
= 0003                   MSD_IOCTL_IN         equ     03H            ;I/O control input
= 0004                   MSD_INPUT            equ     04H
= 0005                   MSD_IN_NOWAIT        equ     05H            ;Non-destructive input with no wait
= 0006                   MSD_IN_STATUS        equ     06H
= 0007                   MSD_IN_FLUSH         equ     07H
= 0008                   MSD_OUTPUT           equ     08H
= 0009                   MSD_OUT_VERIFY       equ     09H
= 000A                   MSD_OUT_STATUS       equ     0AH
= 000B                   MSD_OUT_FLUSH        equ     0BH
= 000C                   MSD_IOCTL_OUT        equ     0CH
= 000D                   MSD_DEV_OPEN         equ     0DH            ;device open and close commands
= 000E                   MSD_DEV_CLOSE        equ     0EH
= 000F                   MSD_REM_MEDIA        equ     0FH            ;removable media command
                         page
                         ;***********************************************************************
                         ;  use this macro to setup the MS-DOS driver header required at the top of
                         ;  any installable device driver
                         ;***********************************************************************
                         msd_header      macro    ATT,STRATEGY_ENTRY,ISR_ENTRY,STRING
                                         dd       -1              ;mark as last driver in list
                                         dw       ATT
                                         dw       STRATEGY_ENTRY
                                         dw       ISR_ENTRY
                                         db       STRING
                                         db       14 dup (?)     ; Pad so it is paragraph aligned.
                                         endm

                         ;***********************************************************************
                         ; the following structures are used to access MS-DOS driver command blocks
                         ;***********************************************************************
                         MSD_REQ_HEADER  struc           ;00; structure for access to MS driver cmds
0000  ??                 MSD_CMDLEN      db      ?       ;00; length of cmd in bytes including data @ end
0001  ??                 MSD_UNIT        db      ?       ;01; unit number for command
0002  ??                 MSD_CMD         db      ?       ;02; command code
0003  ????               MSD_STATUS      dw      ?       ;03; filler with completion status before return
0005      08 [                           db 8 dup (?)    ;  ; area reserved for DOS
                  ??
                        ]

000D  ??                 MSD_MEDIA       db      ?       ;13; most cmds have this defined in the data area
000E  ????               MSD_TRANS       dw      ?       ;14;
0010  ????                               dw      ?       ;16;
0012  ????               MSD_COUNT       dw      ?       ;18;
0014  ????               MSD_START       dw      ?       ;20;
0016                     MSD_REQ_HEADER  ends
                         ;***********************************************************************
                         ; Access to the data area of the INIT driver command
                         ;***********************************************************************
                         MSD_INIT_CMD    struc
0000      0D [                           db      13 dup (?)    ;first cover header area
                  ??
                        ]

000D  ??                 MSD_UNIT_COUNT  db      ?       ; 0B ;number of units service by this driver
000E  ????               MSD_END_OFFSET  dw      ?       ; 0C ;offset of end of code
0010  ????               MSD_END_SEG     dw      ?       ; 0E ;segment address of end of code
0012  ????               MSD_BPB_OFFSET  dw      ?       ; 12 ;
0014  ????               MSD_BPB_SEG     dw      ?       ; 14 ;seg:offset of BPB list for units attached
0016  ??                 MSD_1ST_UNIT    db      ?       ; 16 ;tells driver letter of first unit
0017                     MSD_INIT_CMD    ends
                         page
                         ;***********************************************************************
                         ; Access to the data area for INPUT or OUTPUT driver commands
                         ;***********************************************************************
                         MSD_IO_CMD      struc
0000      0E [                           db      14 dup (?) ;Media byte defined in header.
                  ??
                        ]

000E  ????               MSD_XFER_OFFSET dw      ?
0010  ????               MSD_XFER_SEG    dw      ?       ;full address of buffer for data transfer
0012  ????               MSD_XFER_COUNT  dw      ?       ;could be bytes or block count
0014  ????               MSD_1ST_BLK     dw      ?       ;address of first block to read or write
0016  ????               MSD_VERR_SEG    dw      ?       ;pointer to volume id if err code => 0Fh
0018                     MSD_IO_CMD      ends



                         ;***********************************************************************
                         ;
                         ; Structure Definition For  segment 40h, STD-BIOS Data Area
                         ;
                         ;***********************************************************************
                         SEGMENT40       struc
                         ; EIA communication base I/O port address table
0000  ????               S40_RS232_PORT1_ADR     dw  ?          ; 040:0000 address of serial port 1
0002  ????               S40_RS232_PORT2_ADR     dw  ?          ; 040:0002 address of serial port 2
0004  ????               S40_RS232_PORT3_ADR     dw  ?          ; 040:0004 address of serial port 3
0006  ????               S40_RS232_PORT4_ADR     dw  ?          ; 040:0006 address of serial port 4
                         ;
                         ; Parallel printer base I/O port address table
0008  ????               S40_PRINT_PORT1_ADR     dw  ?          ; 040:0008 address of parallel port 1
```

Equates File (continued)

```
000A  ????                   S40_PRINT_PORT2_ADR      dw   ?            ; 040:000A address of parallel port 2
000C  ????                   S40_PRINT_PORT3_ADR      dw   ?            ; 040:000C address of parallel port 3
000E  ????                   S40_PRINT_PORT4_ADR      dw   ?            ; 040:000E address of parallel port 4
                             ;
                             ; System configuration
0010  ????                   S40_EQUIPMENT_FLAG       dw   ?            ; 040:0010 word identifying installed devices
0012  ??                     S40_MFG_INIT             db   ?            ; 040:0012 manufacturing initailization/test byte
0013  ????                   S40_MEMORY_SIZE          dw   ?            ; 040:0013 memory size in 1k bytes
0015  ??                     S40_MFG_ERR_FLAG1        db   ?            ; 040:0015 manufacturing scratchpad
0016  ??                     S40_MFG_ERR_FLAG2        db   ?            ; 040:0016 manufacturing error codes
                             ;
                             ; Keyboard data area
0017  ??                     S40_KBD_STATE1           db   ?            ; 040:0017 state of special keys: shift, caps, etc.
0018  ??                     S40_KBD_STATE2           db   ?            ; 040:0018 secondary state of special keys

0019  ??                     S40_ALT_INPUT_ACCUM      db   ?            ; 040:0019 accumulator for alt/numpad entry
001A  ????                   S40_KBD_BUF_HEAD         dw   ?            ; 040:001A keyboard buffer head pointer
001C  ????                   S40_KBD_BUF_TAIL         dw   ?            ; 040:001C keyboard buffer tail pointer
001E      10 [               S40_KBD_BUFFER           dw   16 dup (?)   ; 040:001E keyboard buffer, 15 entries + overrun
                ????
                      ]

                             ;
                             ; Floppy diskette data area
003E  ??                     S40_FLOPPY_SEEK_STAT     db   ?            ; 040:003E floppy drive status
003F  ??                     S40_FLOPPY_MOTOR_STAT    db   ?            ; 040:003F floppy drive motor status
0040  ??                     S40_FLOPPY_TIME_OUT      db   ?            ; 040:0040 floppy drive timeout counter
0041  ??                     S40_FLOPPY_RETURN_STAT   db   ?            ; 040:0041 floppy drive return code/error status
0042      07 [               S40_FLOPPY_CONTRL_STAT   db   7 dup (?)    ; 040:0042 floppy controller status/hard disk
                ??
                    ]

                                                                        ;          command/param port copies
                             ; CRT video display area
0049  ??                     S40_CRT_MODE             db   ?            ; 040:0049 current video mode
004A  ????                   S40_CRT_WIDTH            dw   ?            ; 040:004A current number of screen columns
004C  ????                   S40_CRT_LENGTH           dw   ?            ; 040:004C current length of screen in bytes
004E  ????                   S40_CRT_PAGE_ADR         dw   ?            ; 040:004E starting address of current display page
0050      08 [               S40_CRT_CURSOR_POS       dw   8 dup (?)    ; 040:0050 cursor coordinates (row,column)
                ????
                      ]

                                                                        ;          for up to 8 pages
0060  ????                   S40_CRT_CURSOR_MODE      dw   ?            ; 040:0060 current cursor mode setting
0062  ??                     S40_CRT_DISPLAY_PAGE     db   ?            ; 040:0062 current display page
0063  ????                   S40_CRT_PORT_ADR         dw   ?            ; 040:0063 base I/O port address for
                                                                        ;          active crt controller
0065  ??                     S40_CRT_MODE_SEL_REG     db   ?            ; 040:0065 mode select register copy
0066  ??                     S40_CRT_PALETTE          db   ?            ; 040:0066 color palette register copy
                             ;
                             ; Option ROM data area
0067  ????                   S40_XROM_INIT_ADR        dw   ?            ; 040:0067 offset address for optional
                                                                        ;          I/O rom init routine
0069  ????                   S40_XROM_SEGMENT         dw   ?            ; 040:0069 segment address for optional I/O rom
006B  ??                     S40_XROM_INT_FLAG        db   ?            ; 040:006B flags last interrupt that occured
                             ;
                             ; Timer data area
006C  ????                   S40_TIMR_LOW             dw   ?            ; 040:006C least significant word of timer count
006E  ????                   S40_TIMR_HIGH            dw   ?            ; 040:006E Most significant word of timer count
0070  ??                     S40_TIMR_OVR_FLOW        db   ?            ; 040:0070 24-hour timer tick rollover counter
                             ;
                             ; System data area
0071  ??                     S40_SYS_BREAK_FLAG       db   ?            ; 040:0071 system break request flag
0072  ????                   S40_SYS_RESET_FLAG       dw   ?            ; 040:0072 system reset flag
                             ;
                             ; Hard disk data area
0074  ??                     S40_FD_STATUS            db   ?            ; 040:0074 hard disk status, last int 13 operation
0075  ??                     S40_FD_COUNT             db   ?            ; 040:0075 number of hard disks present
0076  ??                     S40_FD_CONTROL           db   ?            ; 040:0076 copy of hard disk controller register
0077  ??                     S40_FD_PORT_OFFSET       db   ?            ; 040:0077 hard disk port offset
                             ;
                             ; Parallel printer timeout table
0078  ??                     S40_PRINT_TIMEOUT1       db   ?            ; 040:0078 parallel printer 1 timeout count
0079  ??                     S40_PRINT_TIMEOUT2       db   ?            ; 040:0079 parallel printer 2 timeout count
007A  ??                     S40_PRINT_TIMEOUT3       db   ?            ; 040:007A parallel printer 3 timeout count
007B  ??                     S40_PRINT_TIMEOUT4       db   ?            ; 040:007B parallel printer 4 timeout count
                             ;
                             ;
                             ; Serial port timeout table
007C  ??                     S40_RS232_TIMEOUT1       db   ?            ; 040:007C serial port 1 timeout count
007D  ??                     S40_RS232_TIMEOUT2       db   ?            ; 040:007D serial port 2 timeout count
007E  ??                     S40_RS232_TIMEOUT3       db   ?            ; 040:007E serial port 3 timeout count
007F  ??                     S40_RS232_TIMEOUT4       db   ?            ; 040:007F serial port 4 timeout count
                             ;
                             ; Keyboard buffer pointers
0080  ????                   S40_KBD_BUF_START        dw   ?            ; 040:0080 pointer to physical start of
                                                                        ;          keyboard buffer
0082  ????                   S40_KBD_BUF_END          dw   ?            ; 040:0082 pointer to physical end of
                                                                        ;          keyboard buffer
                             ; Enhanced graphics adapter (EGA) data area
0084  ??                     S40_EGA_CRT_ROW_CNT      db   ?            ; 040:0084 number of crt rows minus one
0085  ????                   S40_EGA_CHAR_SIZE        dw   ?            ; 040:0085 number of bytes per character
                                                                        ;          in font table
```

Equates File (continued)

```
0087  ??                        S40_EGA_INFO1           db   ?           ; 040:0087 EGA miscellaneous information
0088  ??                        S40_EGA_INFO2           db   ?           ; 040:0088 EGA miscellaneous information
                                ;
                                ; Reserved
0089     02 [                                           db   2 dup (?)   ; 040:0089
                 ??
                      ]


                                ; Floppy disk rate area
008B  ??                        S40_FLOPPY_LAST_RATE    db ?             ; 040:008B last floppy data rate selected
                                ; Additional fixed disk data area
008C  ??                        S40_AFD_STATUS_REG      db   ?           ; 040:008C fixed disk status register copy
008D  ??                        S40_AFD_ERROR_REG       db   ?           ; 040:008D fixed disk error register copy
008E  ??                        S40_AFD_INTR_FLAG       db   ?           ; 040:008E fixed disk interrupt flag
008F  ??                        S40_AFD_CTRL_FLAG       db   ?           ; 040:008F fixed disk controller flag
                                ;
                                ; Additional floppy diskette data area
0090  ??                        S40_AFLOPPY_MEDIA0      db   ?           ; 040:0090 drive 0 media state
0091  ??                        S40_AFLOPPY_MEDIA1      db   ?           ; 040:0091 drive 1 media state
0092  ??                        S40_AFLOPPY_OPER0       db   ?           ; 040:0092 drive 0 operation state
0093  ??                        S40_AFLOPPY_OPER1       db   ?           ; 040:0093 drive 1 operation state
0094  ??                        S40_AFLOPPY_TRACK0      db   ?           ; 040:0094 drive 0 current track
0095  ??                        S40_AFLOPPY_TRACK1      db   ?           ; 040:0095 drive 1 current track
0096  ??                        S40_AFLOPPY_RESERVED    db   ?           ; 040:0096 floppy disk reserved byte
                                ;
                                ; Keyboard LED data area
0097  ??                        S40_KBD_LED_FLAGS       db   ?           ; 040:0097 keyboard LED flags
                                ; Real-Time clock data area
0098  ????                      S40_RTC_WAIT_OFFSET     dw   ?           ; 040:0098 offset address of user wait flag
009A  ????                      S40_RTC_WAIT_SEGMENT    dw   ?           ; 040:009A segment address of user wait flag
009C  ????                      S40_RTC_WAIT_CNT_LOW    dw   ?           ; 040:009C low word of wait count
009E  ????                      S40_RTC_WAIT_CNT_HIGH   dw   ?           ; 040:009E high word of wait count
00A0  ??                        S40_RTC_WAIT_ACTV_FLG   db   ?           ; 040:00A0 wait active flag
                                ;
                                ; Reserved
00A1     07 [                                           db   7 dup (?)   ; 040:00A1
                 ??
                      ]


                                ;
                                ; Pointer to EGA data area
00A8  ????????                  S40_EGA_TBL_PTR         dd   ?           ; 040:00A8 pointer to table of EGA pointers
                                ;
                                ; Reserved
00AC     44 [                                           db   68 dup (?)  ; 040:00AC
                 ??
                      ]


                                ;
                                ; Intra-application communications area
00F0     10 [                   S40_INTRA_APPL          db   16 dup (?)  ; 040:00F0 available to any application
                 ??
                      ]


                                ;
                                ; Print screen status
0100  ??                        S40_PSCRN_STATUS        db   ?           ; 040:0100 flag for print screen in progress
                                ;
                                ; Reserved
0101     03 [                                           db   3 dup (?)   ; 040:0101
                 ??
                      ]


                                ;
                                ; DOS data area
0104  ??                        S40_SINGLE_DRV_STAT     db   ?           ; 040:0104 status of floppy for single floppy
                                                                         ;          systems, ie currently drive A: or B:
                                ; Reserved
0105     19 [                                           db   25 dup (?)  ; 040:0105
                 ??
                      ]

011E                            SEGMENT40       ends



                                ;*************************************************************************
                                ; STD-BIOS table addresses
                                ;
 = 0000                         S40_RS232_PORT_TBL      equ              word ptr S40_RS232_PORT1_ADR
 = 0008                         S40_PRINT_PORT_TBL      equ              word ptr S40_PRINT_PORT1_ADR
 = 0078                         S40_PRINT_TIMEOUT_TBL   equ              byte ptr S40_PRINT_TIMEOUT1
 = 007C                         S40_RS232_TIMEOUT_TBL   equ              byte ptr S40_RS232_TIMEOUT1
 = 0090                         S40_AFLOPPY_MEDIA       equ              byte ptr S40_AFLOPPY_MEDIA0
 = 0092                         S40_AFLOPPY_OPER        equ              byte ptr S40_AFLOPPY_OPER0
 = 0094                         S40_AFLOPPY_TRACK       equ              byte ptr S40_AFLOPPY_TRACK0
                                ;*************************************************************************

                                 page



                                ;*************************************************************************
                                ; S40_EQUIPMENT_FLAG    word
                                ;*************************************************************************
```

Equates File (continued)

```
                          ;                                      FEDCBA9876543210b
        = C000            S40E_DEVICE_PRINTRS     equ      1100000000000000b       ; number of printers
        = 0E00            S40E_DEVICE_RS232       equ      0000111000000000b       ; number of RS232 ports
        = 00C0            S40E_DEVICE_FLOPPY      equ      0000000011000000b       ; number of floppy drives
        = 0030            S40E_DEVICE_VIDEO       equ      0000000000110000b       ; initial video mode
        = 0002            S40E_DEVICE_MATH        equ      0000000000000010b       ; 80287 installed
        = 0001            S40E_DEVICE_BOOT        equ      0000000000000001b       ; floppy boot device present
                          ;*************************************************************************
                          ;
                          ;         Bit     Value    Definition
                          ;         F-E     0        no printers installed
                          ;                 1        one printer installed
                          ;                 2        two printers installed
                          ;                 3        three printers installed
                          ;         D-C     ----     reserved
                          ;         B-9     0        no RS-232 ports installed
                          ;                 1        one RS-232 port installed
                          ;                 2        two RS-232 ports installed
                          ;                 3        three RS-232 ports installed
                          ;                 4        four RS-232 ports installed
                          ;         8       ----     reserved
                          ;         7-6     0        1 floppy diskette drive installed, iff bit 0=1
                          ;                 1        2 floppy diskette drives installed, iff bit 0=1
                          ;         5-4     1        initial video mode of 40-column color
                          ;                 2        initial video mode of 80-column color
                          ;                 3        initial video mode of 80-column monochrome
                          ;         3-2     ----     reserved
                          ;         1       0        math coprocesor not present
                          ;                 1        math coprocessor present
                          ;         0       0        no diskette drives present
                          ;                 1        some number of floppy diskette drives present,
                          ;                          see bits 7-6

                           page
                          ;*************************************************************************
                          ; Keyboard LEDS S40_KBD_LED_FLAGS
                          ;*************************************************************************
                          ;                                      76543210b
        = 0004            S40E_KBD_LED_CAPS       equ      00000100b       ; caps lock LED state
        = 0002            S40E_KBD_LED_NUM        equ      00000010b       ; num lock LED state
        = 0001            S40E_KBD_LED_SCROLL     equ      00000001b       ; scroll lock LED state
                          ;*************************************************************************
                          ;
                          ;         Bit     Value    Definition
                          ;         7-3     ----     reserved
                          ;         2       0        <Caps lock> LED is off
                          ;                 1        <Caps lock> LED is on
                          ;         1       0        <Num lock> LED is on
                          ;                 1        <Num lock> LED is off
                          ;         0       0        <Scroll lock> LED is off
                          ;                 1        <Scroll lock> LED is on
                          ;
                          ;*************************************************************************

                          ;*************************************************************************
                          ; S40_KBD_STATE1
                          ;*************************************************************************
                          ;                                      76543210b
        = 0080            S40E_KBD_ST1_INSERT     equ      10000000b       ; insert mode state
        = 0040            S40E_KBD_ST1_CAPS       equ      01000000b       ; caps lock mode state
        = 0020            S40E_KBD_ST1_NUM        equ      00100000b       ; num lock mode state
        = 0010            S40E_KBD_ST1_SCROLL     equ      00010000b       ; scroll lock mode state
        = 0008            S40E_KBD_ST1_ALT        equ      00001000b       ; alt key state
        = 0004            S40E_KBD_ST1_CTRL       equ      00000100b       ; control key state
        = 0002            S40E_KBD_ST1_LSHIFT     equ      00000010b       ; left shift key state
        = 0001            S40E_KBD_ST1_RSHIFT     equ      00000001b       ; right shift key state
                          ;*************************************************************************
                          ;
                          ;         Bit     Value    Definition
                          ;         7       0        insert state inactive
                          ;                 1        insert state active
                          ;         6       0        caps lock state inactive
                          ;                 1        caps lock state active
                          ;         5       0        num lock state inactive
                          ;                 1        num lock state active
                          ;         4       0        scroll lock state inactive
                          ;                 1        scroll lock state active
                          ;         3       0        <Alt> key not depressed (inactive)
                          ;                 1        <Alt> key depressed (active)
                          ;         2       0        <CTRL> key not depressed (inactive)
                          ;                 1        <CTRL> key depressed (active)
                          ;         1       0        left <Shift> key not depressed (inactive)
                          ;                 1        left <Shift> key depressed (active)
                          ;         0       0        right <Shift> key not depressed (inactive)
                          ;                 1        right <Shift> key depressed (active)
                          ;
                          ;*************************************************************************

                          ;*************************************************************************
                          ; S40_KBD_STATE2
                          ;*************************************************************************
                          ;                                      76543210b
        = 0080            S40E_KBD_ST2_INSERT     equ      10000000b       ; insert key state
        = 0040            S40E_KBD_ST2_CAPS       equ      01000000b       ; caps lock key state
        = 0020            S40E_KBD_ST2_NUM        equ      00100000b       ; num lock key state
```

Equates File (continued)

```
= 0010                S40E_KBD_ST2_SCROLL      equ      00010000b       ; scroll lock key state
= 0008                S40E_KBD_ST2_PAUSE       equ      00001000b       ; pause (<CTRL>-<Numlock>) state
= 0004                S40E_KBD_ST2_SYSREQ      equ      00000100b       ; sys req key state
                      ;*****************************************************************************
                      ;
                      ;         Bit     Value    Definition
                      ;         7       0        <Ins> key not depressed
                      ;                 1        <Ins> key depressed
                      ;         6       0        <Caps lock> key not depressed
                      ;                 1        <Caps lock> key depressed
                      ;         5       0        <Num lock> key not depressed
                      ;                 1        <Num lock> key depressed
                      ;         4       0        <ScrLck> key not depressed
                      ;                 1        <ScrLck> key depressed
                      ;         3       0        pause state (<CTRL>-<Num lock>) inactive
                      ;                 1        pause state active
                      ;         2       0        <Sys req> key not depressed
                      ;                 1        <Sys req> key depressed
                      ;         1-0     ----     reserved
                      ;
                      ;*****************************************************************************

                      page
                      ;*****************************************************************************
                      ; S40_AFLOPPY_MEDIA0
                      ;*****************************************************************************
                      ;                                 76543210b
= 00C0                S40E_MEDIA0_RATE         equ      11000000b       ; drive 0 data transfer rate
= 0020                S40E_MEDIA0_STEP         equ      00100000b       ; drive 0 seek step flag
= 0010                S40E_MEDIA0_KNOWN        equ      00010000b       ; drive 0 media known flag
= 0007                S40E_MEDIA0_TYPE         equ      00000111b       ; drive 0 media type field
                      ;*****************************************************************************
                      ; S40_AFLOPPY_MEDIA1
                      ;*****************************************************************************
                      ;                                 76543210b
= 00C0                S40E_MEDIA1_RATE         equ      11000000b       ; drive 1 data transfer rate
= 0020                S40E_MEDIA1_STEP         equ      00100000b       ; drive 1 seek step flag
= 0010                S40E_MEDIA1_KNOWN        equ      00010000b       ; drive 1 media known flag
= 0007                S40E_MEDIA1_TYPE         equ      00000111b       ; drive 1 media type field
                      ;*****************************************************************************
                      ;         Bit     Value    Definition
                      ;         7-6     0        data transfer rate is 500kb/sec
                      ;                 1        data transfer rate is 300kb/sec
                      ;                 2        data transfer rate is 250kb/sec
                      ;         5       0        single step all seeks
                      ;                 1        double step all seeks
                      ;         4       0        type of diskette in drive unknown
                      ;                 1        type of diskette in drive known
                      ;         3       ----     reserved
                      ;         2-0     0        attempting 360k diskette in 360k drive
                      ;                 1        attempting 360k diskette in 1.2mb drive
                      ;                 2        attempting 1.2mb diskette in 1.2mb drive
                      ;                 3        determined 360k diskette in 360k drive
                      ;                 4        determined 360k diskette in 1.2mb drive
                      ;                 5        determined 1.2mb diskette in 1.2mb drive
                      ;*****************************************************************************

                      page
                      ;*****************************************************************************
                      ; S40_FLOPPY_RETURN_STAT
                      ;*****************************************************************************
                      ;                                 76543210b
= 0080                S40E_FLOPPY_RSTAT_TMO    equ      10000000b       ; timeout error flag
= 0040                S40E_FLOPPY_RSTAT_SEEK   equ      01000000b       ; seek error flag
= 0020                S40E_FLOPPY_RSTAT_CTRLR  equ      00100000b       ; controller error flag
= 001F                S40E_FLOPPY_RSTAT_ERR    equ      00011111b       ; error code field
                      ;*****************************************************************************
                      ;
                      ;        Bit     Value    Definition
                      ;         7       1        timeout error, diskette failed to respond in time
                      ;         6       1        seek error; seek to track failed
                      ;         5       1        controller error; diskette controller chip failed
                      ;         4-0     1        bad command; invalid command request
                      ;                 2        address error; address mark on diskette not found
                      ;                 3        write protect error
                      ;                 4        sector not found; unable to locate sector, diskette
                      ;                          damaged or unformatted
                      ;                 6        media changed; the drive door was opened
                      ;                          on a 1.2mb diskette drive
                      ;                 8        DMA error; DMA failed to respond in time
                      ;                 9        segment wrap; attempt to perform DMA accros
                      ;                          a segment boundary
                      ;                 10       CRC error; crc check on data failed

                      page
                      ;*****************************************************************************
                      ; S40_FLOPPY_MOTOR_STAT
                      ;*****************************************************************************
                      ;                                 76543210b
= 0080                S40E_FLOPPY_MOTR_WRITE   equ      10000000b       ; write operation flag
= 0020                S40E_FLOPPY_MOTR_SELCT1  equ      00100000b       ; drive one select flag
= 0010                S40E_FLOPPY_MOTR_SELCT0  equ      00010000b       ; drive zero select flag
= 0002                S40E_FLOPPY_MOTR_RUN1    equ      00000010b       ; drive one motor flag
= 0001                S40E_FLOPPY_MOTR_RUN0    equ      00000001b       ; drive zero motor flag
                      ;*****************************************************************************
```

Equates File (continued)

```
                              ;         Bit     Value   Definition
                              ;         7       0       current operation is not a write
                              ;                 1       current operation is a write
                              ;         6       ----    reserved
                              ;         5       0       drive one in not selected
                              ;                 1       drive one is selected
                              ;         4       0       drive zero is not selected
                              ;                 1       drive zero is selected
                              ;         3-2     ----    reserved
                              ;         1       0       drive one motor is not running
                              ;                 1       drive one motor is running
                              ;         0       0       drive zero motor is not running
                              ;                 1       drive zero motor is running
                              ;****************************************************************************

                              ;****************************************************************************
                              ; S40_FLOPPY_SEEK_STAT
                              ;****************************************************************************
                                                        76543210b
        = 0080                S40E_FLOPPY_SEEK_INT    equ     10000000b       ; interrupt occured flag
        = 0002                S40E_FLOPPY_SEEK_RECAL1 equ     00000010b       ; drive one recalibration flag
        = 0001                S40E_FLOPPY_SEEK_RECAL0 equ     00000001b       ; drive zero recalibration flag
                              ;****************************************************************************
                              ;         Bit     Value   Definition
                              ;         7       1       diskette hardware interrupt occured
                              ;         6-2     ----    reserved
                              ;         1-0     0       indicates corresponding drive (1 or 0) needs
                              ;                         recalibration before next seek
                              ;                 1       indicates corresponding drive (1 or 0) does not
                              ;                         need recalibration before next seek
                              ;****************************************************************************
                                        END
```

```
Macros:

                N a m e                 Length

MSD_HEADER . . . . . . . . . . .        0006
POPPF . . . . . . . . . . . . .         0003
SYSCALL . . . . . . . . . . . .         0004

Structures and records

                N a m e                 Width   # fields
                                        Shift   Width   Mask    Initial

DESCRIBE . . . . . . . . . . . .        0030    0018
  D_SOURCE . . . . . . . . . . .        0010
  D_HPHIL_ID . . . . . . . . . .        0011
  D_DESC_MASK . . . . . . . . .         0012
  D_IO_MASK . . . . . . . . . .         0013
  D_XDESC_MASK . . . . . . . . .        0014
  D_MAX_AXIS . . . . . . . . . .        0015
  D_CLASS . . . . . . . . . . .         0016
  D_PROMPTS . . . . . . . . . .         0017
  D_RESERVED . . . . . . . . . .        0018
  D_BURST_LEN . . . . . . . . .         0019
  D_WR_REG . . . . . . . . . . .        001A
  D_RD_REG . . . . . . . . . . .        001B
  D_TRANSITION . . . . . . . . .        001C
  D_STATE . . . . . . . . . . .         001D
  D_RESOLUTION . . . . . . . . .        001E
  D_SIZE_X . . . . . . . . . . .        0020
  D_SIZE_Y . . . . . . . . . . .        0022
  D_ABS_X . . . . . . . . . . .         0024
  D_ABS_Y . . . . . . . . . . .         0026
  D_REL_X . . . . . . . . . . .         0028
  D_REL_Y . . . . . . . . . . .         002A
  D_ACCUM_X . . . . . . . . . .         002C
  D_ACCUM_Y . . . . . . . . . .         002E
HP_GLB_HEADER . . . . . . . . .         0060    0013
  T_HP_HEADER . . . . . . . . .         0000
  T_USED_AND_RESERVED . . . . .         0002
  T_HP_LAST_DS . . . . . . . . .        000E
  T_HP_MAX_DS . . . . . . . . .         0010
  T_HP_NXT_VCTR . . . . . . . .         0012
  T_SND_FLAG . . . . . . . . . .        0014
  T_SND_CLICK_COUNT . . . . . .         0015
  T_SND_CLICK_DURA . . . . . . .        0016
  T_SND_CLICK_VOLUME . . . . . .        0017
  T_SND_BEEP_CYCLE . . . . . . .        0018
  T_SND_BEEP_DURA . . . . . . .         001A
  T_SND_BEEP_COUNT . . . . . . .        001C
  T_STR_NEXT_INDEX . . . . . . .        001E
  T_STR_ROOT . . . . . . . . . .        0020
  T_STR_VCT_HDR . . . . . . . .         0024
  T_STR_MSG_HDR . . . . . . . .         0032
  T_8259_FLAGS . . . . . . . . .        0040
HP_SHEADER . . . . . . . . . . .        0010    0009
```

Equates File (continued)

```
  DH_ATR . . . . . . . . . . . . . .      0000
  DH_NAME_INDEX. . . . . . . . . . .      0002
  DH_V_DEFAULT . . . . . . . . . . .      0004
  DH_P_CLASS . . . . . . . . . . . .      0006
  DH_C_CLASS . . . . . . . . . . . .      0008
  DH_V_PARENT. . . . . . . . . . . .      000A
  DH_V_CHILD . . . . . . . . . . . .      000C
  DH_MAJOR . . . . . . . . . . . . .      000E
  DH_MINOR . . . . . . . . . . . . .      000F
HP_TABLE_ENTRY . . . . . . . . . . .      0006     0003
  HP_ENTRY_IP. . . . . . . . . . . .      0000
  HP_ENTRY_CS. . . . . . . . . . . .      0002
  HP_ENTRY_DS. . . . . . . . . . . .      0004
LDESCRIBE. . . . . . . . . . . . . .      0030     0017
  LD_SOURCE. . . . . . . . . . . . .      0010
  LD_HPHIL_ID. . . . . . . . . . . .      0011
  LD_DEVICE_STATE. . . . . . . . . .      0012
  LD_INDEX . . . . . . . . . . . . .      0014
  LD_MAX_AXIS. . . . . . . . . . . .      0015
  LD_CLASS . . . . . . . . . . . . .      0016
  LD_PROMPTS . . . . . . . . . . . .      0017
  LD_RESERVED. . . . . . . . . . . .      0018
  LD_RES2. . . . . . . . . . . . . .      0019
  LD_RES3. . . . . . . . . . . . . .      001A
  LD_RES4. . . . . . . . . . . . . .      001B
  LD_TRANSITION. . . . . . . . . . .      001C
  LD_STATE . . . . . . . . . . . . .      001D
  LD_RESOLUTION. . . . . . . . . . .      001E
  LD_SIZE_X. . . . . . . . . . . . .      0020
  LD_SIZE_Y. . . . . . . . . . . . .      0022
  LD_ABS_X . . . . . . . . . . . . .      0024
  LD_ABS_Y . . . . . . . . . . . . .      0026
  LD_REL_X . . . . . . . . . . . . .      0028
  LD_REL_Y . . . . . . . . . . . . .      002A
  LD_ACCUM_X . . . . . . . . . . . .      002C
  LD_ACCUM_Y . . . . . . . . . . . .      002E
MSD_INIT_CMD . . . . . . . . . . . .      0017     0007
  MSD_UNIT_COUNT . . . . . . . . . .      000D
  MSD_END_OFFSET . . . . . . . . . .      000E
  MSD_END_SEG. . . . . . . . . . . .      0010
  MSD_BPB_OFFSET . . . . . . . . . .      0012
  MSD_BPB_SEG. . . . . . . . . . . .      0014
  MSD_1ST_UNIT . . . . . . . . . . .      0016
MSD_IO_CMD . . . . . . . . . . . . .      0018     0006
  MSD_XFER_OFFSET. . . . . . . . . .      000E
  MSD_XFER_SEG . . . . . . . . . . .      0010
  MSD_XFER_COUNT . . . . . . . . . .      0012
  MSD_1ST_BLK. . . . . . . . . . . .      0014
  MSD_VERR_SEG . . . . . . . . . . .      0016
MSD_REQ_HEADER . . . . . . . . . . .      0016     000A
  MSD_CMDLEN . . . . . . . . . . . .      0000
  MSD_UNIT . . . . . . . . . . . . .      0001
  MSD_CMD. . . . . . . . . . . . . .      0002
  MSD_STATUS . . . . . . . . . . . .      0003
  MSD_MEDIA. . . . . . . . . . . . .      000D
  MSD_TRANS. . . . . . . . . . . . .      000E
  MSD_COUNT. . . . . . . . . . . . .      0012
  MSD_START. . . . . . . . . . . . .      0014
SEGMENT40. . . . . . . . . . . . . .      011E     0057
  S40_RS232_PORT1_ADR. . . . . . . .      0000
  S40_RS232_PORT2_ADR. . . . . . . .      0002
  S40_RS232_PORT3_ADR. . . . . . . .      0004
  S40_RS232_PORT4_ADR. . . . . . . .      0006
  S40_PRINT_PORT1_ADR. . . . . . . .      0008
  S40_PRINT_PORT2_ADR. . . . . . . .      000A
  S40_PRINT_PORT3_ADR. . . . . . . .      000C
  S40_PRINT_PORT4_ADR. . . . . . . .      000E
  S40_EQUIPMENT_FLAG . . . . . . . .      0010
  S40_MFG_INIT . . . . . . . . . . .      0012
  S40_MEMORY_SIZE. . . . . . . . . .      0013
  S40_MFG_ERR_FLAG1. . . . . . . . .      0015
  S40_MFG_ERR_FLAG2. . . . . . . . .      0016
  S40_KBD_STATE1 . . . . . . . . . .      0017
  S40_KBD_STATE2 . . . . . . . . . .      0018
  S40_ALT_INPUT_ACCUM. . . . . . . .      0019
  S40_KBD_BUF_HEAD . . . . . . . . .      001A
  S40_KBD_BUF_TAIL . . . . . . . . .      001C
  S40_KBD_BUFFER . . . . . . . . . .      001E
  S40_FLOPPY_SEEK_STAT . . . . . . .      003E
  S40_FLOPPY_MOTOR_STAT. . . . . . .      003F
  S40_FLOPPY_TIME_OUT. . . . . . . .      0040
  S40_FLOPPY_RETURN_STAT . . . . . .      0041
  S40_FLOPPY_CONTRL_STAT . . . . . .      0042
  S40_CRT_MODE . . . . . . . . . . .      0049
  S40_CRT_WIDTH. . . . . . . . . . .      004A
  S40_CRT_LENGTH . . . . . . . . . .      004C
  S40_CRT_PAGE_ADR . . . . . . . . .      004E
  S40_CRT_CURSOR_POS . . . . . . . .      0050
  S40_CRT_CURSOR_MODE. . . . . . . .      0060
  S40_CRT_DISPLAY_PAGE . . . . . . .      0062
  S40_CRT_PORT_ADR . . . . . . . . .      0063
  S40_CRT_MODE_SEL_REG . . . . . . .      0065
  S40_CRT_PALETTE. . . . . . . . . .      0066
```

Equates File (continued)

```
  S40_XROM_INIT_ADR . . . . . . . . .      0067
  S40_XROM_SEGMENT  . . . . . . . . .      0069
  S40_XROM_INT_FLAG . . . . . . . . .      006B
  S40_TIMR_LOW  . . . . . . . . . . .      006C
  S40_TIMR_HIGH . . . . . . . . . . .      006E
  S40_TIMR_OVR_FLOW . . . . . . . . .      0070
  S40_SYS_BREAK_FLAG  . . . . . . . .      0071
  S40_SYS_RESET_FLAG  . . . . . . . .      0072
  S40_FD_STATUS . . . . . . . . . . .      0074
  S40_FD_COUNT  . . . . . . . . . . .      0075
  S40_FD_CONTROL  . . . . . . . . . .      0076
  S40_FD_PORT_OFFSET  . . . . . . . .      0077
  S40_PRINT_TIMEOUT1  . . . . . . . .      0078
  S40_PRINT_TIMEOUT2  . . . . . . . .      0079
  S40_PRINT_TIMEOUT3  . . . . . . . .      007A
  S40_PRINT_TIMEOUT4  . . . . . . . .      007B
  S40_RS232_TIMEOUT1  . . . . . . . .      007C
  S40_RS232_TIMEOUT2  . . . . . . . .      007D
  S40_RS232_TIMEOUT3  . . . . . . . .      007E
  S40_RS232_TIMEOUT4  . . . . . . . .      007F
  S40_KBD_BUF_START . . . . . . . . .      0080
  S40_KBD_BUF_END . . . . . . . . . .      0082
  S40_EGA_CRT_ROW_CNT . . . . . . . .      0084
  S40_EGA_CHAR_SIZE . . . . . . . . .      0085
  S40_EGA_INFO1 . . . . . . . . . . .      0087
  S40_EGA_INFO2 . . . . . . . . . . .      0088
  S40_FLOPPY_LAST_RATE  . . . . . . .      008B
  S40_AFD_STATUS_REG  . . . . . . . .      008C
  S40_AFD_ERROR_REG . . . . . . . . .      008D
  S40_AFD_INTR_FLAG . . . . . . . . .      008E
  S40_AFD_CTRL_FLAG . . . . . . . . .      008F
  S40_AFLOPPY_MEDIA0  . . . . . . . .      0090
  S40_AFLOPPY_MEDIA1  . . . . . . . .      0091
  S40_AFLOPPY_OPER0 . . . . . . . . .      0092
  S40_AFLOPPY_OPER1 . . . . . . . . .      0093
  S40_AFLOPPY_TRACK0  . . . . . . . .      0094
  S40_AFLOPPY_TRACK1  . . . . . . . .      0095
  S40_AFLOPPY_RESERVED  . . . . . . .      0096
  S40_KBD_LED_FLAGS . . . . . . . . .      0097
  S40_RTC_WAIT_OFFSET . . . . . . . .      0098
  S40_RTC_WAIT_SEGMENT  . . . . . . .      009A
  S40_RTC_WAIT_CNT_LOW  . . . . . . .      009C
  S40_RTC_WAIT_CNT_HIGH . . . . . . .      009E
  S40_RTC_WAIT_ACTV_FLG . . . . . . .      00A0
  S40_EGA_TBL_PTR . . . . . . . . . .      00A8
  S40_INTRA_APPL  . . . . . . . . . .      00F0
  S40_PSCRN_STATUS  . . . . . . . . .      0100
  S40_SINGLE_DRV_STAT . . . . . . . .      0104
STR_HEADER  . . . . . . . . . . . . .      000E      0005
  STR_NXT_HDR . . . . . . . . . . . .      0000
  STR_UPPER_BOUND . . . . . . . . . .      0004
  STR_LOWER_BOUND . . . . . . . . . .      0006
  STR_LIST_PTR  . . . . . . . . . . .      0008
  STR_SEGMENT . . . . . . . . . . . .      000C
VIDEO_DATA  . . . . . . . . . . . . .      0040      000D
  VID_ATR . . . . . . . . . . . . . .      0000
  VID_NAME_INDEX  . . . . . . . . . .      0002
  VID_V_DEFAULT . . . . . . . . . . .      0004
  VID_PRIMARY . . . . . . . . . . . .      0006
  VID_SECONDARY . . . . . . . . . . .      0007
  VID_FOUND_ROM . . . . . . . . . . .      0008
  VID_IDS . . . . . . . . . . . . . .      0009
  VID_STATUS  . . . . . . . . . . . .      000D
  VID_EXT_STATUS  . . . . . . . . . .      0011
  VID_PARM_BLOCK  . . . . . . . . . .      0015
  VID_LAST_IBM_MODE . . . . . . . . .      003C
  VID_EXT_MODE  . . . . . . . . . . .      003D
  VID_PADDING . . . . . . . . . . . .      003E

Symbols:

                 N a m e                   Type    Value    Attr

  ATR_0 . . . . . . . . . . . . . . .        Number   0001
  ATR_BOT . . . . . . . . . . . . . .        Number   0A00
  ATR_CSHARE  . . . . . . . . . . . .        Number   0008
  ATR_DEVCFG  . . . . . . . . . . . .        Number   4000
  ATR_ENTRY . . . . . . . . . . . . .        Number   1000
  ATR_FREE  . . . . . . . . . . . . .        Number   0200
  ATR_HP  . . . . . . . . . . . . . .        Number   8000
  ATR_IND . . . . . . . . . . . . . .        Number   0800
  ATR_INP . . . . . . . . . . . . . .        Number   0C00
  ATR_ISR . . . . . . . . . . . . . .        Number   2000
  ATR_LOG . . . . . . . . . . . . . .        Number   0600
  ATR_MAJOR . . . . . . . . . . . . .        Number   0020
  ATR_MAP_CALL  . . . . . . . . . . .        Number   0080
  ATR_MID . . . . . . . . . . . . . .        Number   0060
  ATR_MINOR . . . . . . . . . . . . .        Number   0040
  ATR_NOADDR  . . . . . . . . . . . .        Number   0000
  ATR_PSHARE  . . . . . . . . . . . .        Number   0010
  ATR_ROM . . . . . . . . . . . . . .        Number   0004
  ATR_RSVD  . . . . . . . . . . . . .        Number   0000
  ATR_SRVC  . . . . . . . . . . . . .        Number   0400
  ATR_STRING  . . . . . . . . . . . .        Number   0100
  ATR_SUBADD  . . . . . . . . . . . .        Number   0060
  ATR_TYPE7 . . . . . . . . . . . . .        Number   0E00
```

Equates File (continued)

```
ATR_TYPE_MASK . . . . . . . . . .      Number  0E00
ATR_YIELD . . . . . . . . . . . .      Number  0002
BUTTON_ERROR  . . . . . . . . . .      Number  0002
CLASS_ASCII . . . . . . . . . . .      Number  0002
CLASS_BINARY  . . . . . . . . . .      Number  0003
CLASS_GIDCCP  . . . . . . . . . .      Number  0005
CLASS_JOY . . . . . . . . . . . .      Number  0007
CLASS_KBD . . . . . . . . . . . .      Number  0000
CLASS_KEYPAD  . . . . . . . . . .      Number  000C
CLASS_MOUSE . . . . . . . . . . .      Number  0004
CLASS_PADDLE  . . . . . . . . . .      Number  0009
CLASS_TABLET  . . . . . . . . . .      Number  0006
CLASS_THUMB . . . . . . . . . . .      Number  000A
CLASS_TRACKBALL . . . . . . . . .      Number  000B
CLASS_TS  . . . . . . . . . . . .      Number  0001
CLASS_UNDEF8  . . . . . . . . . .      Number  0008
CLASS_UNDEFD  . . . . . . . . . .      Number  000D
CLASS_UNDEFE  . . . . . . . . . .      Number  000E
CLASS_UNDEFF  . . . . . . . . . .      Number  000F
CLIP_ENABLED  . . . . . . . . . .      Number  0004
CL_ALL  . . . . . . . . . . . . .      Number  FFFF
CL_ASCII  . . . . . . . . . . . .      Number  0002
CL_BLK  . . . . . . . . . . . . .      Number  0080
CL_BOOT . . . . . . . . . . . . .      Number  0040
CL_BYTE . . . . . . . . . . . . .      Number  0800
CL_CCP  . . . . . . . . . . . . .      Number  2000
CL_COMM . . . . . . . . . . . . .      Number  0400
CL_CON  . . . . . . . . . . . . .      Number  1000
CL_EXTEND . . . . . . . . . . . .      Number  0001
CL_FILT . . . . . . . . . . . . .      Number  0100
CL_GID  . . . . . . . . . . . . .      Number  0008
CL_INTERFACE  . . . . . . . . . .      Number  0200
CL_KBD  . . . . . . . . . . . . .      Number  4000
CL_KBDFC  . . . . . . . . . . . .      Number  8000
CL_LGID . . . . . . . . . . . . .      Number  0020
CL_NULL . . . . . . . . . . . . .      Number  0000
CL_PGID . . . . . . . . . . . . .      Number  0010
CL_PTS  . . . . . . . . . . . . .      Number  0004
DESCRIBE_SIZE . . . . . . . . . .      Number  0030
D_ADDR_MASK . . . . . . . . . . .      Number  000F
D_BUFFER  . . . . . . . . . . . .      Alias   D_SIZE_X
D_CCP_STATE . . . . . . . . . . .      Number  001E
D_CLASS_CURRENT . . . . . . . . .      Number  00F0
D_CLASS_DEFAULT . . . . . . . . .      Number  000F
D_REMAINDER_ACCUM . . . . . . . .      Alias   D_ACCUM_X
D_SAMPLE_ABSOLUTE . . . . . . . .      Alias   D_ABS_X
D_SAMPLE_RELATIVE . . . . . . . .      Alias   D_REL_X
D_SIZE  . . . . . . . . . . . . .      Alias   D_SIZE_X
D_TYPE_MASK . . . . . . . . . . .      Number  00F0
EVENT_ENABLED . . . . . . . . . .      Number  0010
F10_GET_INFO  . . . . . . . . . .      Number  6F01
F10_GET_RES . . . . . . . . . . .      Number  6F04
F10_GET_STMODE  . . . . . . . . .      Number  000F
F10_INQUIRE . . . . . . . . . . .      Number  6F00
F10_MOD_INFO  . . . . . . . . . .      Number  6F03
F10_RD_CHARATR  . . . . . . . . .      Number  0008
F10_RD_CURPOS . . . . . . . . . .      Number  0003
F10_RD_PENPOS . . . . . . . . . .      Number  0004
F10_RD_PIXEL  . . . . . . . . . .      Number  000D
F10_SCROLL_DN . . . . . . . . . .      Number  0007
F10_SCROLL_UP . . . . . . . . . .      Number  0006
F10_SET_CURPOS  . . . . . . . . .      Number  0002
F10_SET_CURSIZE . . . . . . . . .      Number  0001
F10_SET_INFO  . . . . . . . . . .      Number  6F02
F10_SET_MODE  . . . . . . . . . .      Number  0000
F10_SET_PAGE  . . . . . . . . . .      Number  0005
F10_SET_PALLET  . . . . . . . . .      Number  000B
F10_WRS_00  . . . . . . . . . . .      Number  1300
F10_WRS_01  . . . . . . . . . . .      Number  1301
F10_WRS_02  . . . . . . . . . . .      Number  1302
F10_WRS_03  . . . . . . . . . . .      Number  1303
F10_WR_CHARATR  . . . . . . . . .      Number  0009
F10_WR_CHARCUR  . . . . . . . . .      Number  000A
F10_WR_CHARTEL  . . . . . . . . .      Number  000E
F10_WR_PIXEL  . . . . . . . . . .      Number  000C
F10_XSET_MODE . . . . . . . . . .      Number  6F05
F13_ALT_RESET . . . . . . . . . .      Number  000D
F13_CHG_STATUS  . . . . . . . . .      Number  0016
F13_FORMAT_FLEX . . . . . . . . .      Number  0005
F13_FORMAT_HDISC  . . . . . . . .      Number  0007
F13_GET_DASD  . . . . . . . . . .      Number  0015
F13_GET_HPARMS  . . . . . . . . .      Number  0008
F13_RD_LSTATUS  . . . . . . . . .      Number  0001
F13_RD_SECTORS  . . . . . . . . .      Number  0002
F13_RESET_DISC  . . . . . . . . .      Number  0000
F13_SET_DASD  . . . . . . . . . .      Number  0017
F13_TRACK_SEEK  . . . . . . . . .      Number  000C
F13_VR_SECTORS  . . . . . . . . .      Number  0004
F13_WR_SECTORS  . . . . . . . . .      Number  0003
F14_EXINIT  . . . . . . . . . . .      Number  6F01
F14_GET_BUFFER  . . . . . . . . .      Number  6F03
F14_INIT  . . . . . . . . . . . .      Number  0000
F14_INQUIRE . . . . . . . . . . .      Number  6F00
```

Equates File (continued)

```
F14_PUT_BUFFER . . . . . . . . . .     Number  6F02
F14_RECV . . . . . . . . . . . . .     Number  0002
F14_STATUS . . . . . . . . . . . .     Number  0003
F14_TRM_BUFFER . . . . . . . . . .     Number  6F04
F14_XMIT . . . . . . . . . . . . .     Number  0001
F15_BLOCK_MOVE . . . . . . . . . .     Number  0087
F15_DEVICE_CLOSE . . . . . . . . .     Number  0081
F15_DEVICE_OPEN. . . . . . . . . .     Number  0080
F15_DEV_BUSY . . . . . . . . . . .     Number  008A
F15_ENTER_PROT . . . . . . . . . .     Number  0089
F15_GET_XMEM_SIZE. . . . . . . . .     Number  0088
F15_INT_COMPLETE . . . . . . . . .     Number  008B
F15_JOYSTICK . . . . . . . . . . .     Number  0084
F15_PROG_TERM. . . . . . . . . . .     Number  0082
F15_SYS_REQ. . . . . . . . . . . .     Number  0085
F15_WAIT . . . . . . . . . . . . .     Number  0086
F15_WAIT_EVENT . . . . . . . . . .     Number  0083
F16_DEF_ATTR . . . . . . . . . . .     Number  6F01
F16_DEF_MAPPING. . . . . . . . . .     Number  6F04
F16_GET_ATTR . . . . . . . . . . .     Number  6F02
F16_GET_KEY. . . . . . . . . . . .     Number  0000
F16_GET_MAPPING. . . . . . . . . .     Number  6F05
F16_INQUIRE. . . . . . . . . . . .     Number  6F00
F16_KBD. . . . . . . . . . . . . .     Number  6F08
F16_KBD_RESET. . . . . . . . . . .     Number  6F09
F16_KEY_STATE. . . . . . . . . . .     Number  0002
F16_SET_ATTR . . . . . . . . . . .     Number  6F03
F16_SET_MAPPING. . . . . . . . . .     Number  6F06
F16_SET_XLATORS. . . . . . . . . .     Number  6F07
F16_STATUS . . . . . . . . . . . .     Number  0001
F17_GET_BUFFER . . . . . . . . . .     Number  6F03
F17_INIT . . . . . . . . . . . . .     Number  0001
F17_INQUIRE. . . . . . . . . . . .     Number  6F00
F17_PUT_BUFFER . . . . . . . . . .     Number  6F02
F17_PUT_CHAR . . . . . . . . . . .     Number  0000
F17_READ_STATUS. . . . . . . . . .     Number  6F01
F17_STATUS . . . . . . . . . . . .     Number  0002
F17_TRM_BUFFER . . . . . . . . . .     Number  6F04
F1A_GET_DATE . . . . . . . . . . .     Number  0004
F1A_GET_RTC. . . . . . . . . . . .     Number  0002
F1A_RD_CLK_CNT . . . . . . . . . .     Number  0000
F1A_RESET_ALARM. . . . . . . . . .     Number  0007
F1A_SET_ALARM. . . . . . . . . . .     Number  0006
F1A_SET_CLK_CNT. . . . . . . . . .     Number  0001
F1A_SET_DATE . . . . . . . . . . .     Number  0005
F1A_SET_RTC. . . . . . . . . . . .     Number  0003
F33_COND_OFF . . . . . . . . . . .     Number  0010
F33_DISABLE. . . . . . . . . . . .     Number  0002
F33_DISABLE_LIGHT. . . . . . . . .     Number  000E
F33_ENABLE . . . . . . . . . . . .     Number  0001
F33_ENABLE_LIGHT . . . . . . . . .     Number  000D
F33_GRAPH_CURSOR . . . . . . . . .     Number  0009
F33_INQUIRE. . . . . . . . . . . .     Number  6F00
F33_INSTALL. . . . . . . . . . . .     Number  0000
F33_MOTION . . . . . . . . . . . .     Number  000B
F33_PUT_CURSOR . . . . . . . . . .     Number  0004
F33_RATIO. . . . . . . . . . . . .     Number  000F
F33_REPORT_DATA. . . . . . . . . .     Number  0003
F33_REPORT_PRESS . . . . . . . . .     Number  0005
F33_REPORT_RELEASE . . . . . . . .     Number  0006
F33_SET_HORIZ. . . . . . . . . . .     Number  0007
F33_SET_USR. . . . . . . . . . . .     Number  000C
F33_SET_VERT . . . . . . . . . . .     Number  0008
F33_SPEED. . . . . . . . . . . . .     Number  0013
F33_TEXT_CURSOR. . . . . . . . . .     Number  000A
F33_XTEND_GCSR . . . . . . . . . .     Number  0012
F_CMOS_GET . . . . . . . . . . . .     Number  0022
F_CMOS_RET . . . . . . . . . . . .     Number  0024
F_DEF_MASKS. . . . . . . . . . . .     Number  000A
F_GET_BLOCK. . . . . . . . . . . .     Alias   F_GET_BUFFER
F_GET_BUFFER . . . . . . . . . . .     Number  000C
F_GET_BYTE . . . . . . . . . . . .     Number  0008
F_GET_WORD . . . . . . . . . . . .     Number  0010
F_INQUIRE. . . . . . . . . . . . .     Number  0006
F_INQUIRE_ALL. . . . . . . . . . .     Number  0008
F_INQUIRE_FIRST. . . . . . . . . .     Number  000A
F_INS_BASEHPVT . . . . . . . . . .     Number  0004
F_INS_FIND . . . . . . . . . . . .     Number  0018
F_INS_FIXGETDS . . . . . . . . . .     Number  000E
F_INS_FIXGLBDS . . . . . . . . . .     Number  0010
F_INS_FIXOWNDS . . . . . . . . . .     Number  000C
F_INS_FREEGETDS. . . . . . . . . .     Number  0014
F_INS_FREEGLBDS. . . . . . . . . .     Number  0016
F_INS_FREEOWNDS. . . . . . . . . .     Number  0012
F_INS_XCHGFIX. . . . . . . . . . .     Number  0006
F_INS_XCHGFREE . . . . . . . . . .     Number  000A
F_INS_XCHGRSVD . . . . . . . . . .     Number  0008
F_IO_CONTROL . . . . . . . . . . .     Number  0004
F_ISR. . . . . . . . . . . . . . .     Number  0000
F_PUT_BLOCK. . . . . . . . . . . .     Alias   F_PUT_BUFFER
F_PUT_BUFFER . . . . . . . . . . .     Number  000A
F_PUT_BYTE . . . . . . . . . . . .     Number  0006
F_PUT_SPRITE . . . . . . . . . . .     Number  0010
F_PUT_WORD . . . . . . . . . . . .     Number  000E
```

Equates File (continued)

F_RAM_GET	Number	001E
F_RAM_RET	Number	0020
F_REMOVE_SPRITE	Number	0012
F_REPORT_ENTRY	Number	000C
F_SAMPLE	Number	0006
F_SET_LIMITS_X	Number	000C
F_SET_LIMITS_Y	Number	000E
F_SND_BEEP	Number	003A
F_SND_BEEP_DISABLE	Number	0038
F_SND_BEEP_ENABLE	Number	0036
F_SND_CLICK	Number	0034
F_SND_CLICK_DISABLE	Number	0032
F_SND_CLICK_ENABLE	Number	0030
F_SND_SET_BEEP	Number	003C
F_SND_TONE	Number	003E
F_STR_DEL_BUCKET	Number	0042
F_STR_GET_FREE_INDEX	Number	0040
F_STR_GET_INDEX	Number	0048
F_STR_GET_STRING	Number	0046
F_STR_PUT_BUCKET	Number	0044
F_SYSTEM	Number	0002
F_TRACK_INIT	Number	0004
F_TRACK_OFF	Number	0008
F_TRACK_ON	Number	0006
F_YIELD	Number	002A
GID_A08	Number	0002
GID_A16	Number	0003
GID_R08	Number	0000
GID_R16	Number	0001
GID_UNDEF	Number	000F
HP_ENTRY	Number	006F
INDX_BARCODE	Number	0825
INDX_BOOT_ERROR_MSG	Number	0820
INDX_DRIVE_A	Number	0800
INDX_DRIVE_B	Number	0801
INDX_DRIVE_C	Number	0802
INDX_DRIVE_D	Number	0803
INDX_DRIVE_E	Number	0804
INDX_DRIVE_F	Number	0805
INDX_DRIVE_G	Number	0806
INDX_DRIVE_H	Number	0807
INDX_DRIVE_I	Number	0808
INDX_DRIVE_J	Number	0809
INDX_DRIVE_K	Number	080A
INDX_DRIVE_L	Number	080B
INDX_DRIVE_M	Number	080C
INDX_DRIVE_N	Number	080D
INDX_DRIVE_O	Number	080E
INDX_DRIVE_P	Number	080F
INDX_DRIVE_Q	Number	0810
INDX_DRIVE_R	Number	0811
INDX_DRIVE_S	Number	0812
INDX_DRIVE_T	Number	0813
INDX_DRIVE_U	Number	0814
INDX_DRIVE_V	Number	0815
INDX_DRIVE_W	Number	0816
INDX_DRIVE_X	Number	0817
INDX_DRIVE_Y	Number	0818
INDX_DRIVE_Z	Number	0819
INDX_HP_COPYRIGHT	Number	081A
INDX_INVALID_ROM_MSG	Number	081D
INDX_KEYBOARD	Number	0824
INDX_KNOB	Number	0826
INDX_KYB_LOCKED_MSG	Number	081E
INDX_MOUSE	Number	0823
INDX_RETRY_MSG	Number	081C
INDX_SETUP_MSG	Number	081B
INDX_STRIKE_F1_MSG	Number	081F
INDX_TABLET	Number	0822
INDX_TOUCH	Number	0821
INT_8041_OBF	Number	0069
INT_BOOT	Number	0019
INT_BREAKPOINT	Number	0003
INT_BREAK_EVENT	Number	001B
INT_CLOCK	Number	001A
INT_DISC	Number	0013
INT_DIVIDE_ZERO	Number	0000
INT_DOS	Number	0021
INT_EQUIPMENT	Number	0011
INT_FLOPPY_DIRECT	Number	0040
INT_FLOPPY_PARMS	Number	001E
INT_GRAPHICS_CHAR	Number	001F
INT_HDISC_PARMS0	Number	0041
INT_HDISC_PARMS1	Number	0046
INT_HPHIL	Number	006C
INT_HPMOUSE	Number	0033
INT_IRQ0_TIMER	Number	0008
INT_IRQ1_KBD_ISR	Number	0009
INT_IRQ2	Number	000A
INT_IRQ3_SERIAL1	Number	000B
INT_IRQ4_SERIAL0	Number	000C
INT_IRQ5_PRN1	Number	000D
INT_IRQ6_FLOPPY	Number	000E

Equates File (continued)

INT_IRQ7_PRNO	Number	000F
INT_KBD	Number	0016
INT_MEM_SIZE	Number	0012
INT_NMI	Number	0002
INT_OVERFLOW	Number	0004
INT_PRINTER	Number	0017
INT_PRINT_SCREEN	Number	0005
INT_RTC_EVENT	Number	004A
INT_SERIAL	Number	0014
INT_SINGLE_STEP	Number	0001
INT_SVC_REQUEST	Number	0068
INT_SYSTEM	Number	0015
INT_TIMER_TICK	Number	001C
INT_VIDEO	Number	0010
INT_VIDEO_PARMS	Number	001D
IRQ10	Number	0072
IRQ11	Number	0073
IRQ12	Number	0074
IRQ13_287	Number	0075
IRQ14_HDISC	Number	0076
IRQ15	Number	0077
IRQ8_RTC	Number	0070
IRQ9_REDIRECT	Number	0071
ISR_IN_PROGRESS	Number	0001
LDESCRIBE_SIZE	Number	0030
LD_BUFFER	Alias	LD_RESOLUTION
LD_CLASS_CURRENT	Number	00F0
LD_CLASS_DEFAULT	Number	000F
LD_REMAINDER_ACCUM	Alias	LD_ACCUM_X
LD_RES_MASK	Number	000F
LD_SAMPLE_ABSOLUTE	Alias	LD_ABS_X
LD_SAMPLE_RELATIVE	Alias	LD_REL_X
LD_SIZE	Alias	LD_SIZE_X
LD_TYPE_MASK	Number	00F0
MSD_BAD_DCHG	Number	000D
MSD_BAD_LENGTH	Number	0005
MSD_BLD_BPB	Number	0002
MSD_CRC_ERROR	Number	0004
MSD_DEV_CLOSE	Number	000E
MSD_DEV_OPEN	Number	000D
MSD_DONE_STATUS	Number	0001
MSD_ERR_STATUS	Number	0081
MSD_GEN_FAILURE	Number	000C
MSD_INIT	Number	0000
MSD_INPUT	Number	0004
MSD_IN_FLUSH	Number	0007
MSD_IN_NOWAIT	Number	0005
MSD_IN_STATUS	Number	0006
MSD_IOCTL_IN	Number	0003
MSD_IOCTL_OUT	Number	000C
MSD_MEDIA_CHK	Number	0001
MSD_NOT_READY	Number	0002
MSD_OUTPUT	Number	0008
MSD_OUT_FLUSH	Number	000B
MSD_OUT_STATUS	Number	000A
MSD_OUT_VERIFY	Number	0009
MSD_PAPER_OUT	Number	0009
MSD_READ_FAULT	Number	000B
MSD_REM_MEDIA	Number	000F
MSD_SEC_NOTFND	Number	0008
MSD_SEEK_ERROR	Number	0006
MSD_UNKNOWN_CMD	Number	0003
MSD_UNKNOWN_MEDIA	Number	0007
MSD_UNKNOWN_UNIT	Number	0001
MSD_WRITE_FAULT	Number	000A
MSD_WRITE_PROTECT	Number	0000
RS_BAD_PARAMETER	Number	00FA
RS_BREAK	Number	000C
RS_BUSY	Number	00F8
RS_DATA_NREADY	Number	000A
RS_DONE	Number	0006
RS_FAIL	Number	00FE
RS_FRAME	Number	00EE
RS_NOT_SERVICED	Number	0004
RS_NO_VECTOR	Number	00F6
RS_OFFLINE	Number	00F4
RS_OUT_OF_PAPER	Number	00F2
RS_OVERRUN	Number	0008
RS_PARITY	Number	00F0
RS_SUCCESSFUL	Number	0000
RS_TIMEOUT	Number	00FC
RS_UNSUPPORTED	Number	0002
S40E_DEVICE_BOOT	Number	0001
S40E_DEVICE_FLOPPY	Number	00C0
S40E_DEVICE_MATH	Number	0002
S40E_DEVICE_PRINTRS	Number	C000
S40E_DEVICE_RS232	Number	0E00
S40E_DEVICE_VIDEO	Number	0030
S40E_FLOPPY_MOTR_RUN0	Number	0001
S40E_FLOPPY_MOTR_RUN1	Number	0002
S40E_FLOPPY_MOTR_SELCT0	Number	0010
S40E_FLOPPY_MOTR_SELCT1	Number	0020
S40E_FLOPPY_MOTR_WRITE	Number	0080
S40E_FLOPPY_RSTAT_CTRLR	Number	0020

Equates File (continued)

```
S40E_FLOPPY_RSTAT_ERR. . . . . .        Number  001F
S40E_FLOPPY_RSTAT_SEEK . . . . .        Number  0040
S40E_FLOPPY_RSTAT_TMO. . . . . .        Number  0080
S40E_FLOPPY_SEEK_INT . . . . . .        Number  0080
S40E_FLOPPY_SEEK_RECAL0. . . . .        Number  0001
S40E_FLOPPY_SEEK_RECAL1. . . . .        Number  0002
S40E_KBD_LED_CAPS. . . . . . . .        Number  0004
S40E_KBD_LED_NUM . . . . . . . .        Number  0002
S40E_KBD_LED_SCROLL. . . . . . .        Number  0001
S40E_KBD_ST1_ALT . . . . . . . .        Number  0008
S40E_KBD_ST1_CAPS. . . . . . . .        Number  0040
S40E_KBD_ST1_CTRL. . . . . . . .        Number  0004
S40E_KBD_ST1_INSERT. . . . . . .        Number  0080
S40E_KBD_ST1_LSHIFT. . . . . . .        Number  0002
S40E_KBD_ST1_NUM . . . . . . . .        Number  0020
S40E_KBD_ST1_RSHIFT. . . . . . .        Number  0001
S40E_KBD_ST1_SCROLL. . . . . . .        Number  0010
S40E_KBD_ST2_CAPS. . . . . . . .        Number  0040
S40E_KBD_ST2_INSERT. . . . . . .        Number  0080
S40E_KBD_ST2_NUM . . . . . . . .        Number  0020
S40E_KBD_ST2_PAUSE . . . . . . .        Number  0008
S40E_KBD_ST2_SCROLL. . . . . . .        Number  0010
S40E_KBD_ST2_SYSREQ. . . . . . .        Number  0004
S40E_MEDIA0_KNOWN. . . . . . . .        Number  0010
S40E_MEDIA0_RATE . . . . . . . .        Number  00C0
S40E_MEDIA0_STEP . . . . . . . .        Number  0020
S40E_MEDIA0_TYPE . . . . . . . .        Number  0007
S40E_MEDIA1_KNOWN. . . . . . . .        Number  0010
S40E_MEDIA1_RATE . . . . . . . .        Number  00C0
S40E_MEDIA1_STEP . . . . . . . .        Number  0020
S40E_MEDIA1_TYPE . . . . . . . .        Number  0007
S40_AFLOPPY_MEDIA. . . . . . . .        E BYTE  0090
S40_AFLOPPY_OPER . . . . . . . .        E BYTE  0092
S40_AFLOPPY_TRACK. . . . . . . .        E BYTE  0094
S40_PRINT_PORT_TBL . . . . . . .        E WORD  0008
S40_PRINT_TIMEOUT_TBL. . . . . .        E BYTE  0078
S40_RS232_PORT_TBL . . . . . . .        E WORD  0000
S40_RS232_TIMEOUT_TBL. . . . . .        E BYTE  007C
SF_CLIPPING_OFF. . . . . . . . .        Number  0010
SF_CLIPPING_ON . . . . . . . . .        Number  000E
SF_CLOSE . . . . . . . . . . . .        Number  0010
SF_CLR_CRTSW . . . . . . . . . .        Number  0018
SF_CLR_RAMSW . . . . . . . . . .        Number  0014
SF_CREATE_EVENT. . . . . . . . .        Number  0008
SF_CREAT_INTR. . . . . . . . . .        Number  000A
SF_CRV_CRV_MAJ_MIN . . . . . . .        Number  0004
SF_CRV_DISABLE_REPEAT. . . . . .        Text    000E
SF_CRV_RECONFIGURE . . . . . . .        Number  0006
SF_CRV_REPEAT. . . . . . . . . .        Text    000C
SF_CRV_REPORT_NAME . . . . . . .        Number  000E
SF_CRV_REPORT_STATUS . . . . . .        Number  000C
SF_CRV_SELF_TEST . . . . . . . .        Number  000A
SF_CRV_WR_ACK. . . . . . . . . .        Text    000A
SF_CRV_WR_PROMPTS. . . . . . . .        Number  0008
SF_DEF_ATTR. . . . . . . . . . .        Number  0008
SF_DEF_LINKS . . . . . . . . . .        Number  0000
SF_DELET_INTR. . . . . . . . . .        Number  000C
SF_DISABLE_HPHIL . . . . . . . .        Number  000A
SF_DISABLE_KBD . . . . . . . . .        Number  0006
SF_DISABLE_SVC . . . . . . . . .        Number  0002
SF_DISBL_INTR. . . . . . . . . .        Number  0010
SF_ENABLE_HPHIL. . . . . . . . .        Number  0008
SF_ENABLE_KBD. . . . . . . . . .        Number  0004
SF_ENABLE_SVC. . . . . . . . . .        Number  0000
SF_ENABL_INTR. . . . . . . . . .        Number  000E
SF_EVENT_OFF . . . . . . . . . .        Number  000C
SF_EVENT_ON. . . . . . . . . . .        Number  000A
SF_GET_ATTR. . . . . . . . . . .        Number  000A
SF_GET_LINKS . . . . . . . . . .        Number  0002
SF_INIT. . . . . . . . . . . . .        Number  0000
SF_INTERVAL. . . . . . . . . . .        Number  0014
SF_KEYBOARD_LED. . . . . . . . .        Number  0012
SF_KEYBOARD_REPEAT . . . . . . .        Number  0010
SF_LOCK. . . . . . . . . . . . .        Number  0000
SF_MOUSE_COM . . . . . . . . . .        Number  0000
SF_MOUSE_OVERRIDE. . . . . . . .        Number  0002
SF_OPEN. . . . . . . . . . . . .        Number  000E
SF_PASS_THRU . . . . . . . . . .        Number  001A
SF_REPORT_STATE. . . . . . . . .        Number  0004
SF_SET_ATTR. . . . . . . . . . .        Number  000C
SF_SET_CRTSW . . . . . . . . . .        Number  0016
SF_SET_LINKS . . . . . . . . . .        Number  0004
SF_SET_RAMSW . . . . . . . . . .        Number  0012
SF_START . . . . . . . . . . . .        Number  0002
SF_TEST. . . . . . . . . . . . .        Number  0016
SF_TIMEOUT . . . . . . . . . . .        Number  0012
SF_TRACK_OFF . . . . . . . . . .        Number  0006
SF_TRACK_ON. . . . . . . . . . .        Number  0004
SF_UNLOCK. . . . . . . . . . . .        Number  0002
SF_VERSION_DESC. . . . . . . . .        Number  0006
SF_VID_GET_INFO. . . . . . . . .        Number  0002
SF_VID_GET_RES . . . . . . . . .        Number  0008
SF_VID_ID_HP . . . . . . . . . .        Number  0000
SF_VID_MOD_INFO. . . . . . . . .        Number  0006
```

Equates File (continued)

```
SF_VID_SET_INFO. . . . . . . . .          Number  0004
SF_VID_SET_MODE. . . . . . . . .          Number  000A
TRACK_ENABLED. . . . . . . . . .          Number  0008
T_ABS08. . . . . . . . . . . . .          Number  0042
T_ABS16. . . . . . . . . . . . .          Number  0043
T_GID. . . . . . . . . . . . . .          Number  0040
T_KC_ASCII . . . . . . . . . . .          Number  0002
T_KC_BUTTON. . . . . . . . . . .          Number  0009
T_KC_HPHIL_ENVOY . . . . . . . .          Number  0007
T_KC_HP_CCP. . . . . . . . . . .          Number  000D
T_KC_HP_SOFTKEY. . . . . . . . .          Number  000B
T_KC_IBM_AT. . . . . . . . . . .          Number  0008
T_KC_IBM_PC. . . . . . . . . . .          Number  000A
T_KC_IS_FUNCTION . . . . . . . .          Number  000C
T_KC_ITF . . . . . . . . . . . .          Number  0004
T_KC_NUMPAD. . . . . . . . . . .          Number  000F
T_KC_QWERTY. . . . . . . . . . .          Number  000E
T_KC_R0. . . . . . . . . . . . .          Number  0000
T_KC_R1. . . . . . . . . . . . .          Number  0001
T_KC_R3. . . . . . . . . . . . .          Number  0003
T_KC_R5. . . . . . . . . . . . .          Number  0005
T_KC_WILD. . . . . . . . . . . .          Number  0006
T_MOUSE. . . . . . . . . . . . .          Number  0044
T_POINTER. . . . . . . . . . . .          Number  0047
T_REL08. . . . . . . . . . . . .          Number  0040
T_REL16. . . . . . . . . . . . .          Number  0041
T_STATE. . . . . . . . . . . . .          Number  0020
T_STRING . . . . . . . . . . . .          Number  0010
T_TABLET . . . . . . . . . . . .          Number  0046
T_TS . . . . . . . . . . . . . .          Number  0045
T_UNKNOWN. . . . . . . . . . . .          Number  004F
VID_BLOCK_SIZE . . . . . . . . .          Number  0027
V_8041 . . . . . . . . . . . . .          Number  00AE
V_CCP. . . . . . . . . . . . . .          Number  004E
V_CCPCUR . . . . . . . . . . . .          Number  008A
V_CCPGID . . . . . . . . . . . .          Number  00A2
V_CCPNUM . . . . . . . . . . . .          Number  0096
V_DOLITTLE . . . . . . . . . . .          Number  0006
V_EVENT_POINTER. . . . . . . . .          Number  006C
V_EVENT_TABLET . . . . . . . . .          Number  0066
V_EVENT_TOUCH. . . . . . . . . .          Number  0060
V_FUNCTION . . . . . . . . . . .          Number  0042
V_HPHIL. . . . . . . . . . . . .          Number  0114
V_LHPMOUSE . . . . . . . . . . .          Number  00CC
V_LNULL. . . . . . . . . . . . .          Number  0108
V_LPOINTER . . . . . . . . . . .          Number  00C0
V_LTABLET. . . . . . . . . . . .          Number  00BA
V_LTOUCH . . . . . . . . . . . .          Number  00C6
V_NUMPAD . . . . . . . . . . . .          Number  0048
V_OFF. . . . . . . . . . . . . .          Number  009C
V_PGID_CCP . . . . . . . . . . .          Number  00B4
V_PNULL. . . . . . . . . . . . .          Number  000C
V_QWERTY . . . . . . . . . . . .          Number  0036
V_RAW. . . . . . . . . . . . . .          Number  0090
V_S8259. . . . . . . . . . . . .          Number  001E
V_SCOPY. . . . . . . . . . . . .          Number  0000
V_SINPUT . . . . . . . . . . . .          Number  002A
V_SKEY2FKEY. . . . . . . . . . .          Number  00A8
V_SOFTKEY. . . . . . . . . . . .          Number  003C
V_STRACK . . . . . . . . . . . .          Number  005A
V_SVIDEO . . . . . . . . . . . .          Number  0054
V_SYSTEM . . . . . . . . . . . .          Number  0012

4862 Bytes free

Warning Severe
Errors  Errors
0       0
```

APPENDIX F

F. DEFAULT DEVICE MAPPING

The following table describes the device mappings which are setup during SYSGEN. The default mapping device is listed first. Other mappable devices are listed following the default device.

Input System

Physical Device	Logical Device	Mappable Driver
Mouse	Cursor Control Pad	V_PGID_CCP V_LHPMOUSE V_LPOINTER V_LTOUCH V_LTABLET
Rotary Knob	Cursor Control Pad	V_PGID_CCP V_LHPMOUSE V_LPOINTER V_LTOUCH V_LTABLET
Touch Screen	Touch Screen	V_LTOUCH V_LHPMOUSE V_LPOINTER V_PGID_CCP V_LTABLET
Tablet	Tablet	V_LTABLET V_LHPMOUSE V_LPOINTER V_LTOUCH V_PGID_CCP
Keyboard	Keyboard Subsystem	V_8041

Keyboard Subsystem

Keypad	Translator Service	Mappable Driver
Function keys →	V__FUNCTION →	non-mappable
HP softkeys →	V__SOFTKEY →	SKEY2FKEY V__OFF V__RAW
QWERTY Pad →	V__QWERTY →	non-mappable
Numeric Pad →	V__NUMPAD →	non-mappable
Cursor Control Pad	V__CCP →	V__CCPNUM V__LHPMOUSE V__OFF V__RAW V__CCPGID (if installed)

Discs

DISC A:	Flexible Disc 0 Upper Drive
DISC B:	Flexible Disc 1 Lower Drive
DISC C:	Internal Hard Disc
DISC D:	External Disc
DISC E:	RAM disc

Discs on the system are only mappable using ASSIGN.COM.

Character I/O Devices

COM1:	Serial Port 0
COM2:	Serial Port 1
LPT1: or PRN:	Parallel Port 0
LPT2:	Parallel Port 1
LPT3:	Parallel Port 2

These ports are only mappable using MODE.COM.

APPENDIX G

G. DRIVER WRITER'S GUIDE

This appendix describes how a programmer can add drivers to the ROM BIOS. One of the important features of the EX-BIOS is the ease with which it can be expanded. This capability allows programmers to take full advantage of HP system components (such as the HP-HIL touch screen, mouse, tablet, etc.). In addition, the EX-BIOS architecture provides a simple, yet powerful way to integrate OEM and third-party products into the system.

G.1 Who Should Read This Appendix

This appendix is intended for all programmers and advanced users who wish to utilize EX-BIOS capabilities not supported by system software. It assumes that the reader is familiar with the contents of Sections 1 through 10, iAPX286 programming, DOS concepts in general, and DOS installable device drivers in particular. The reader should consult the publications listed under the References section at the end of this manual for additional information on these topics.

G.2 Introduction

This appendix presents two examples of how drivers that interface to the system's EX-BIOS can be written. All aspects of how a driver gets connected and used through the EX-BIOS are discussed.

The typical steps involved in connecting a driver into the EX-BIOS are:

- A driver added to the system can be one of three types: ROM driver, MS-DOS installable device driver or MS-DOS command that executes and stays resident.
- The driver gets called to initialize. At this point the driver will determine what machine it is executing on, obtain memory for its data segment, get an EX-BIOS vector address assigned and be added to the HP__VECTOR__TABLE.
- Any time after initialization the driver can respond to service requests in two ways. It responds to a hardware service request when it is called with its F__ISR (AH = 0) function or it responds to an application service request when it is called with any other driver specific function.

The above sequence is a general description of a driver's lifecycle. It is not necessary that all drivers follow the same steps. The sections below outline what are the necessary elements of an EX-BIOS driver.

Note

> For a detailed explanation of the calls to V__SYSTEM used below see Section 9.

G.3 Installation of Device Drivers

Each type of device driver is installed in a different manner depending on how it is brought into the system. There are three ways that an EX-BIOS driver can be installed in the system. An I/O adapter card can have an EX-BIOS driver which can be installed in the system when the adapter's ROM gets called to initialize. The adapter's initialization routines can use all of the V__SYSTEM functions to properly connect the driver. Note that because the adapter's code modules are initialized during the system generation process (SYSGEN), an EX-BIOS driver on an adapter card can not depend on other EX-BIOS drivers already being present and initialized (V__SYSTEM is the only driver usable at this point).

An MS-DOS installable device driver can also install an EX-BIOS driver. The driver must have two interfaces, one driver interface for MS-DOS and one driver interface for the EX-BIOS functions. This type of EX-BIOS driver can use all other EX-BIOS drivers already present in the system.

Finally, an MS-DOS command that stays resident can also be used to install an EX-BIOS driver. This driver can use all previously installed EX-BIOS drivers. This is the preferred method of installing EX-BIOS drivers since it only requires the EX-BIOS driver interface and functions.

G.4 Initialization

This section covers the possible steps the driver must take to insure proper initialization.

G.4.1 Product Identification

This section discusses several methods available through ROM BIOS functions for software to determine whether its host is an HP Vectra.

HP Vectra Feature/Revision Identification (V_SCOPY):

The V_SCOPY (00H) vector entry has a data segment (DS) that points to the system's copyright string. The driver can look at this string to determine if the machine is an HP Vectra. The following example illustrates how to get this string:

```
MOV   BP, V_SCOPY      ; Call the COPYRIGHT vector
PUSH  DS               ; which will set the DS and return
INT   HP_ENTRY
PUSH  DS               ; Save DS of copyright string
POP   ES               ; in ES. ES:0 is address of string
POP   DS               ; Recover old DS.
```

HP Vectra Indicator Word, Revision Word, and Date Codes

At ROM address 0F00F8H the HP Vectra has the following data.

```
DW    'HP'
DW     0000
DW     Revision_code
DW     Date_code        ; Byte 0 = year, byte 1 = week
```

This code can be used to discern the HP Vectra from other industry standard products and thus take advantage of the unique features of the HP Vectra. This method is not the preferred method.

STD-BIOS Extension Functions

The STD-BIOS Functions Fnn__INQUIRE (6F00H) indicate to the calling application that STD-BIOS extension functions are loaded and have not been replaced. The STD-BIOS drivers listed in table G.1 below support this function.

Table G.1

STD-BIOS Drivers That Support Fnn__INQUIRE

Interrupt	Function
INT 10	VIDEO
INT 14	SERIAL
INT 16	KEYBOARD
INT 17	PRINTER

To find out if the STD-BIOS extensions for the Video driver are in place use the following code:

```
        MOV   AX, F10__INQUIRE               ; Call video function (6F00)
        MOV   BX, 0FFFFH                     ; Make sure BX <> 'HP'
        INT   INT__VIDEO                     ; Interrupt 10H
        CMP   BX, 'HP'                       ; Are video extensions present?
        JE    VIDEO__EXTENSIONS__PRESENT
VIDEO__EXTENSIONS__NOT__PRESENT:


VIDEO__EXTENSIONS__PRESENT:
```

G.4.2 Obtaining Memory From the EX-BIOS

The system allows EX-BIOS drivers to obtain limited amounts of memory independent of the operating system. This feature is especially important for I/O ROM adapters since their cost can be reduced if they do not require dedicated RAM. When the I/O ROM module is initialized, it can ask for EX-BIOS memory.

This feature of the EX-BIOS system can also be utilized by application programs and system software. Any program needing a limited amount of RAM outside the operating system domain can obtain this from the EX-BIOS system.

The functions F__RAM__GET and F__RAM__RET in the V__SYSTEM driver can be used to manipulate the EX-BIOS free memory. The driver can also use the installation functions F__INS__FREEGETDS or F__INS__FIXGETDS to obtain free memory. See Section 9 for more details of these functions.

G.4.3 Getting a Free Vector

In order for an application to access an EX-BIOS driver it must call the driver through the HP__VECTOR__TABLE. Thus, each driver must request a free vector from this table.

To get a free vector from the HP__VECTOR__TABLE, a driver can use the function F__INS__XCHGFREE, F__INS__FREEOWNDS, F__INS__FREEGETDS or F__INS__FREEGLBDS in the V__SYSTEM driver. Each of these functions installs the driver at the next available free vector.

Once the driver has a vector address installed in the table, an application can call the driver by loading BP with the vector address of the driver and doing an HP__ENTRY interrupt (6FH).

G.5 EX-BIOS Driver Functions

EX-BIOS drivers support a standard set of functions and subfunctions. Nine standard function codes are defined, and several of these functions have subfunctions defined within them. These functions and subfunctions are summarized in table G.2. A detailed description of each defined function and subfunction follows.

If a driver receives a function it does not implement, it must return a status code of RS__UNSUPPORTED (02H) in the AH register. This lets the application know that the driver has not handled this function, but that it can continue if it is appropriate. This protocol frees the driver from having to implement all the defined functions and allows applications to call drivers in a consistent way.

If a driver receives a function code that it does not implement, it may also "delegate" the function to another driver. A driver may be written so that it calls another driver when it receives an unimplemented function or subfunction request.

Programmers may write drivers that implement functions and subfunctions that are not defined. However, two guidelines should be observed when defining additional functions or subfunctions. First, whenever possible, newly defined function or subfunction numbers should not conflict with existing numbers. Secondly, function and subfunction numbers should be consistent between drivers of the same class.

Table G.2

EX-BIOS Driver Function Code Summary

Function Subfunction	Register AH	 AL	Definition
F__ISR	00		Responds to a logical Interrupt Service Request (ISR).
F__SYSTEM			Executes one of several standard subfunctions.
SF__INIT*	02	00	Starts the initialization of a driver.
SF__START*	02	02	Completes the initialization process of the driver.
SF__REPORT__STATE	02	04	Reports the state of the driver.
SF__VERSION__DESC*	02	06	Reports the revision number and datecode of the driver.
SF__DEF__ATTR	02	08	Reports the default configuration of the driver.
SF__GET__ATTR	02	0A	Reports the current configuration of the driver.
SF__SET__ATTR	02	0C	Overrides the current configuration of the driver.
SF__OPEN	02	0E	Reserves the driver for exclusive access. Requests any resources required by the driver.
SF__CLOSE	02	10	Releases the driver from exclusive access.
SF__TIMEOUT	02	12	Reports to the driver that a requested timeout has occurred.
SF__INTERVAL	02	14	Reports to the driver that a requested 60 Hz interval has expired.
SF__TEST	02	16	Performs a hardware test.

Function	Register		
Subfunction	AH	AL	Definition
F__IO__CONTROL			Executes the following subfunctions and any driver dependant subfunctions.
SF__LOCK	04	00	Reserves the sub-address device specified for exclusive access.
SF__UNLOCK	04	02	Releases the sub-address specified from the exclusive access.
F__PUT__BYTE	06		Writes a byte of data.
F__GET__BYTE	08		Reads a byte of data.
F__PUT__BUFFER	0A		Writes a variable length buffer of data (supported by character devices).
F__PUT__BLOCK	0A		Writes a fixed length buffer of data (supported by block devices).
F__GET__BUFFER	0C		Reads a variable length buffer of data (supported by character devices).
F__GET__BLOCK	0C		Reads a fixed length block of data (supported by block devices).
F__PUT__WORD	0E		Writes a word of data.
F__GET__WORD	10		Reads a word of data.

Note: Functions marked with an asterisk (*) should be supported by all drivers. These functions may perform no useful function. However, they should return a status code of RS__DONE or RS__SUCCESSFUL as opposed to RS__UNSUPPORTED.

The following is a list of predefined driver function codes and a brief description of their purpose and parameters:

EX-BIOS Driver Function Definitions

F__ISR (AH = 00H)

This function processes either a logical or a physical interrupt event. It reports whether or not it handled the event through its Return Status Code (see table G.2). The driver may require the service of its parent driver to handle the interrupt.

EX-BIOS drivers do not usually enable interrupts (STI) while processing this function code. Drivers should service this interrupt within 250 microseconds or maintain interrupts off for no more than 250 microseconds at a time. Drivers should expect 40 bytes of stack when called. If a driver enables interrupts it must provide 40 bytes of stack for other ISR's.

On Entry: AH = F_ISR

On Exit: AH = RS_SUCCESSFUL
or RS_NOT_SERVICED

F_SYSTEM (AH = 02H)

This function contains a set of subfunctions that execute system-oriented tasks. These subfunctions include driver setup, configuration, and control. The F_SYSTEM subfunctions are described in detail below.

SF_INIT (AX = 0200H)

This starts the initialization process of a driver. The function does not return to the caller until the driver is ready to be called by another driver. All system services (V_SYSTEM) are assumed to be operational when a driver is called by this function.

The driver is responsible for a brief hardware check and to report RS_FAIL if the test failed. A driver need only execute a test procedure if it directly interfaces to physical hardware.

If the driver requires EX-BIOS RAM the BX and DX registers can be used to reserve available memory (see Section 9).

On Entry: AH = F_SYSTEM (02H)
AL = SF_INIT (00H)
BX = "last used DS"
BP = Driver's vector address

On Exit: AH = Return Status Code
BX = New "last used DS"

Recommended for hardware test failure:

AH = RS_FAIL
ES:DI = pointer to a string of information about the nature of the error
CX = length of the string pointed to by ES:DI

SF_START (AX = 0202H)

This function notifies a driver that it may call other drivers for any additional setup it may require. All other ROM drivers and ROM services are present, active and capable of being accessed. This function does not usually return to the caller until all its internal and external setup is complete.

On Entry: AH = F_SYSTEM (02H)
AL = SF_START (02H)
BP = Driver's vector address

On Exit: AH = Return Status Code

SF_REPORT_STATE (AX = 0204H)

Reports a word of status or state information to the caller in the DX register. The format of the state information will be presented bit wise and should be presented in the same format for all drivers of the same class.

On Entry: AH = F_SYSTEM (02H)
AL = SF_REPORT_STATE
BP = Driver's vector address

On Exit: AH = Return Status Code
BX = State of Driver

SF_VERSION_DESC (AX = 0206H)

Reports the version number of the driver code and an optional describe record which contains other driver-dependent information.

On Entry: AH = F_SYSTEM (02H)
AL = SF_VERSION_DESC (06H)
BP = Driver's vector address

On Exit: AH = Return Status Code
BX = Version number, YYWW is a BCD number where,
WW is the week of the year
YY is the number of years since 1960
CX = Number of bytes in data buffer
ES:DI = Pointer to describe record

SF_DEF_ATTR (AX = 0208H)

Returns a pointer in ES:DI to a parameter block containing the driver's default configuration values. This function does not set the defaults; it only reports them.

```
On Entry:  AH  = F_SYSTEM (02H)
           AL  = SF_DEF_ATTR (08H)
           BP  = Driver's vector address

On Exit:   AH  = Return Status Code
           CX  = Number of bytes in data buffer
        ES:DI  = Pointer to a data buffer
```

SF_GET_ATTR (AX = 020AH)

Reports the configuration values defined by the parameter block. Baud rates, HPIB addresses, etc. may be reported by this command.

```
On Entry:  AH  = F_SYSTEM (02H)
           AL  = SF_GET_ATTR (0AH)
           BP  = Driver's vector address

On Exit:   AH  = Return Status Code
           CX  = Number of bytes in data buffer
        ES:DI  = Pointer to a data buffer
```

SF_SET_ATTR (AX = 020CH)

Sets the parameter block defined by ES:DI as the configuration values. Baud rates, HPIB addresses, etc. may be defined by this command.

```
On Entry:  AH  = F_SYSTEM (02H)
           AL  = SF_SET_ATTR (0CH)
           BP  = Driver's vector address
           CX  = Number of bytes in data buffer
        ES:DI  = Pointer to a data buffer

On Exit:   AH  = Return Status Code
        ES:DI  = Pointer to a data buffer
```

SF__OPEN (AX = 020EH)

Allows exclusive access to this driver. All resources required for driver operation will be acquired at this time. This function has special meaning for the the HP-HIL driver, the HPIB driver and the HPIL driver. Since these drivers support shared interfaces, control of the resource HP-HIL (obtained from the driver V__HPHIL), control of the HPIB (in contention with other PC's on the bus), and control of the HPIL (in contention with other PC's on the loop) is requested and obtained. Control should be kept until a single operation is performed on the resource. A status of RS__BUSY will be reported if the device has previously been opened. RS__SUCCESSFUL will be reported if the device is available. A busy status does not prevent access to the driver. All functions will execute (perhaps improperly) whether a driver has been opened or not.

On Entry:	AH	=	F__SYSTEM (02H)
	AL	=	SF__OPEN (0EH)
	BP	=	Driver's vector address
On Exit:	AH	=	Return Status Code

SF__CLOSE (AX = 0210H)

Closes the requested resource. Again this function has special meaning for the interface class of devices, HPIB, HP-HIL, and HPIL. The driver goes to a state where control can be obtained by or passed to another controller.

On Entry:	AH	=	F__SYSTEM (02H)
	AL	=	SF__CLOSE (10H)
	BP	=	Driver's vector address
On Exit:	AH	=	Return Status Code

SF__TIMEOUT (AX = 0212H)

Reports to the driver that its timer event number has occurred.

On Entry:	AH	=	F__SYSTEM (02H)
	AL	=	SF__TIMEOUT (12H)
	BP	=	Driver's vector address
On Exit:	AH	=	Return Status Code

SF__INTERVAL (AX = 0214H)

Reports to the driver that its interval event number has occurred.

On Entry: AH = F__SYSTEM (02H)
AL = SF__INTERVAL (14H)
BP = Driver's vector address

On Exit: AH = Return Status Code

SF__TEST (AX = 0216H)

The driver performs a hardware test and reports RS__FAIL if the test failed. A driver need only execute a test procedure if it directly interfaces to physical hardware.

On Entry: AH = F__SYSTEM (02H)
AL = SF__TEST (16H)
BP = Driver's vector address

On Exit: AH = Return Status Code

On test failure:

CX = The length of the string pointed to by ES:DI
ES:DI = Pointer to a string of information about the nature of the error

F__IO__CONTROL (AH = 04H)

This is a collection of driver dependant control subfunctions. Drivers of the same class should implement similar subfunctions. The following is a list of predefined driver subfunction codes and a brief description of their purpose and parameters:

SF__LOCK (AX = 0400H)

Reserves the indicated addresses on an already allocated driver for exclusive access.

On Entry: AH = F__IO__CONTROL (04H)
AL = SF__LOCK (00H)
DH,DL = Major and minor address (Optional)
BP = Driver's vector address

On Exit: AH = Return Status Code

SF_UNLOCK (AX = 0402H)

Releases the indicated address from exclusive access.

On Entry:	AH	=	F_IO_CONTROL (04H)
	AL	=	SF_UNLOCK (02H)
	DH,DL	=	Major and minor address (optional)
	BP	=	Driver's vector address
On Exit:	AH	=	Return Status Code

F_PUT_BYTE (AH = 06H)

This is a generic put data byte function.

On Entry:	AH	=	F_PUT_BYTE (06H)
	AL	=	Data byte
	BP	=	Driver's vector address
On Exit:	AH	=	Return Status Code

F_GET_BYTE (AH = 08H)

This is a generic get data byte function.

On Entry:	AH	=	F_GET_BYTE (08H)
	BP	=	Driver's vector address
On Exit:	AH	=	Return Status Code
	AL	=	Data byte

F_PUT_BUFFER OR F_PUT_BLOCK (AH = 0AH)

Puts a number of bytes to a device. The difference between a buffer device and a block device is that a buffer device accepts variable length records, while a block device accepts fixed length records. Thus, a printer is a data buffer device and a disc is a block device. Usually, a block device requires more parameters than a data buffer device, consequently there is a different format for parameter passing.

F__PUT__BUFFER (AH = 0AH)

This is a generic put data buffer or put data block function. Either a string write or a disc block write could use this function.

On Entry: AH = F__PUT__BUFFER (0AH)
CX = Data byte count or block count
ES:DI = Pointer to data buffer
BP = Driver's vector address

On Exit: AH = Return Status Code

F__PUT__BLOCK (AH = 0AH)

Writes a fixed block of data to a block device.

On Entry: AH = F__PUT__BLOCK (0AH)
DH = Major number
DL = Minor number
ES:DI = Command Block

Word 0,1: Data transfer address
Word 2: Block count
Word 3: Block address LSW
Word 4: Block address MSW (for some devices this word is ignored).

BP = Driver's vector address

On Exit: AH = Return Status Code
BX = Operation status

F__GET__BUFFER OR F__GET__BLOCK (AH = 0CH)

F__GET__BUFFER (AH = 0CH)

This is a generic get buffer or get block function. Either string reads or disc block reads could use this function.

On Entry: AH = F__GET__BUFFER (0CH)
CX = Byte count or block count
DS:SI = Pointer to data buffer
BP = Driver's vector address

On Exit: AH = Return Status Code

F__GET__BLOCK (AH = 0CH)

Reads a fixed length block of data from a device.

On Entry: AH = F__GET__BLOCK (0CH)
DH = Major number
DL = Minor number
ES:DI = Command Block

Word 0,1: Data transfer address
Word 2: Block count
Word 3: Block address LSW
Word 4: Block address MSW (for some devices this word is ignored).

BP = Driver's vector address

On Exit: AH = Return Status Code
BX = Operation status

F__PUT__WORD (AH = 0EH)

This is a generic put word of data function. If the destination device is byte wide then the byte in the DL register is written first followed by the byte in the DH register.

On Entry: AH = F__PUT__WORD (0EH)
DX = Data word
BP = Driver's vector address

On Exit: AH = Return Status Code

F__GET__WORD (AH = 10H)

This is a generic get word of data function. If the source device is byte wide then the first byte is read into the DL register and the second byte is read into the DH register.

On Entry: AH = F__GET__WORD (10H)
BP = Driver's vector address

On Exit: AH = Return Status Code
DX = Data word

G.6 Return Status Codes

The conventions for assigning return status codes are as follows:

- If possible, use a return status that has already been defined.
- Error conditions should be reported with a negative byte (0FEH—080H).
- Status or exceptional conditions "soft errors" should be reported with a positive byte (02—7EH).
- Good Status is always reported as 00H.

Table G.3 summarizes the already assigned status codes.

Table G.3

EX-BIOS Return Status Codes

Return Status	Code	Indication
RS_SUCCESSFUL	000H	The requested function executed correctly.
RS_UNSUPPORTED	002H	The requested function or subfunction is not implemented or is unsupported.
RS_NOT_SERVICED	004H	The requested function was not executed by this driver. Any drivers which are chained on this interrupt vector should be called in turn until a return status of RS_SUCCESSFUL or some other error is reported.
RS_DONE	006H	This return status is used by the Input System translators to indicate that an ISR event has been handled and no further processing should be done.
RS_FAIL	0FEH (-02H)	The driver failed the operation in an error state.
RS_TIMEOUT	0FCH (-04H)	The device timed-out on a physical event in an error state.
RS_BAD_PARAMETER	0FAH (-06H)	The driver received a bad parameter.
RS_BUSY	0F8H (-08H)	The requested driver is busy.
RS_NO_VECTOR	0F6H (-0AH)	HP_VECTOR_TABLE is out of RAM or room for more drivers.
RS_OFFLINE	0F4H (-0CH)	Device is offline.
RS_OUT_OF_PAPER	0F2H (-0EH)	Device is out of paper.

G.7 Driver Headers

The EX-BIOS driver header (HP__SHEADER) is a formatted data structure similar to the DOS device driver's header. It defines the attributes of a driver, defines the linkage of a driver and identifies the driver. It also allows the programmer to define how the driver links with other drivers.

All EX-BIOS drivers must have an HP__SHEADER. Programmers are not required to provide a complete HP__SHEADER to use the HP__VECTOR__TABLE. But, if they choose to take advantage of the advanced features of the EX-BIOS the programmer must implement a complete HP__SHEADER. Table G.5 shows a complete driver header and what fields must be present.

Table G.4

Driver Header Table

Variable	Offset	Type	Definition
DH__ATR*	0	Word	Driver Attribute Field
DH__NAME__INDEX	2	Word	Driver String Index Field
DH__V__DEFAULT	4	Word	Driver's Default Logical Device Vector
DH__P__CLASS**	6	Word	Driver's Parent Class
DH__C__CLASS**	8	Word	Driver's Child Class
DH__V__PARENT**	0AH	Word	Driver's Parent Vector
DH__V__CHILD**	0CH	Word	Driver's Child Vector
DH__MAJOR**	0EH	Byte	Subaddress Field
DH__MINOR**	0FH	Byte	Subaddress Field

*This is the only field required for a driver to be in the HP__VECTOR__TABLE.

**These fields are only required by drivers that want to do device mapping.

G.7.1 HP__SHEADER Fields

DH__ATR: Each bit in the DH__ATR field indicates a property of the driver for device mapping purposes. These bits are defined in table G.5.

Table G.5

Device Attributes Bits

Bit	ATR Name	Data	Description
15	ATR__HP	1	The following bytes form a complete driver header.
		0	The bytes that follow are not a driver header.
14	ATR__DEVCFG		Reserved.
13	ATR__ISR	1	The driver can be mapped with DH__V__PARENT.
12	ATR__ENTRY	1	The driver can be mapped with DH__V__CHILD.
11:9	ATR__TYPE__MASK		These three bits indicate the driver type.
	ATR__RSVD	000	This is a reserved vector.
	ATR__FREE	001	This is a free vector. The V__SYSTEM service allocates free vectors to new drivers upon request.
	ATR__SRVC	010	This driver is an EX-BIOS service.
	ATR__LOG	011	This is a logical driver. Its mapping direction is from parent to child.
	ATR__IND	100	This is a mappable driver that cannot be the last in the chain of drivers.
	ATR__BOT	101	This is a mappable driver that is the last in a chain of drivers. This driver can only be a child driver. This driver maps with ATR__LOG, ATR__IND and ATR__BOT drivers.
	ATR__INP	110	This driver is an input driver and is mappable.
		111	Reserved
8	ATR__STRING		Reserved
7	ATR__MAP__CALL	1	This driver's SF__START subfunction should be called whenever the driver is remapped.

Bit	ATR Name	Data	Description
6:5	ATR__SUBADD		These bits specify what type of major and minor addresses the driver requires.
	ATR__NOADDR	00	The driver does not require any address.
	ATR__MAJOR	01	This driver requires that a major address be specified and stored in the parent driver's DH__MAJOR header record. The range of possible major addresses is 0 through the contents of this header's DH__MAJOR.
	ATR__MINOR	10	This driver requires that a minor address be specified and stored in the parent driver's DH__MINOR header record. The range of possible MINOR addresses is 0 through the contents of this header's DH__MINOR. A driver cannot require a minor address unless it also requires a major address.
	ATR__MID	11	This driver requires a major address, a minor address, and a mid address. The minor address field is split into an upper and a lower nibble, with the upper nibble indicating the mid address and the lower nibble indicating the minor address. The range of addresses possible is specified by the child physical driver.
4	ATR__PSHARE	0	This driver cannot be shared between several parent drivers.
3	ATR__CSHARE	0	This driver cannot be shared between several child drivers.
2	ATR__ROM	1	This driver header is in ROM and cannot be altered unless copied to RAM. 1 Reserved
1	ATR__YIELD		Reserved.
0	Reserved		

DH__NAME__INDEX:

The DH__NAME__INDEX is used to derive the localization string index of the driver. This is given by the function F__STR__GET__STRING in the V__SYSTEM driver. See Section 9 for additional information.

DH__V__DEFAULT:

The DH__V__DEFAULT field contains the driver's default vector address.

DH__P__CLASS and DH__C__CLASS: In conjunction, these fields indicate which drivers may be mapped together. DH__P__CLASS and DH__C__CLASS are bit masks. Each bit position represents a set of drivers. If a bit is set then the driver is in that set of drivers. The DH__P__CLASS field indicates a driver is in from 0 to 16 different driver sets. A driver can only map to another driver if its DH__P__CLASS field matches at least one bit position of another driver's DH__C__CLASS field. Furthermore, DH__ATR field is another condition of mapping. The bits are defined in table G.6.

Table G.6

Class Bit Positions

Hex	Bit	Class Name	Definition (If bit = '1', driver is member of class)
8000	0FH	CL__KBDFC	This set of drivers maps to the f1 through f8 softkeys of the keyboard.
4000	0EH	CL__KBD	Keyboard (this is not the device accessed through INT 16).
2000	0DH	CL__CCP	Cursor pad device (for example, V__CCPCUR, V__CCP NUM, V__OFF, V__RAW, V__CCP, V__FUNCTION).
1000	0CH	CL__CON	This set of devices map to the console device.
0800	0BH	CL__BYTE	Serial output device, which may be capable of limited input.
0400	0AH	CL__COMM	Reserved
0200	09H	CL__INTERFACE	An interface class controlling multiple resources transparent to the operating system. It provides major, middle, and minor address modes for the calling application or driver. Examples are the HP-HIL driver, the HPIB driver, and the HPIL driver.
0100	08H	CL__FILT	Serial output device filter. This driver can be mapped in between a logical driver and a physical driver and it can translate from one character set to another.

		Class	**Definition**
Hex	**Bit**	**Name**	**(If bit = '1', driver is member of class)**
0080	07H	CL__BLK	Addressed block device.
0040	06H	CL__BOOT	Logical device used as the priority boot device. If set on a physical device, the device is capable of being a boot device. Typically a physical driver would have both the CL__BOOT bit set and the CL__BLK bit set.
0020	05H	CL__LGID	Logical graphics input device (for example V__LTABLET, V__LPOINTER V__LHPMOUSE, physical GID devices and the keyboard driver). This class maps to logical devices which are not the child of another driver.
0010	04H	CL__PG!D	This class of driver can map to a device which is the child of another driver.
0008	03H	CL__GID	This class is reserved for all diivers which can map to an event.
0004	02H	CL__PTS	Physical touch device (for example, physical GID drivers or V__LTOUCH).
0002	01H	CL__01	Reserved
0001	00H	CL__00	Class Extension Bit
			Special Group Classes
FFFF	–	CL__ALL	This device maps to all other devices (V__PNULL).
0000	–	CL__NULL	This device maps to no other driver.

DH__V__PARENT: The DH__V__PARENT field contains a vector to the driver that is called when the current driver receives an F__ISR function code that it cannot or doesn't know how to process.

DH__V__CHILD: The DH__V__CHILD field contains a vector to the driver that is called if this driver decides it cannot handle the request function (as long as that function is not F__ISR).

DH__MAJOR: Major address range.

DH__MINOR: Minor address range.

See the HP__SHEADER macro definition in the equate files listed in Appendix E.

G.7.2 Driver Mapping

Two drivers may be mapped together if the drivers have matching parent and child class records. The mapping rule for the drivers is defined in table G.7.

Table G.7

PARENT/CHILD Mapping Rules

Parent		Child		
E	**I**	**E**	**I**	**Connection Rule**
0	0	0	0	— Drivers are not to be connected
0	0	0	1	— ″
0	0	1	0	— ″
0	0	1	1	— ″
0	1	0	0	— ″
0	1	0	1	— Child's DH__V__PARENT ← parent's vector address
0	1	1	0	— Drivers can not be connected
0	1	1	1	— Child's DH__V__PARENT ← parent's vector address
1	0	0	0	— Driver's are not connected
1	0	0	1	— ″
1	0	1	0	— Parent's DH__V__CHILD ← child's vector address
1	0	1	1	– Parent's DH__V__CHILD ← child's vector address
1	1	0	0	– Driver's are not connected
1	1	0	1	– Child's DH__V__PARENT ← parent's vector address
1	1	1	0	– Parent's DH__V__CHILD ← child's vector address
1	1	1	1	– Child's DH__V__PARENT ← parent's vector address and Parent's DH__V__CHILD ← child's vector address

Where,
E = ATR__ENTRY bit state
I = ATR__ISR bit state

G.8 Accessing Driver from an Application

When an application needs to access a driver the following sequence must take place:

```
MOV     BP, driver's vector address     ; i.e. V_SYSTEM (12H)
MOV     AH, function code
MOV     AL, subfunction code
.
.                                       ; any other data passed
.                                       ; in registers
PUSH    DS                              ; Saves application's DS
INT     HP_ENTRY                        ; (6FH)
POP     DS
```

G.9 Examples of EX-BIOS Drivers

G.9.1 Cursor Pad Scan Code To HP Mouse Driver

The first example driver is called CPP2GID. This driver implements the V_CCPGID EX-BIOS driver. As such, it translates from cursor control pad keys into graphics input device data.

The driver is installed into the HP_VECTOR_TABLE. The SF_INIT subroutine of the driver asks for enough EX-BIOS RAM to store the driver header and describe record. The DH_V_PARENT field of the V_CCPGID driver header is initialized to V_LHPMOUSE. The DOS driver portion calls SF_START of the EX-BIOS driver. SF_START initializes the DH_V_PARENT field of the V_CCP driver header to V_CCPGID. Then V_LHPMOUSE driver is called with the override function.

The installable driver completes initialization by printing an initialization completed message and returning back to DOS.

Now when the keyboard driver calls V_CCP to process a cursor control pad key, V_CCP calls V_CCPGID. The F_ISR of V_CCPGID decodes which key was actually hit. The driver converts the cursor movement keys (up, down, left, and right) into relative movement data. If the key pressed was an insert or delete key, it is reported as the left or right button respectively. First the driver changes the describe record and then reports either a button press or a button release. After the input data is given to V_LHPMOUSE, the data is available thru the INT 33H STD-BIOS driver.

CCP__TO__GID__FILTER

```
                                  .286c
                                  page 59,132
                                  title CCP_TO_GID_FILTER installable driver
                                  ;***DRIVER HEADER*******************************************************
                                  ;
                                  ; NAME: CCP_TO_GID_FILTER Installed DRIVER
                                  ;
                                  ; DESCRIPTION:  This is an EX-BIOS driver which converts cursor
                                  ;               control pad cursor keys into GID, T_REL16, movements
                                  ;               It is a brother to the V_CCPCUR, V_CCPNUM, V_RAW, and
                                  ;               V_OFF, filters of the V_CCP translator.
                                  ;
                                  ;               One cursor key report generates one micky in the direction
                                  ;               indicated by the cursor pad key.  In addition the cursor
                                  ;               control pad <Ins> key is mapped to the B1 <o> mouse button
                                  ;               and the cursor control pad <DEL> key is mapped to the B2 <oo>
                                  ;               mouse button.
                                  ;
                                  ;  OPERATION:
                                  ;               This driver is installed through the MS-DOS installed device
                                  ;               driver system with the command line:
                                  ;
                                  ;               device=CCP2GID.EXE
                                  ;
                                  ;               The driver links itself into the HP_VECTOR_TABLE and maps
                                  ;               itself to be the parent driver of the V_CCP driver.
                                  ;
                                  ;               The driver then returns to DOS releasing the initialization
                                  ;               code it no longer requires back to DOS.
                                  ;
                                  ;  PARAMETERS
                                  ;
                                  ;    ON ENTRY:           in MS-DOS portion: es:bx points to
                                  ;                                           System Request Header
                                  ;                        in HP portion      ah contains function
                                  ;                                           code, al usually contains
                                  ;                                           the output character
                                  ;
                                  ;    ON EXIT:            in MS-DOS portion. status is returned in
                                  ;                                           System Request Header
                                  ;                        in HP portion:     ah contains the return
                                  ;                                           status code
                                  ;
                                  ; REGISTERS ALTERED:     in MS-DOS portion. none
                                  ;                        in HP portion      ax, bx, di, bp
                                  ;
                                  ;***********************************************************************
                                  HP_SHEADER              struc
      0000  0000                  DH_ATR                  dw      0
      0002  0000                  DH_NAME_INDEX           dw      0
      0004  0000                  DH_V_DEFAULT            dw      0
      0006  0000                  DH_P_CLASS              dw      0
      0008  0000                  DH_C_CLASS              dw      0
      000A  0000                  DH_V_PARENT             dw      0
      000C  0000                  DH_V_CHILD              dw      0
      000E  00                    DH_MAJOR                db      0
      000F  00                    DH_MINOR                db      0
      0010                        HP_SHEADER              ends

      = 006F                      HP_ENTRY                equ     06FH

                                  SYSCALL                 macro   vector
                                   ifnb                   <vector>
                                                          mov     bp,vector
                                   endif
                                                          int     HP_ENTRY
                                                          endm

      = 0008                      ATR_CSHARE              equ     0008H
      = 4000                      ATR_DEVCFG              equ     4000H
      = 8000                      ATR_HP                  equ     8000H
      = 2000                      ATR_ISR                 equ     2000H
      = 0600                      ATR_LOG                 equ     0600H
      = 2000                      CL_CCP                  equ     2000H
      = 0020                      CL_LGID                 equ     0020H

                                  DESCRIBE                STRUC
      0000      10 [                          db          size HP_SHEADER dup (?) ; this data is always offset by
                         ??
                              ]

      0010  ??                    D_SOURCE        db      ?         ; 7-4 (high nibble) contains the GID type
                                                                    ; 3-0 (low nibble) is the address of the device
      0011  ??                    D_HPHIL_ID      db      ?         ; device id byte returned by an HPHIL device
      0012  ??                    D_DESC_MASK     db      ?         ; describe header from HPHIL device
      0013  ??                    D_IO_MASK       db      ?         ; I/O descriptor byte from device
      0014  ??                    D_XDESC_MASK    db      ?         ; extended describe byte from device
      0015  ??                    D_MAX_AXIS      db      ?         ; maximum number of axis reported
      0016  ??                    D_CLASS         db      ?         ; device class
                                                                    ; 7-4 (high nibble) contains current class
                                                                    ; 3-0 (low nibble) contain the default class
      0017  ??                    D_PROMPTS       db      ?         ; number of buttons/prompts
                                                                    ; 7-4 (high nibble) is the number of prompts
                                                                    ; 3-0 (low nibble) is the number of buttons
```

CCP__TO__GID__FILTER

```
0018  ??                        D_RESERVED      db      ?       ; reserved for future
0019  ??                        D_BURST_LEN     db      ?       ; maximum burst length output to a device
                                                                ; if devices supports more than 255 bytes then
                                                                ; 255 bytes is the default maximum
001A  ??                        D_WR_REG        db      ?       ; number of write registers supported by a device
001B  ??                        D_RD_REG        db      ?       ; number of read registers supported by a device
001C  ??                        D_TRANSITION    db      ?       ; transitions reported per button
001D  ??                        D_STATE         db      ?       ; current state of buttons
001E  ????                      D_RESOLUTION    dw      ?       ; counts / cm (m) returned by HPHIL device
0020  ????                      D_SIZE_X        dw      ?       ; Maximum count of in units of resolution
0022  ????                      D_SIZE_Y        dw      ?       ;
0024  ????                      D_ABS_X         dw      ?       ; data reported from device
0026  ????                      D_ABS_Y         dw      ?       ; that reports absolute data
0028  ????                      D_REL_X         dw      ?       ; data reported from device
002A  ????                      D_REL_Y         dw      ?       ; that is relitive
002C  ????                      D_ACCUM_X       dw      ?       ; these are used to accumulate scaling
002E  ????                      D_ACCUM_Y       dw      ?       ; remainder
0030                            DESCRIBE                ENDS

= 0030                          DESCRIBE_SIZE           equ     size DESCRIBE

= 001E                          D_CCP_STATE             equ     D_STATE + 1
=                               D_SIZE                  equ     D_SIZE_X
=                               D_SAMPLE_ABSOLUTE       equ     D_ABS_X
=                               D_SAMPLE_RELATIVE       equ     D_REL_X
=                               D_REMAINDER_ACCUM       equ     D_ACCUM_X
=                               D_BUFFER                equ     D_SIZE_X        ; offset where buffer begins
= 00F0                          D_CLASS_CURRENT         equ     0F0H
= 000F                          D_CLASS_DEFAULT         equ     00FH
                                ; The field LD_SOURCE uses the following to access the defined nibbles
= 000F                          D_ADDR_MASK             equ     00FH
= 00F0                          D_TYPE_MASK             equ     0F0H

= 000E                          F_INS_FIXGETDS          equ     000EH
= 0004                          F_IO_CONTROL            equ     0004H
= 0002                          SF_MOUSE_OVERRIDE       equ     0002H
= 0000                          F_ISR                   equ     0000H
= 0002                          F_SYSTEM                equ     0002H
= 0002                          SF_START                equ     0002H

                                ;****************************************************************************
                                ; the following structures are used to access MS-DOS driver command blocks
                                ;****************************************************************************
                                MSD_HEADER      macro   ATT,STRATEGY_ENTRY,ISR_ENTRY,STRING
                                                dd       -1             ;mark as last driver in list
                                                dw       ATT
                                                dw       STRATEGY_ENTRY
                                                dw       ISR_ENTRY
                                                db       STRING
                                                db       14 dup (?)      ; Pad so it is paragraph aligned.
                                                endm

                                MSD_REQ_HEADER  struc           ;00; structure for access to MS driver cmds
0000  ??                        MSD_CMDLEN      db      ?       ;00; length of cmd in bytes including data @ end
0001  ??                        MSD_UNIT        db      ?       ;01; unit number for command
0002  ??                        MSD_CMD         db      ?       ;02; command code
0003  ????                      MSD_STATUS      dw      ?       ;03; filler with completion status before return
0005      08 [                                  db 8 dup (?)    ;  ; area reserved for DOS
                ??
                    ]


000D  ??                        MSD_MEDIA       db      ?       ;13; most cmds have this defined in the data area
000E  ????                      MSD_TRANS       dw      ?       ;14;
0010  ????                                      dw      ?       ;16;
0012  ????                      MSD_COUNT       dw      ?       ;18;
0014  ????                      MSD_START       dw      ?       ;20;
0016                            MSD_REQ_HEADER          ends

                                MSD_INIT_CMD            struc
0000      0D [                                  db      13 dup (?)   ;first cover header area
                ??
                    ]

000D  ??                        MSD_UNIT_COUNT  db      ?       ; 0B ;number of units service by this driver
000E  ????                      MSD_END_OFFSET  dw      ?       ; 0C ;offset of end of code
0010  ????                      MSD_END_SEG     dw      ?       ; 0E ;segment address of end of code
0012  ????                      MSD_BPB_OFFSET  dw      ?       ; 12 ;
0014  ????                      MSD_BPB_SEG     dw      ?       ; 14 ;seg:offset of BPB list for units attached
0016  ??                        MSD_1ST_UNIT    db      ?       ; 16 ,tells driver letter of first unit
0017                            MSD_INIT_CMD            ends

= 0000                          MSD_INIT                equ     0000H
= 0003                          MSD_UNKNOWN_CMD         equ     0003H
= 000F                          MSD_REM_MEDIA           equ     000FH
= 0081                          MSD_ERR_STATUS          equ     10000001B       ;used as upper byte in status wrd
= 0001                          MSD_DONE_STATUS         equ     00000001B       ;  bit 15=err bit 8=done

= 0006                          RS_DONE                 equ     0006H
= 0000                          RS_SUCCESSFUL           equ     0000H
= 0002                          RS_UNSUPPORTED          equ     0002H

= 0009                          T_KC_BUTTON             equ     0009H
= 0041                          T_REL16                 equ     0041H
```

```

 = 0006                         V_DOLITTLE              equ     0006H
 = 00A2                         V_CCPGID                equ     00A2H
 = 00CC                         V_LHPMOUSE              equ     00CCH
 = 0012                         V_SYSTEM                equ     0012H
 = 004E                         V_CCP                   equ     004EH

 = 0080                         UP_DOWN_BIT             equ     10000000B       ;Key up or down
 = 00FF                         INIT_BUT_STATE          equ     0FFH    ;All off
 = 004C                         MSE_NUM_BUTTON          equ     004CH   ;Offset of number of button in mouse RAM
 = 0030                         CCP2GID_DESC_SIZE       equ     48
 = E608                         CCP2GID_HP_ATTR         equ     ATR_HP+ATR_DEVCFG+ATR_ISR+ATR_LOG+ATR_CSHARE

                               ;=====================================================================;
                               ; MS-DOS device drivers start at an offset of 0 rather than 100h.     ;
                               ;=====================================================================;
                               CGROUP                  group   CODE
 0000                          CODE                    segment public 'CODE'
                                                       assume  cs:CODE, ds:NOTHING
 0000                                                  org     0
 0000                          CCP2GID_INSTALLED       LABEL   FAR
                               ;=====================================================================;
                               ; This is the start of MS-DOS driver portion of the code.  It pretends ;
                               ; to be a standard MS-DOS driver long enough to be loaded and          ;
                               ; initialized via CONFIG.SYS.  After that this section of code will    ;
                               ; not be used.  (section 1)                                            ;
                               ;=====================================================================;
                               ; This is the MS-DOS device driver header.  It must be the first thing
                               ; in the code segment.  Consult the HP Vectra MS-DOS Programmers Refer-
                               ; ence Manual for more  information.  BE SURE YOU DON'T RELEASE THE
                               ; HEADER AREA AS AVAILABLE  MEMORY, EVEN ON AN ERROR. THE SYSTEM WILL
                               ; CRASH IF YOU DO.
                               ; This is the only resident portion of the DOS driver, the rest
                               ; of the DOS driver is returned to DOS memory.
                               ;=====================================================================;
                                       MSD_HEADER 08000h,dev_strategy,dev_int," CCP2GID"  ;device header
 0000  FF FF FF FF            +                dd        -1              ;mark as last driver in list
 0004  8000                   +                dw        08000h
 0006  01AB R                 +                dw        dev_strategy
 0008  01D6 R                 +                dw        dev_int
 000A  20 43 43 50 32 47      +                db        " CCP2GID"
 0012      0E [               +                db        14 dup (?)      ; Pad so it is paragraph aligned.

                                subttl CCP2GID DRIVER Main entry point
                                page
                               ;**********************************************************************
                               ; CS: Relative Data Area For Driver
                               ;**********************************************************************
 0020  ????                    sav_bx:         dw      ?
 0022  ????                    sav_cx:         dw      ?
 0024  ????                    sav_dx:         dw      ?
 0026  ????                    sav_es:         dw      ?
 0028  ????                    top_hp_entry:   dw      ?

                               ;**********************************************************************
                               ; This is the EX-BIOS installed driver CCP2GID.

                               ;**********************************************************************
 002A                          CCP2GID_DRIVER  PROC    FAR
 002A  80 FC 00                                cmp     ah,F_ISR                ;Is the function F_ISR?
 002D  74 0B                                   je      short CCP2GID_ISR
 002F  80 FC 02                                cmp     ah,F_SYSTEM             ;Is the function F_SYSTEM?
 0032  75 03                                   jne     CCP2GID_UNSUPPORTED
 0034  E9 010D R                               jmp     CCP2GID_SYSTEM
 0037                          CCP2GID_UNSUPPORTED:
 0037  B4 02                                   mov     ah,RS_UNSUPPORTED       ;This driver doesn't support
 0039  CF                                      iret                            ;any other functions.

 003A                          CCP2GID_DRIVER  ENDP    FAR

                                               subttl CCP2GID_isr function
                               page
                               ;===DRIVER HEADER=====================================================
                               ;
                               ;  NAME:        CCP2GID_ISR
                               ;
                               ;  DESCRIPTION:  This function translates valid ISR event record into
                               ;                mouse type movement or button reports, calls its parent driver
                               ;                with an ISR Event Record and then returns to the calling
                               ;                driver with a return status of RS_DONE
                               ;
                               ;  PARAMETERS
                               ;
                               ;    ON ENTRY:   ISR Event Record of type T_KC_HP_CCP
                               ;                       BP = V_CCPGID
                               ;                       DS = this drivers data segemnt
                               ;                       AH = 0 ( F_ISR )
                               ;
                               ;
                               ;    CALL PARENT:       ISR Event Record of type T_REL16 or T_KC_BUTTON
                               ;
                               ;       T_REL16:        AH = 0 ( F_ISR )
                               ;                       BX = axis 0 value ( X axis or Col )
                               ;                       CX = axis 1 value ( Y axis or Row )
                               ;                      DH = 41H ( T_REL16 )
                               ;                      ES:0 = describe record of V_CCPGID
                               ;                      DL = V_CCPGID/6
                               ;
                               ;        T_KC_BUTTON:  AH = 0 ( F_ISR )
                               ;                      BL = 000H -  break Button 1
                               ;                           001H -  break Button 2
                               ;                           080H -  make Button 1
                               ;                           081H -  make Button 2
                               ;                      DH = T_KC_BUTTON
                               ;                      CX = 0
                               ;                      ES:0 = this device describe record
                               ;                      DL = V_CCPGID/6
                               ;
                               ;    ON EXIT:
                               ;                      AH = RS_DONE
                               ;
                               ; REGISTERS ALTERED:    ax,  bp and ds
                               ;
                               ;===================================================================
003A                           CCP2GID_ISR     label   near

003A  50                                       push    ax
003B  2E: 89 1E 0020 R                         mov     word ptr cs:sav_bx,bx   ;save the keyboard's isr
0040  2E: 89 0E 0022 R                         mov     word ptr cs:sav_cx,cx
0045  2E: 89 16 0024 R                         mov     word ptr cs:sav_dx,dx
004A  2E: 8C 06 0026 R                         mov     word ptr cs:sav_es,es

004F  8C DA                                    mov     dx,ds                   ;point to the mouse isr
0051  8E C2                                    mov     es,dx

0053  32 FF                                    xor     bh,bh                   ;translate the scancode to GID
0055  83 FB 60                                 cmp     bx,60H                  ;check for cursor up
0058  74 21                                    je      short ccp_up
005A  83 FB 61                                 cmp     bx,61H                  ;check for cursor left
005D  74 24                                    je      short ccp_left
005F  83 FB 62                                 cmp     bx,62H                  ;check for cursor down
0062  74 27                                    je      short ccp_down
0064  83 FB 63                                 cmp     bx,63H                  ;check for cursor right
0067  74 2A                                    je      short ccp_right
0069  80 E3 7F                                 and     bl,07FH
006C  83 FB 68                                 cmp     bx,68H                  ;check for INS or button 1
006F  74 3E                                    je      short ccp_but1
0071  83 FB 69                                 cmp     bx,69H                  ;check for DEL or button 2
0074  74 3E                                    je      short ccp_but2
0076  B4 06                                    mov     ah,RS_DONE              ;recieved an unsupported key
0078  EB 7B 90                                 jmp     exit_Isr

007B                           ccp_up:
007B  BB 0000                                  mov     bx,0                    ;no movement on the X-axis
007E  B9 FFF8                                  mov     cx,-8                   ;industry standard upward move
0081  EB 18                                    jmp     short rel_move

0083                           ccp_left:
0083  BB FFF8                                  mov     bx,-8                   ;negative move on the X-axis
0086  B9 0000                                  mov     cx,0                    ;no movement on the Y-axis
0089  EB 10                                    jmp     short rel_move

008B                           ccp_down:
008B  BB 0000                                  mov     bx,0                    ;no movement on the X-axis
008E  B9 0008                                  mov     cx,8                    ;industry standard down move
0091  EB 08                                    jmp     short rel_move

0093                           ccp_right:
0093  BB 0008                                  mov     bx,8                    ;move right on the X-axis
0096  B9 0000                                  mov     cx,0                    ;no movement on the Y-axis
0099  EB 00                                    jmp     short rel_move

009B                           rel_move:
009B  89 1E 0028                               mov     ds:D_REL_X,bx           ;save new relative move (X)
009F  89 0E 002A                               mov     ds:D_REL_Y,cx           ;save new relative move (Y)
00A3  01 1E 0024                               add     ds:D_ABS_X,bx           ;save new absolute position (X)
00A7  01 0E 0026                               add     ds:D_ABS_Y,cx           ;save new absolute position (Y)
00AB  B6 41                                    mov     dh,T_REL16
00AD  EB 3C                                    jmp     short give_to_parent

00AF                           ccp_but1:
00AF  BB 0000                                  mov     bx,0                    ;button one got pushed
00B2  EB 05                                    jmp     short but_process

00B4                           ccp_but2:
00B4  BB 0001                                  mov     bx,1                    ;button two got pushed
00B7  EB 00                                    jmp     short but_process

00B9                           but_process:
00B9  B8 0001                                  mov     ax,0001H                ;get the proper bit set in D_STATE
00BC  8A CB                                    mov     cl,bl
00BE  D2 E0                                    shl     al,cl
00C0  A2 001C                                  mov     byte ptr ds:D_TRANSITION,al     ;record in the describe record
                                                                                       ;which button changed

```

```
00C3  2E: 8B 0E 0020 R                          mov     cx,word ptr cs:sav_bx   ;get the scan code and check for
00C8  F6 C1 80                                  test    cl,UP_DOWN_BIT          ;push or release
00CB  74 06                                     jz      but_push

00CD                           but_release:
00CD  08 06 001D                                or      ds:D_STATE,al           ;show the release in D_STATE by
00D1  EB 08                                     jmp     short button_done       ;setting the bit

00D3                           but_push:
00D3  F6 D0                                     not     al                      ;show the push in D_STATE by
00D5  20 06 001D                                and     ds:D_STATE,al           ;clearing the bit
00D9  EB 00                                     jmp     short button_done

00DB                           button_done:
00DB  2E: A1 0020 R                             mov     ax,word ptr cs:sav_bx   ;was button pushed or released?
00DF  24 80                                     and     al,080H
00E1  0A D8                                     or      bl,al                   ;record in bx
00E3  32 FF                                     xor     bh,bh
00E5  33 C9                                     xor     cx,cx
00E7  B6 09                                     mov     dh,T_KC_BUTTON
00E9  EB 00                                     jmp     short give_to_parent

00EB                           give_to_parent:
00EB  B4 00                                     mov     ah,F_ISR                ;Execute ISR of parent
00ED  B2 1B                                     mov     dl,V_CCPGID/6           ,source vector is this driver
00EF  8B 2E 000A                                mov     bp,ds:DH_V_PARENT       ;Get my parent's vector from my header
                                                SYSCALL
00F3  CD 6F                  +                          int     HP_ENTRY
00F5                           exit_isr:
00F5  2E: 8B 1E 0020 R                          mov     bx,word ptr cs:sav_bx   ;restore to keyboard ISR state
00FA  2E: 8B 0E 0022 R                          mov     cx,word ptr cs:sav_cx
00FF  2E: 8B 16 0024 R                          mov     dx,word ptr cs:sav_dx
0104  2E: 8E 06 0026 R                          mov     es,word ptr cs:sav_es
0109  58                                        pop     ax
010A  B4 06                                     mov     ah,RS_DONE              ;Record on return
010C  CF                                        iret

                                                subttl CCP2GID_system function
                               page
                               ;***DRIVER HEADER*************************************************
                               ;
                               ;  NAME:        CCP2GID_system function
                               ;
                               ;  DESCRIPTION: Decodes the appropriate system function,
                               ;       Supported Subfunctions are:  SF_INIT
                               ;                                    SF_START
                               ;                                    SF_REPORT_STATE
                               ;                                    SF_VERSION_DESC
                               ;
                               ;  PARAMETERS
                               ;
                               ;    ON ENTRY:
                               ;
                               ;    ON EXIT:
                               ;
                               ;  REGISTERS ALTERED:     ax, bx, di, bp
                               ;
                               ;*****************************************************************
010D                           CCP2GID_SYSTEM  label    near

010D  3C 06 90 90                               cmp     al,MAX_CCP2GID_SYS_FN           ;Is the system subfunction
0111  77 0D                                     ja      short   CCP2GID_bad_sys_fn      ;within the valid range?

0113  87 EB                                     xchg    bp,bx                   ;Load the jump table index
0115  8A D8                                     mov     bl,al                   ;into bp.
0117  32 FF                                     xor     bh,bh
0119  87 EB                                     xchg    bp,bx

011B  2E: FF A6 0123 R                          jmp     cs:word ptr     CCP2GID_sys_case[bp]

0120                           CCP2GID_bad_sys_fn:
0120  B4 02                                     mov     ah,RS_UNSUPPORTED       ;Return status as unsupported
0122  CF                                        iret

                               ; CCP2GID_system subfunction jumptable

0123                           CCP2GID_sys_case:
0123  012B R                                    dw      word ptr        CCP2GID_sys_init        ;SF_INIT
0125  0147 R                                    dw      word ptr        CCP2GID_sys_start       ;SF_START
0127  0120 R                                    dw      word ptr        CCP2GID_bad_sys_fn      ;SF_REPORT_STATE
0129  0120 R                                    dw      word ptr        CCP2GID_bad_sys_fn      ;SF_VERSION_DESC
= 0006                         MAX_CCP2GID_SYS_FN       equu    byte ptr        ($ - CCP2GID_sys_case - 2)

                                                subttl  CCP2GID_system function - init subfunction

                               page
                               ;***DRIVER HEADER*************************************************
                               ;
                               ;  NAME:        CCP2GID_system function - init subfunction
                               ;
                               ;  DESCRIPTION: Initializes Describe Record and Exits, allocating a
                               ;               DS.
```

```
                              ;
                              ;  PARAMETERS
                              ;
                              ;    ON ENTRY:  ah = F_SYSTEM
                              ;               al = SF_INIT
                              ;               bp = V_CCPGID
                              ;               bx = last used data segment
                              ;
                              ;    ON EXIT:   ah = RS_SUCCESSFUL
                              ;               bx = last used data segment - this drivers data segment
                              ;
                              ; REGISTERS ALTERED:     ah, bp, and ds
                              ;
                              ;=====================================================================

012B                          CCP2GID_sys_init label  near
012B  06                                      push    es
012C  56                                      push    si
012D  57                                      push    di
012E  51                                      push    cx
012F  83 EB 03                                sub     bx,(CCP2GID_DESC_SIZE+15)/16
0132  8E C3                                   mov     es,bx
0134  BE 0177 R                               mov     si,offset cs:CCP2GID_desc_headr
0137  FC                                      cld
0138  33 FF                                   xor     di,di
013A  B9 0030                                 mov     cx,CCP2GID_DESC_SIZE
013D  F3/ 2E: A4                              rep     movs byte ptr es:[di],cs:[si]
0140  59                                      pop     cx
0141  5F                                      pop     di
0142  5E                                      pop     si
0143  07                                      pop     es
0144  B4 00                                   mov     ah,RS_SUCCESSFUL
0146  CF                                      iret

                                      page
                              ;===DRIVER HEADER=====================================================
                              ;
                              ;  NAME:        CCP2GID_system function - start subfunction
                              ;
                              ;  DESCRIPTION: Relinks the V_CCP driver to this driver, V_CCPGID,
                              ;               so this driver is activated to translate cursor control
                              ;               pad reports to mouse type movements.
                              ;
                              ;
                              ;  PARAMETERS
                              ;
                              ;    ON ENTRY:  AH = F_SYSTEM
                              ;               AL = SF_START
                              ;               BP = V_CCPGID
                              ;
                              ;
                              ;    ON EXIT:   AH = RS_SUCCESSFUL
                              ;
                              ; REGISTERS ALTERED:     ah, ds, bp
                              ;
                              ;=====================================================================


0147                          CCP2GID_sys_start label near
0147  50                              push     ax
0148  1E                              push     ds                        ;Save the ds register
0149  B8 0000                         mov      ax,0                      ;Set the ds to the Int vector
014C  8E D8                           mov      ds,ax
014E  A1 01BE                         mov      ax,ds:[(4 * 6FH) + 2]  ;Get the HP_VECTOR_TABLE segment
0151  2E: A3 0028 R                   mov      word ptr cs:top_hp_entry,ax
0155  8E D8                           mov      ds,ax
0157  A1 0052                         mov      ax,ds:[V_CCP+4]                 ;Get the top of the header for the
015A  8E D8                           mov      ds,ax                           ;V_CCP driver
015C  C7 06 000A 00A2                 mov      ds:[DH_V_PARENT],V_CCPGID       ;Have V_CCP point to me.
0162  2E: A1 0028 R                   mov      ax,word ptr cs:top_hp_entry     ;load ds with HP_VECTOR_TABLE segment
0166  8E D8                           mov      ds,ax
0168  A1 00D0                         mov      ax,ds:[V_LHPMOUSE+4]    ;Point to the top of RAM for the mouse driver
016B  8E D8                           mov      ds,ax
016D  C6 06 004C 02                   mov      byte ptr ds:MSE_NUM_BUTTON,2    ;Define the number of buttons to 2.
0172  1F                              pop      ds                              ;Restore the ds
0173  58                              pop      ax
0174  B4 00                           mov      ah,RS_SUCCESSFUL
0176  CF                              iret

                               subttl DOS-Install Code ( Returned to DOS )
                               page
0177                          RETURN_THE_FOLLOWING_RAM_TO_DOS label    far
                              ; temporary EX-BIOS Header configuration template

0177  E608                    CCP2GID_desc_headr      HP_SHEADER      <CCP2GID_HP_ATTR,V_CCPGID/6,V_CCPGID,CL_CCP,CL_LGID
                      ,V_LHPMOUSE,V_DOLITTLE>
0179  001B
017B  00A2
017D  2000
017F  0020
0181  00CC
0183  0006
0185  00
0186  00
```

CCP__TO__GID__FILTER

```

0187                           CCP2GID_desc:
0187  0F                                 db      0FH             ;D_SOURCE
0188  00                                 db      0               ;D_HPHIL_IO
0189  00                                 db      0               ;D_DESC_MASK
018A  00                                 db      0               ;D_IO_MASK
018B  00                                 db      0               ;D_XDESC_MASK
018C  02                                 db      2               ;D_MASK_AXIS
018D  00                                 db      0               ;D_CLASS
018E  20                                 db      020h            ;D_PROMPTS
018F  00                                 db      0               ;D_RESERVED
0190  00                                 db      0               ;D_BURST_LEN
0191  00                                 db      0               ;D_WR_REG
0192  00                                 db      0               ;D_RD_REG
0193  01                                 db      1               ;D_TRANSITION
0194  FF                                 db      INIT_BUT_STATE  ;D_STATE
0195  00C8                               dw      200             ;D_RESOLUTION
0197  0000                               dw      0               ;D_SIZE_X
0199  0000                               dw      0               ;D_SIZE_Y
019B  0000                               dw      0               ;D_ABS_X
019D  0000                               dw      0               ;D_ABS_Y
019F  0000                               dw      0               ;D_REL_X
01A1  0000                               dw      0               ;D_REL_Y
01A3  0000                               dw      0               ;D_ACCUM_X
01A5  0000                               dw      0               ;D_ACCUM_Y

                               ;===================================================================;
                               ; This completes the MS-DOS device driver section                   ;
                               ; and begins the HP device driver code. (section 2)                 ;
                               ;===================================================================;
01A7  ????                     rh_off          dw      ?       ;request header pointer offset
01A9  ????                     rh_seg          dw      ?       ;request header pointer segment
                               ;===================================================================;
                               ; device strategy (required by MS-DOS 3.1)                          ;
                               ;       save es:bx (address of request header) and return           ;
                               ;===================================================================;
01AB                           dev_strategy    PROC    FAR
                               ;===================================================================;
01AB  2E: 89 1E 01A7 R                 mov     cs:rh_off,bx    ;save offset  of request header ptr.
01B0  2E: 8C 06 01A9 R                 mov     cs:rh_seg,es    ;save segment of request header ptr.
01B5  CB                               ret
01B6                           dev_strategy    ENDP    FAR
                               ;===================================================================;
                               ; device interrupt (required by MS-DOS 3.1)                         ;
                               ;       use the command from the request header block as an index   ;
                               ;       into the table of command processing routines.              ;
                               ;===================================================================;
01B6                           command_table   label word
                               ;===================================================================;
01B6  0286 R                           dw      init            ;initialization
01B8  0213 R                           dw      media_check     ;media check (block only)
01BA  0213 R                           dw      build_bpb       ;build bpb   (block only)
01BC  0213 R                           dw      ioctl_in        ;ioctl input
01BE  0213 R                           dw      input           ;input (read)
01C0  0213 R                           dw      nd_input        ;non-destruct. read   (char only)
01C2  0213 R                           dw      in_stat         ;input status         (char only)
01C4  0213 R                           dw      in_flush        ;input buffer flush   (char only)
01C6  0213 R                           dw      output          ;output (write)
01C8  0213 R                           dw      out_verify      ;output (write) with verify
01CA  0213 R                           dw      out_stat        ;output status        (char only)
01CC  0213 R                           dw      out_flush       ;output buffer flush  (char only)
01CE  0213 R                           dw      ioctl_out       ;ioctl output
01D0  0213 R                           dw      dev_open        ;device open
01D2  0213 R                           dw      dev_close       ;device close
01D4  0213 R                           dw      rem_media       ;removable media

                               ;===================================================================;
01D6                           dev_int         PROC    FAR
                               ;===================================================================;
01D6  9C                               pushf
01D7  FC                               cld                     ;preserve machine state
01D8  60                               pusha
01D9  1E                               push    ds
01DA  06                               push    es
                               ; DS is CS
01DB  0E                               push    cs
01DC  1F                               pop     ds

01DD  C4 36 01A7 R                     les     si,dword ptr ds:[rh_off] ;loads es:si
01E1  26: 8A 5C 02                     mov     bl,es:[si].MSD_CMD          ;get function byte
01E5  80 FB 00                         cmp     bl,MSD_INIT                 ;quit if lower than
01E8  72 11                            jb      bad_cmd                     ; lowest command number
01EA  80 FB 0F                         cmp     bl,MSD_REM_MEDIA            ;quit if higher than
01ED  77 0C                            ja      bad_cmd                     ; highest command number
                               ;
                               ; command is valid; go do it
                               ;
01EF  32 FF                            xor     bh,bh
01F1  D1 E3                            shl     bx,1            ;make BX offset into table
01F3  2E: FF 97 01B6 R                 call    word ptr cs:command_table[bx]
01F8  EB 10 90                         jmp     int_exit
                               ;
```

```
                            ; unknown command routine
                            ;
01FB                        bad_cmd:
01FB  C4 36 01A7 R                  les     si,dword ptr ds:[rh_off] ;reload es:si w/ header addr
01FF  B0 03                         mov     al,MSD_UNKNOWN_CMD
0201  B4 81                         mov     ah,MSD_ERR_STATUS        ;status word now in AX
0203  26: 89 44 03                  mov     es:[si].MSD_STATUS,ax    ;place in request header
0207  EB 01 90                      jmp     int_exit
                            ;
                            ; all finished
                            ;
020A                        int_exit:
020A  C4 1E 01A7 R                  les     bx,dword ptr ds:[rh_off] ;reload es:bx w/ header addr
020E  07                            pop     es
020F  1F                            pop     ds               ;restore all preserved registers
0210  61                            popa
0211  9D                            popf
0212  CB                            ret
0213                        dev_int         ENDP    FAR

                            ;====================================================================;
                            ; All MS-DOS functions except init are unsupported and do nothing.   ;
                            ;====================================================================;
0213                        unsupported     PROC    NEAR
                            ;====================================================================;
0213                        media_check:
0213                        build_bpb:
0213                        ioctl_in:
0213                        input:
0213                        nd_input:
0213                        in_stat:
0213                        in_flush:
0213                        output:
0213                        out_verify:
0213                        out_stat:
0213                        out_flush:
0213                        ioctl_out:
0213                        dev_open:
0213                        dev_close:
0213                        rem_media:
0213                        all_ok:
0213  32 C0                         xor     al,al                    ;0 indicates OK
0215  B4 01                         mov     ah,MSD_DONE_STATUS
0217  C4 36 01A7 R                  les     si,dword ptr ds:[rh_off] ;reload es:si w/ header addr
021B  26: 89 44 03                  mov     es:[si].MSD_STATUS,ax    ;return ok status
021F  C3                            ret
0220                        unsupported     ENDP    NEAR

                             page
                            ;====================================================================;
                            ; init  -  setup variables & establish link in HP_VECTOR_TABLE       ;
                            ;====================================================================;
0220  48 50 20 43 43 50     init_msg        db      "HP CCP2GID installed driver 2.2",0dH,0aH,"$"
      32 47 49 44 20 69
      6E 73 74 61 6C 6C
      65 64 20 64 72 69
      76 65 72 20 32 2E
      32 0D 0A 24
0242  48 50 20 43 43 50     init_msg2       db      "HP CCP2GID installation failed",0dH,0aH,"$"
      32 47 49 44 20 69
      6E 73 74 61 6C 6C
      61 74 69 6F 6E 20
      66 61 69 6C 65 64
      0D 0A 24
0263  48 50 20 43 43 50     init_msg3       db      "HP CCP2GID installation suceeded",0dH,0aH,"$"
      32 47 49 44 20 69
      6E 73 74 61 6C 6C
      61 74 69 6F 6E 20
      73 75 63 65 65 64
      65 64 0D 0A 24
                            ;====================================================================;
0286                        init    PROC    NEAR
                            ;====================================================================;
0286  FA                            cli
                            ;
                            ; Put next available memory location in System Request Header
                            ;
0287  C4 36 01A7 R                  les     si,dword ptr ds:[rh_off]                ;es:si := header addr
028B  8D 06 0177 R                  lea     ax,cs:RETURN_THE_FOLLOWING_RAM_TO_DOS   ;put next free loc
028F  26: 89 44 0E                  mov     es:word ptr [si].MSD_END_OFFSET,ax      ;address in header
0293  8C C8                         mov     ax,cs
0295  26: 89 44 10                  mov     es:word ptr [si].MSD_END_SEG,ax
                            ;
                            ; Put the driver into the HP_VECTOR_TABLE
                            ;
0299  0E                            push    cs
029A  07                            pop     es

                            ; install (init ) V_CCPGID

029B  B4 0E                         mov     ah, F_INS_FIXGETDS   ; Puts the driver in HP_VECTOR_TABLE
029D  BB 00A2                       mov     bx, V_CCPGID         ; and calls to do F_SYSTEM+SF_INIT
02A0  8D 3E 002A R                  lea     di, CCP2GID_DRIVER
```

CCP__TO__GID__FILTER

```
02A4  1E                              push    ds
                                      syscall V_SYSTEM
02A5  BD 0012               +                         mov     bp,V_SYSTEM
02A8  CD 6F                 +                         int     HP_ENTRY
02AA  1F                              pop     ds
                          ; start V_CCPGID
02AB  B4 02                           mov     ah, F_SYSTEM
02AD  B0 02                           mov     al, SF_START
02AF  1E                              push    ds
                                      syscall V CCPGID
02B0  BD 00A2               +                         mov     bp,V_CCPGID
02B3  CD 6F                 +                         int     HP_ENTRY
02B5  1F                              pop     ds
                          ; install HP Mouse Driver whether there is an HP Mouse or not
02B6  B4 04                           mov     ah,F_IO_CONTROL
02B8  B0 02                           mov     al,SF_MOUSE_OVERRIDE
                                      syscall V_LHPMOUSE
02BA  BD 00CC               +                         mov     bp,V_LHPMOUSE
02BD  CD 6F                 +                         int     HP_ENTRY
02BF  1E                              push    ds
02C0  0E                              push    cs
02C1  1F                              pop     ds
                          ; write a message on display saying driver installed
02C2  8D 16 0220 R                    lea     dx, init_msg
02C6  B4 09                           mov     ah, 9
02C8  CD 21                           int     21H
02CA  1F                              pop     ds
02CB  FB                              sti
02CC  E9 0213 R                       jmp     all_ok                  ;all linked so all finished
02CF  C3                              ret
02D0                      init        ENDP    NEAR

02D0                      CODE        ends
                                      end


Macros:

                N a m e                 Length

MSD_HEADER . . . . . . . . . . . .      0006
SYSCALL. . . . . . . . . . . . . .      0002

Structures and records:

                N a m e                 Width   # fields
                                        Shift   Width   Mask    Initial

DESCRIBE . . . . . . . . . . . . .      0030    0018
  D_SOURCE . . . . . . . . . . . . .    0010
  D_HPHIL_ID . . . . . . . . . . . .    0011
  D_DESC_MASK. . . . . . . . . . . .    0012
  D_IO_MASK. . . . . . . . . . . . .    0013
  D_XDESC_MASK . . . . . . . . . . .    0014
  D_MAX_AXIS . . . . . . . . . . . .    0015
  D_CLASS. . . . . . . . . . . . . .    0016
  D_PROMPTS. . . . . . . . . . . . .    0017
  D_RESERVED . . . . . . . . . . . .    0018
  D_BURST_LEN. . . . . . . . . . . .    0019
  D_WR_REG . . . . . . . . . . . . .    001A
  D_RD_REG . . . . . . . . . . . . .    001B
  D_TRANSITION . . . . . . . . . . .    001C
  D_STATE. . . . . . . . . . . . . .    001D
  D_RESOLUTION . . . . . . . . . . .    001E
  D_SIZE_X . . . . . . . . . . . . .    0020
  D_SIZE_Y . . . . . . . . . . . . .    0022
  D_ABS_X. . . . . . . . . . . . . .    0024
  D_ABS_Y. . . . . . . . . . . . . .    0026
  D_REL_X. . . . . . . . . . . . . .    0028
  D_REL_Y. . . . . . . . . . . . . .    002A
  D_ACCUM_X. . . . . . . . . . . . .    002C
  D_ACCUM_Y. . . . . . . . . . . . .    002E
HP_SHEADER . . . . . . . . . . . .      0010    0009
  DH_ATR . . . . . . . . . . . . . .    0000
  DH_NAME_INDEX. . . . . . . . . . .    0002
  DH_V_DEFAULT . . . . . . . . . . .    0004
  DH_P_CLASS . . . . . . . . . . . .    0006
  DH_C_CLASS . . . . . . . . . . . .    0008
  DH_V_PARENT. . . . . . . . . . . .    000A
  DH_V_CHILD . . . . . . . . . . . .    000C
  DH_MAJOR . . . . . . . . . . . . .    000E
  DH_MINOR . . . . . . . . . . . . .    000F
MSD_INIT_CMD . . . . . . . . . . .      0017    0007
  MSD_UNIT_COUNT . . . . . . . . . .    000D
  MSD_END_OFFSET . . . . . . . . . .    000E
  MSD_END_SEG. . . . . . . . . . . .    0010
  MSD_BPB_OFFSET . . . . . . . . . .    0012
  MSD_BPB_SEG. . . . . . . . . . . .    0014
  MSD_1ST_UNIT . . . . . . . . . . .    0016
MSD_REQ_HEADER . . . . . . . . . .      0016    000A
  MSD_CMDLEN . . . . . . . . . . . .    0000
  MSD_UNIT . . . . . . . . . . . . .    0001
```

CCP_TO_GID_FILTER

```
    MSD_CMD. . . . . . . . . . . . . .      0002
    MSD_STATUS . . . . . . . . . . . .      0003
    MSD_MEDIA. . . . . . . . . . . . .      000D
    MSD_TRANS. . . . . . . . . . . . .      000E
    MSD_COUNT. . . . . . . . . . . . .      0012
    MSD_START. . . . . . . . . . . . .      0014

Segments and Groups:

                N a m e                     Size    Align   Combine Class

CGROUP . . . . . . . . . . . . . . . .      GROUP
  CODE . . . . . . . . . . . . . . . .      02D0    PARA    PUBLIC  'CODE'

Symbols:

                N a m e                     Type    Value   Attr

ALL_OK . . . . . . . . . . . . . . .        L NEAR  0213    CODE
ATR_CSHARE . . . . . . . . . . . . .        Number  0008
ATR_DEVCFG . . . . . . . . . . . . .        Number  4000
ATR_HP . . . . . . . . . . . . . . .        Number  8000
ATR_ISR. . . . . . . . . . . . . . .        Number  2000
ATR_LOG. . . . . . . . . . . . . . .        Number  0600
BAD_CMD. . . . . . . . . . . . . . .        L NEAR  01FB    CODE
BUILD_BPB. . . . . . . . . . . . . .        L NEAR  0213    CODE
BUTTON_DONE. . . . . . . . . . . . .        L NEAR  00DB    CODE
BUT_PROCESS. . . . . . . . . . . . .        L NEAR  00B9    CODE
BUT_PUSH . . . . . . . . . . . . . .        L NEAR  00D3    CODE
BUT_RELEASE. . . . . . . . . . . . .        L NEAR  00CD    CODE
CCP2GID_BAD_SYS_FN . . . . . . . . .        L NEAR  0120    CODE
CCP2GID_DESC . . . . . . . . . . . .        L NEAR  0187    CODE
CCP2GID_DESC_HEADR . . . . . . . . .        L 0010  0177    CODE
CCP2GID_DESC_SIZE. . . . . . . . . .        Number  0030
CCP2GID_DRIVER . . . . . . . . . . .        F PROC  002A    CODE    Length =0010
CCP2GID_HP_ATTR. . . . . . . . . . .        Number  E608
CCP2GID_INSTALLED. . . . . . . . . .        L FAR   0000    CODE
CCP2GID_ISR. . . . . . . . . . . . .        L NEAR  003A    CODE
CCP2GID_SYSTEM . . . . . . . . . . .        L NEAR  010D    CODE
CCP2GID_SYS_CASE . . . . . . . . . .        L NEAR  0123    CODE
CCP2GID_SYS_INIT . . . . . . . . . .        L NEAR  012B    CODE
CCP2GID_SYS_START. . . . . . . . . .        L NEAR  0147    CODE
CCP2GID_UNSUPPORTED. . . . . . . . .        L NEAR  0037    CODE
CCP_BUT1 . . . . . . . . . . . . . .        L NEAR  00AF    CODE
CCP_BUT2 . . . . . . . . . . . . . .        L NEAR  00B4    CODE
CCP_DOWN . . . . . . . . . . . . . .        L NEAR  008B    CODE
CCP_LEFT . . . . . . . . . . . . . .        L NEAR  0083    CODE
CCP_RIGHT. . . . . . . . . . . . . .        L NEAR  0093    CODE
CCP_UP . . . . . . . . . . . . . . .        L NEAR  007B    CODE
CL_CCP . . . . . . . . . . . . . . .        Number  2000
CL_LGID. . . . . . . . . . . . . . .        Number  0020
COMMAND_TABLE. . . . . . . . . . . .        L WORD  01B6    CODE
DESCRIBE_SIZE. . . . . . . . . . . .        Number  0030
DEV_CLOSE. . . . . . . . . . . . . .        L NEAR  0213    CODE
DEV_INT. . . . . . . . . . . . . . .        F PROC  01D6    CODE    Length =003D
DEV_OPEN . . . . . . . . . . . . . .        L NEAR  0213    CODE
DEV_STRATEGY . . . . . . . . . . . .        F PROC  01AB    CODE    Length =000B
D_ADDR_MASK. . . . . . . . . . . . .        Number  000F
D_BUFFER . . . . . . . . . . . . . .        Alias   D_SIZE_X
D_CCP_STATE. . . . . . . . . . . . .        Number  001E
D_CLASS_CURRENT. . . . . . . . . . .        Number  00F0
D_CLASS_DEFAULT. . . . . . . . . . .        Number  000F
D_REMAINDER_ACCUM. . . . . . . . . .        Alias   D_ACCUM_X
D_SAMPLE_ABSOLUTE. . . . . . . . . .        Alias   D_ABS_X
D_SAMPLE_RELATIVE. . . . . . . . . .        Alias   D_REL_X
D_SIZE . . . . . . . . . . . . . . .        Alias   D_SIZE_X
D_TYPE_MASK. . . . . . . . . . . . .        Number  00F0
EXIT_ISR . . . . . . . . . . . . . .        L NEAR  00F5    CODE
F_INS_FIXGETDS . . . . . . . . . . .        Number  000E
F_IO_CONTROL . . . . . . . . . . . .        Number  0004
F_ISR. . . . . . . . . . . . . . . .        Number  0000
F_SYSTEM . . . . . . . . . . . . . .        Number  0002
GIVE_TO_PARENT . . . . . . . . . . .        L NEAR  00EB    CODE
HP_ENTRY . . . . . . . . . . . . . .        Number  006F
INIT . . . . . . . . . . . . . . . .        N PROC  0286    CODE    Length =004A
INIT_BUT_STATE . . . . . . . . . . .        Number  00FF
INIT_MSG . . . . . . . . . . . . . .        L BYTE  0220    CODE
INIT_MSG2. . . . . . . . . . . . . .        L BYTE  0242    CODE
INIT_MSG3. . . . . . . . . . . . . .        L BYTE  0263    CODE
INPUT. . . . . . . . . . . . . . . .        L NEAR  0213    CODE
INT_EXIT . . . . . . . . . . . . . .        L NEAR  020A    CODE
IN_FLUSH . . . . . . . . . . . . . .        L NEAR  0213    CODE
IN_STAT. . . . . . . . . . . . . . .        L NEAR  0213    CODE
IOCTL_IN . . . . . . . . . . . . . .        L NEAR  0213    CODE
IOCTL_OUT. . . . . . . . . . . . . .        L NEAR  0213    CODE
MAX_CCP2GID_SYS_FN . . . . . . . . .        E BYTE  0006
MEDIA_CHECK. . . . . . . . . . . . .        L NEAR  0213    CODE
MSD_DONE_STATUS. . . . . . . . . . .        Number  0001
MSD_ERR_STATUS . . . . . . . . . . .        Number  0081
MSD_INIT . . . . . . . . . . . . . .        Number  0000
MSD_REM_MEDIA. . . . . . . . . . . .        Number  000F
MSD_UNKNOWN_CMD. . . . . . . . . . .        Number  0003
MSE_NUM_BUTTON . . . . . . . . . . .        Number  004C
ND_INPUT . . . . . . . . . . . . . .        L NEAR  0213    CODE
```

CCP__TO__GID__FILTER

```
OUTPUT . . . . . . . . . . . . . .          L NEAR  0213    CODE
OUT_FLUSH. . . . . . . . . . . . .          L NEAR  0213    CODE
OUT_STAT . . . . . . . . . . . . .          L NEAR  0213    CODE
OUT_VERIFY . . . . . . . . . . . .          L NEAR  0213    CODE
REL_MOVE . . . . . . . . . . . . .          L NEAR  009B    CODE
REM_MEDIA. . . . . . . . . . . . .          L NEAR  0213    CODE
RETURN_THE_FOLLOWING_RAM_TO_DOS. .          L FAR   0177    CODE
RH_OFF . . . . . . . . . . . . . .          L WORD  01A7    CODE
RH_SEG . . . . . . . . . . . . . .          L WORD  01A9    CODE
RS_DONE. . . . . . . . . . . . . .          Number  0006
RS_SUCCESSFUL. . . . . . . . . . .          Number  0000
RS_UNSUPPORTED . . . . . . . . . .          Number  0002
SAV_BX . . . . . . . . . . . . . .          L NEAR  0020    CODE
SAV_CX . . . . . . . . . . . . . .          L NEAR  0022    CODE
SAV_DX . . . . . . . . . . . . . .          L NEAR  0024    CODE
SAV_ES . . . . . . . . . . . . . .          L NEAR  0026    CODE
SF_MOUSE_OVERRIDE. . . . . . . . .          Number  0002
SF_START . . . . . . . . . . . . .          Number  0002
TOP_HP_ENTRY . . . . . . . . . . .          L NEAR  0028    CODE
T_KC_BUTTON. . . . . . . . . . . .          Number  0009
T_REL16. . . . . . . . . . . . . .          Number  0041
UNSUPPORTED. . . . . . . . . . . .          N PROC  0213    CODE    Length =000D
UP_DOWN_BIT. . . . . . . . . . . .          Number  0080
V_CCP. . . . . . . . . . . . . . .          Number  004E
V_CCPGID . . . . . . . . . . . . .          Number  00A2
V_DOLITTLE . . . . . . . . . . . .          Number  0006
V_LHPMOUSE . . . . . . . . . . . .          Number  00CC
V_SYSTEM . . . . . . . . . . . . .          Number  0012

 43048 Bytes free

 Warning Severe
 Errors  Errors
 0       0


ALL_OK . . . . . . . . . . . . . .      691#    772
ATR_CSHARE . . . . . . . . . . . .       69#    200
ATR_DEVCFG . . . . . . . . . . . .       70#    200
ATR_HP . . . . . . . . . . . . . .       71#    200
ATR_ISR. . . . . . . . . . . . . .       72#    200
ATR_LOG. . . . . . . . . . . . . .       73#    200

BAD_CMD. . . . . . . . . . . . . .      640     642     653#
BUILD_BPB. . . . . . . . . . . . .      610     677#
BUTTON_DONE. . . . . . . . . . . .      380     385     387#
BUT_PROCESS. . . . . . . . . . . .      361     365     367#
BUT_PUSH . . . . . . . . . . . . .      376     382#
BUT_RELEASE. . . . . . . . . . . .      378#

CCP2GID_BAD_SYS_FN . . . . . . . .      435     444#    453     454
CCP2GID_DESC . . . . . . . . . . .      560#
CCP2GID_DESC_HEADR . . . . . . . .      488     550#
CCP2GID_DESC_SIZE. . . . . . . . .      199#    486     491
CCP2GID_DRIVER . . . . . . . . . .      247#    257     743
CCP2GID_HP_ATTR. . . . . . . . . .      200#    550
CCP2GID_INSTALLED. . . . . . . . .      209#
CCP2GID_ISR. . . . . . . . . . . .      249     303#
CCP2GID_SYSTEM . . . . . . . . . .      252     432#
CCP2GID_SYS_CASE . . . . . . . . .      442     450#    455
CCP2GID_SYS_INIT . . . . . . . . .      451     481#
CCP2GID_SYS_START. . . . . . . . .      452     524#
CCP2GID_UNSUPPORTED. . . . . . . .      251     253#
CCP_BUT1 . . . . . . . . . . . . .      325     359#
CCP_BUT2 . . . . . . . . . . . . .      327     363#
CCP_DOWN . . . . . . . . . . . . .      320     341#
CCP_LEFT . . . . . . . . . . . . .      318     336#
CCP_RIGHT. . . . . . . . . . . . .      322     346#
CCP_UP . . . . . . . . . . . . . .      316     331#
CGROUP . . . . . . . . . . . . . .      205
CL_CCP . . . . . . . . . . . . . .       74#    550
CL_LGID. . . . . . . . . . . . . .       75#    550
CODE . . . . . . . . . . . . . . .      205     206#    206     207     776
COMMAND_TABLE. . . . . . . . . . .      606#    648

DESCRIBE . . . . . . . . . . . . .       77#    112     114
DESCRIBE_SIZE. . . . . . . . . . .      114#
DEV_CLOSE. . . . . . . . . . . . .      622     689#
DEV_INT. . . . . . . . . . . . . .      228     626#    669
DEV_OPEN . . . . . . . . . . . . .      621     688#
DEV_STRATEGY . . . . . . . . . . .      227     595#    600
DH_ATR . . . . . . . . . . . . . .       49#
DH_C_CLASS . . . . . . . . . . . .       53#
DH_MAJOR . . . . . . . . . . . . .       56#
DH_MINOR . . . . . . . . . . . . .       57#
DH_NAME_INDEX. . . . . . . . . . .       50#
DH_P_CLASS . . . . . . . . . . . .       52#
DH_V_CHILD . . . . . . . . . . . .       55#
DH_V_DEFAULT . . . . . . . . . . .       51#
DH_V_PARENT. . . . . . . . . . . .       54#    399     534
D_ABS_X. . . . . . . . . . . . . .      106#    118     354
```

CCP__TO__GID__FILTER

```
D_ABS_Y . . . . . . . . . . . . . .     107#    355
D_ACCUM_X . . . . . . . . . . . . .     110#    120
D_ACCUM_Y . . . . . . . . . . . . .     111#
D_ADDR_MASK . . . . . . . . . . . .     125#
D_BUFFER  . . . . . . . . . . . . .     121#
D_BURST_LEN . . . . . . . . . . . .      96#
D_CCP_STATE . . . . . . . . . . . .     116#
D_CLASS . . . . . . . . . . . . . .      89#
D_CLASS_CURRENT . . . . . . . . . .     122#
D_CLASS_DEFAULT . . . . . . . . . .     123#
D_DESC_MASK . . . . . . . . . . . .      85#
D_HPHIL_ID  . . . . . . . . . . . .      84#
D_IO_MASK . . . . . . . . . . . . .      86#
D_MAX_AXIS  . . . . . . . . . . . .      88#
D_PROMPTS . . . . . . . . . . . . .      92#
D_RD_REG  . . . . . . . . . . . . .     100#
D_REL_X . . . . . . . . . . . . . .     108#    119     352
D_REL_Y . . . . . . . . . . . . . .     109#    353
D_REMAINDER_ACCUM . . . . . . . . .     120#
D_RESERVED  . . . . . . . . . . . .      95#
D_RESOLUTION  . . . . . . . . . . .     103#
D_SAMPLE_ABSOLUTE . . . . . . . . .     118#
D_SAMPLE_RELATIVE . . . . . . . . .     119#
D_SIZE  . . . . . . . . . . . . . .     117#
D_SIZE_X  . . . . . . . . . . . . .     104#    117     121
D_SIZE_Y  . . . . . . . . . . . . .     105#
D_SOURCE  . . . . . . . . . . . . .      82#
D_STATE . . . . . . . . . . . . . .     102#    116     379     384
D_TRANSITION  . . . . . . . . . . .     101#    371
D_TYPE_MASK . . . . . . . . . . . .     126#
D_WR_REG  . . . . . . . . . . . . .      99#
D_XDESC_MASK  . . . . . . . . . . .      87#

EXIT_ISR  . . . . . . . . . . . . .     329     402#

F_INS_FIXGETDS  . . . . . . . . . .     128#    741
F_IO_CONTROL  . . . . . . . . . . .     129#    758
F_ISR . . . . . . . . . . . . . . .     131#    248     397
F_SYSTEM  . . . . . . . . . . . . .     132#    250     750

GIVE_TO_PARENT  . . . . . . . . . .     357     394     396#

HP_ENTRY  . . . . . . . . . . . . .      60#    401     747     755     762
HP_SHEADER  . . . . . . . . . . . .      48#     58      78

INIT  . . . . . . . . . . . . . . .     608     722#    774
INIT_BUT_STATE  . . . . . . . . . .     197#    574
INIT_MSG  . . . . . . . . . . . . .     703#    767
INIT_MSG2 . . . . . . . . . . . . .     709#
INIT_MSG3 . . . . . . . . . . . . .     715#
INPUT . . . . . . . . . . . . . . .     612     679#
INT_EXIT  . . . . . . . . . . . . .     649     658     662#
IN_FLUSH  . . . . . . . . . . . . .     615     682#
IN_STAT . . . . . . . . . . . . . .     614     681#
IOCTL_IN  . . . . . . . . . . . . .     611     678#
IOCTL_OUT . . . . . . . . . . . . .     620     687#

MAX_CCP2GID_SYS_FN  . . . . . . . .     434     455#
MEDIA_CHECK . . . . . . . . . . . .     609     676#
MSD_1ST_UNIT  . . . . . . . . . . .     174#
MSD_BPB_OFFSET  . . . . . . . . . .     172#
MSD_BPB_SEG . . . . . . . . . . . .     173#
MSD_CMD . . . . . . . . . . . . . .     150#    638
MSD_CMDLEN  . . . . . . . . . . . .     148#
MSD_COUNT . . . . . . . . . . . . .     160#
MSD_DONE_STATUS . . . . . . . . . .     181#    693
MSD_END_OFFSET  . . . . . . . . . .     170#    730
MSD_END_SEG . . . . . . . . . . . .     171#    732
MSD_ERR_STATUS  . . . . . . . . . .     180#    656
MSD_HEADER  . . . . . . . . . . . .     224
MSD_INIT  . . . . . . . . . . . . .     177#    639
MSD_INIT_CMD  . . . . . . . . . . .     164#    175
MSD_MEDIA . . . . . . . . . . . . .     157#
MSD_REM_MEDIA . . . . . . . . . . .     179#    641
MSD_REQ_HEADER  . . . . . . . . . .     147#    162
MSD_START . . . . . . . . . . . . .     161#
MSD_STATUS  . . . . . . . . . . . .     151#    657     695
MSD_TRANS . . . . . . . . . . . . .     158#
MSD_UNIT  . . . . . . . . . . . . .     149#
MSD_UNIT_COUNT  . . . . . . . . . .     169#
MSD_UNKNOWN_CMD . . . . . . . . . .     178#    655
MSE_NUM_BUTTON  . . . . . . . . . .     198#    539

ND_INPUT  . . . . . . . . . . . . .     613     680#

OUTPUT  . . . . . . . . . . . . . .     616     683#
OUT_FLUSH . . . . . . . . . . . . .     619     686#
OUT_STAT  . . . . . . . . . . . . .     618     685#
OUT_VERIFY  . . . . . . . . . . . .     617     684#

REL_MOVE  . . . . . . . . . . . . .     334     339     344     349     351#
REM_MEDIA . . . . . . . . . . . . .     623     690#
```

CCP_TO_GID_FILTER

```
RETURN_THE_FOLLOWING_RAM_TO_DOS.   547#   729
RH_OFF . . . . . . . . . . . . .   589#   597    637    654    663    694    728
RH_SEG . . . . . . . . . . . . .   590#   598
RS_DONE. . . . . . . . . . . . .   183#   328    408
RS_SUCCESSFUL. . . . . . . . . .   184#   497    542
RS_UNSUPPORTED . . . . . . . . .   185#   254    445

SAV_BX . . . . . . . . . . . . .   237#   306    374    388    403
SAV_CX . . . . . . . . . . . . .   238#   307    404
SAV_DX . . . . . . . . . . . . .   239#   308    405
SAV_ES . . . . . . . . . . . . .   240#   309    406
SF_MOUSE_OVERRIDE. . . . . . . .   130#   759
SF_START . . . . . . . . . . . .   133#   751
SYSCALL. . . . . . . . . . . . .   400    745    753    760
TOP_HP_ENTRY . . . . . . . . . .   241#   530    535
T_KC_BUTTON. . . . . . . . . . .   187#   393
T_REL16. . . . . . . . . . . . .   188#   356
UNSUPPORTED. . . . . . . . . . .   674#   697
UP_DOWN_BIT. . . . . . . . . . .   196#   375

V_CCP. . . . . . . . . . . . . .   194#   532
V_CCPGID . . . . . . . . . . . .   191#   398    534    550    550    742    754
V_DOLITTLE . . . . . . . . . . .   190#   550
V_LHPMOUSE . . . . . . . . . . .   192#   537    550    761
V_SYSTEM . . . . . . . . . . . .   193#   746

158 Symbols

54092 Bytes Free
```

G.9.2 Application Resident EX-BIOS Driver

This example demonstrates the use of an application resident EX-BIOS driver. The driver utilizes the Touch Screen logical device driver V__LTOUCH, and its associated event driver V__EVENT__TOUCH.

The driver utilizes V__LTOUCH to move the cursor around the screen. V__LTOUCH returns the current row and column address of the point the screen is being touched. The example driver in turn utilizes the STD-BIOS Video driver (INT 10H) to change to position of the displayed cursor to match the screen coordinates returned by V__LTOUCH.

This driver also utilizes the button state data returned by V__LTOUCH. When the screen is touched (a button make) the driver changes the shape of the cursor from an underline to a box or full character cell. The shape of the cursor is restored to an underline when the finger is removed (a button break).

Notice in the initialization section of the code that the CS:IP of the driver's service routine (TOUCH__HANDLER) and the driver's DS are substituted into the V__EVENT__TOUCH vector in the HP__VECTOR__TABLE. The existing contents of that vector are returned by the function. The driver stores these values in its data area, and restores them when the driver terminates (a '^' character is typed at the keyboard). All HP__VECTOR__TABLE vectors that are replaced with application program resident drivers should restore the original values in the vector when the application program terminates.

The listing for this driver can be found in Section 4.

G.9.3 Non-HP-HIL Input Devices

The next program listing is an example of how to integrate non-HP-HIL input devices into the Input System. This driver interfaces to an RS-232 mouse. It converts data frames received from the mouse into GID motion and button ISR Event Records. It integrates itself into the Input System by calling the V__SINPUT driver once these ISR Event Records have been constructed.

The PGID driver is the physical device driver for all devices inputting graphic motion and button state data. The initialization code must create a PGID driver for the V__SINPUT to pass the ISR Event Record. It builds a driver header and physical describe record, allocates a free HP__VECTOR__TABLE vector, and installs the PGID driver with V__LHPMOUSE as its parent driver.

The driver is structured as a DOS installable device driver. The COM port the mouse is connected to can be specified in the CONFIG.SYS command line.

RS-232 Mouse Driver

```
                              .286c
                              .LFCOND
                              PAGE 59,132
                              TITLE RS-232 MOUSE DRIVER
                              SUBTTL PREFACE
                              ;**********************************************************************
                              ;*                                                                    *
                              ;*                      RS-232 MOUSE DRIVER EXAMPLE                   *
                              ;*                                                                    *
                              ;**********************************************************************
                              ;
                              ;**********************************************************************
                              ;*                              DESCRIPTION                           *
                              ;**********************************************************************
                              ;
                              ; This driver illustrates the integration of non-HP-HIL devices into the
                              ; HP Vectra Input System.  This driver supports any mouse with an RS-232
                              ; interface, such as the MOUSE SYSTEMS mouse.  The driver is installed as
                              ; an MS-DOS device driver at boot time.
                              ;
                              ;The command line DEVICE=EXAMPLE.SYS [/n] should be entered in the
                              ;CONFIG.SYS file in the root directory of the boot drive.  If the optional
                              ;COM port number, /n is not included in the command line, the driver will
                              ;attempt to install the mouse on COM1.  If the optional COM port number is
                              ;present in the command line, the driver will attempt to install the mouse
                              ;on that COM port number.  The driver checks to make sure the port is present
                              ;and will issue an error message if a non-existent port number is specified.
                              ;
                              ;**********************************************************************
                              ;*                              CHANGE LOG                            *
                              ;**********************************************************************
                              ;
                              ;Revision A.01.01 - 12/02/85  SMM
                              ;
                              ;**********************************************************************
                              ;
                              SUBTTL EQUATES, RECORDS, AND DATA STRUCTURES
                              PAGE
                              ;**********************************************************************
                              ;**********************************************************************
                              ;                       EQUATES AND DATA STRUCTURES
                              ;**********************************************************************
                              ;**********************************************************************
                              ;
                              ;************************  GENERAL EQUATES  ************************
                              
 = 0000                       FALSE           EQU 0
 =-0001                       TRUE            EQU NOT FALSE
 =                            DEBUG           EQU TRUE
                              
                              ;*  MS-DOS INSTALLABLE DEVICE DRIVER EQUATES, RECORDS, AND STRUCTURES  **
                              
                              ;STRUCTURES
                              
                              REQ_HEADER      STRUC                   ;Initialization Request Header
                                                                      ;structure definition
 0000  ??                     RH_LENGTH       DB ?                    ;Length of Request Header.
 0001  ??                     RH_UNIT_CODE    DB ?                    ;Unit code.
 0002  ??                     RH_CMD_CODE     DB ?                    ;Command code.
 0003  ????                   RH_STATUS       DW ?                    ;Returned status.
 0005  ????????????????       RH_RESERVED     DQ ?                    ;Reserved for MS-DOS.
 000D  ??                     RH_UNIT_CNT     DB ?                    ;Unit count.
 000E  ????                   RH_END_OFF      DW ?                    ;Offset of ending address.
 0010  ????                   RH_END_SEG      DW ?                    ;Segment of ending address.
 0012  ????????               RH_BPB          DD ?                    ;BPB Pointer (not used).
 0016  ??                     RH_DRIV         DB ?                    ;Drive code (not used).
                              
 0017                         REQ_HEADER      ENDS
                              
 = 0012                       RH_CMD_LINE     EQU DWORD PTR RH_BPB    ;On INIT entry, points to CONFIG.SYS
                                                                      ;command line (i.e. all after DEVICE=).
                              
                              ;RECORDS
                              
                              ATTR RECORD  DEV:1, IOCTL:1, IBM:1, X:1, OCREM:1, Y:6, SPEC:1, CLK:1, NUL:1, STDO:1, STDI:1
                              
                                                                      ;DEV = 1 for character device, 0 for block device.
                                                                      ;IOCTL = 1 if IOCTL commands are supported.
                                                                      ;IBM = 1 if block device is in non-IBM format.
                                                                      ;X = Not used.
                                                                      ;OCREM = 1 if character device supports open and
                                                                      ;        close commands, 1 if block device has
                                                                      ;        removable media.
                                                                      ;Y = Not used.
                                                                      ;SPEC = 1 if INT 29H fast console I/O is installed.
                                                                      ;CLK = 1 if device is a clock device.
                                                                      ;NUL = 1 if device is a nul device.
                                                                      ;STDO = 1 if device is the Standard Output device.
                                                                      ;STDI = 1 if device is the Standard Input device.
                              
                              STATUS  RECORD ERROR:1, Z:5, BUSY:1, DONE:1, ERR_TYPE:8
```

```
                                                                      ;ERROR = 1 if error condition detected.
                                                                      ;Z = Not used.
                                                                      ;BUSY = 1 if device busy.
                                                                      ;DONE = 1 when command completed.
                                                                      ;ERR_TYPE = Error type.  See equates next.

                                        ;EQUATES

                                        ;Error codes.  Returned as part of status word defined above.

       = 0000                           MSD_WRITE_PROT          EQU 00H             ;write protect.
       = 0001                           MSD_UNKNOWN_UNIT        EQU 01H             ;unknown unit.
       = 0002                           MSD_NOT_RDY             EQU 02H             ;device not ready.
       = 0003                           MSD_UNKNOWN_CMD         EQU 03H             ;unknown command.
       = 0004                           MSD_CRC_ERROR           EQU 04H             ;CRC error.
       = 0005                           MSD_BAD_LENGTH          EQU 05H             ;bad driver request structure length.
       = 0006                           MSD_SEEK_ERROR          EQU 06H             ;seek error.
       = 0007                           MSD_UNKNOWN_MEDIA       EQU 07H             ;unknown media.
       = 0008                           MSD_SEC_NOT_FOUND       EQU 08H             ;sector not found.
       = 0009                           MSD_PAPER_OUT           EQU 09H             ;paper out.
       = 000A                           MSD_WRITE_FAULT         EQU 0AH             ;write fault.
       = 000B                           MSD_READ_FAULT          EQU 0BH             ;read fault.
       = 000C                           MSD_GEN_FAILURE         EQU 0CH             ;general failure.

                                        ;Commands.

       = 0000                           MSD_INIT        EQU 00H                     ;Initialize.
       = 0001                           MSD_MEDIA_CHK   EQU 01H                     ;Media check.
       = 0002                           MSD_BLD_BPB     EQU 02H                     ;Build BIOS Parameter Block (BPB).
       = 0003                           MSD_IOCTL_IN    EQU 03H                     ;IOCTL input.
       = 0004                           MSD_INPUT       EQU 04H                     ;Input from device.
       = 0005                           MSD_IN_NOWAIT   EQU 05H                     ;Non-destructive, no-wait input.
       = 0006                           MSD_IN_STATUS   EQU 06H                     ;Return status of input device.
       = 0007                           MSD_IN_FLUSH    EQU 07H                     ;Flush input buffer.
       = 0008                           MSD_OUTPUT      EQU 08H                     ;Output to device.
       = 0009                           MSD_OUT_VERIFY  EQU 09H                     ;Output with verify to device.
       = 000A                           MSD_OUT_STATUS  EQU 0AH                     ;Return status of output device.
       = 000B                           MSD_OUT_FLUSH   EQU 0BH                     ;Flush output buffer.
       = 000C                           MSD_IOCTL_OUT   EQU 0CH                     ;IOCTL output.
       = 000D                           MSD_DEV_OPEN    EQU 0DH                     ;Open device.
       = 000E                           MSD_DEV_CLOSE   EQU 0EH                     ;Close device.
       = 000F                           MSD_REM_MEDIA   EQU 0FH                     ;Removable media check.

                                        ;MS-DOS equates.

       = 0009                           PRINT_STR       EQU 09H                     ;MS-DOS print string function number.
       = 0021                           DOS_ENTRY       EQU 21H                     ;MS-DOS interrupt.

                                        ;ASCII equates.

       = 000A                           LF              EQU 0AH
       = 000D                           CR              EQU 0DH

                                        ;***********  EX-BIOS DRIVER EQUATES, RECORDS, AND STRUCTURES  **********

                                        ;STRUCTURES

                                        HP_HEADER       STRUC                       ;HP Driver Header.

       0000  0000                       DH_ATR          DW 0                        ;Driver attribute.
       0002  0000                       DH_NAME_INDEX   DW 0                        ;Index number for driver string.
       0004  0000                       DH_V_DEFAULT    DW 0                        ;????
       0006  0000                       DH_P_CLASS      DW 0                        ;Driver parent class.
       0008  0000                       DH_C_CLASS      DW 0                        ;Driver child class.
       000A  0000                       DH_V_PARENT     DW 0                        ;Vector number of driver's parent.
       000C  0000                       DH_V_CHILD      DW 0                        ;Vector number of driver's child.
       000E  00                         DH_MAJOR        DB 0                        ;Major address of device.
       000F  00                         DH_MINOR        DB 0                        ;Minor address of device.

       0010                             HP_HEADER       ENDS


                                        DESCRIBE        STRUC                       ;Physical describe record.

       0000  ??                         D_SOURCE        DB ?                        ;Upper nibble contains GID type.
                                                                                    ;Lower nibble HP-HIL address.
       0001  ??                         D_HPHIL_ID      DB ?                        ;Device ID byte returned by HP-HIL device.
       0002  ??                         D_DESC_MASK     DB ?                        ;????
       0003  ??                         D_IO_MASK       DB ?                        ;I/O descriptor byte from device.
       0004  ??                         D_XDESC_MASK    DB ?                        ;Extended descriptor byte from device.
       0005  ??                         D_MAX_AXIS      DB ?                        ;Maximum number of axes reported by device.
       0006  ??                         D_CLASS         DB ?                        ;Device class.
                                                                                    ;Upper nibble contains current class.
                                                                                    ;Lower nibble contain default class.
       0007  ??                         D_PROMPTS       DB ?                        ;Number of buttons/prompts
                                                                                    ;Upper nibble contains number of prompts.
                                                                                    ;Lower nibble contains number of buttons.
       0008  ??                         D_RESERVED      DB ?                        ;Reserved.
       0009  ??                         D_BURST_LEN     DB ?                        ;Maximum burst length.
       000A  ??                         D_WR_REG        DB ?                        ;Number of write registers supported.
       000B  ??                         D_RD_REG        DB ?                        ;Number of read registers supported.
       000C  ??                         D_TRANSITION    DB ?                        ;Transitions reported per button.
       000D  ??                         D_STATE         DB ?                        ;Current state of buttons.
       000E  ????                       D_RESOLUTION    DW ?                        ;Counts/cm returned by device.
```

```
0010  ????                          D_SIZE_X       DW ?                   ;Maximum count along X axis in units of resolution.
0012  ????                          D_SIZE_Y       DW ?                   ;Maximum count along Y axis in units of resolution.
0014  ????                          D_ABS_X        DW ?                   ;Absolute data device X motion.
0016  ????                          D_ABS_Y        DW ?                   ;Absolute data device Y motion.
0018  ????                          D_REL_X        DW ?                   ;Relative data device X motion.
001A  ????                          D_REL_Y        DW ?                   ;Relative data device Y motion.
001C  ????                          D_ACCUM_X      DW ?                   ;X axis scaling accumulator.
001E  ????                          D_ACCUM_Y      DW ?                   ;Y axis scaling accumulator.

0020                                DESCRIBE       ENDS

= 004C                              MSE_NUM_BUTTON          equ     004CH   ;Offset of number of button in mouse RAM

                                    ;RECORDS

                                    HP_ATTR RECORD HP:1, DEVCFG:1, ISR:1, ENTRY:1, TYPE:3, STR:1, MAP_CALL:1, A:1, SUBADD:2, PS
                              HARE:1, CSHARE:1, ROM:1, B:1

                                    ;EQUATES

                                    ;EX-BIOS driver vector addresses and driver function numbers.

= 0006                              V_DOLITTLE     EQU 0006H              ;DOLITTLE driver vector address (NUL driver).

= 0012                              V_SYSTEM       EQU 0012H              ;SYSTEM driver vector address.
= 0004                              F_INS_BASEHPVT EQU 04H
= 000A                              F_INS_XCHGFREE EQU 0AH

= 002A                              V_SINPUT       EQU 002AH              ;INPUT driver vector address
= 0000                              F_ISR          EQU 00H
= 0002                              F_SYSTEM       EQU 02H
= 0004                              F_IO_CONTROL   EQU 04H

= 000C                              F_INQUIRE_ENTRY EQU 0CH               ; inquire about PGID CS:IP

= 00CC                              V_LHPMOUSE     EQU 00CCH              ;LHPMOUSE driver vector address.
= 0002                              SF_MOUSE_OVERRIDE EQU 02H

= 006F                              HP_ENTRY       EQU 6FH                ;EX-BIOS interrupt number.

                                    ;ISR Event Record data types.

= 0009                              T_KC_BUTTON    EQU 09H                ;Button data type.
= 0040                              T_REL08        EQU 40H
= 0041                              T_REL16        EQU 41H                ;16 bit relative motion data type.
= 0042                              T_ABS08        EQU 42H
= 0043                              T_ABS16        EQU 43H

                                    ;EX-BIOS Return Status Codes

= 0000                              RS_SUCCESSFUL  EQU 00H
= 0002                              RS_UNSUPPORTED EQU 02H
= 0006                              RS_DONE        EQU 06H
= 00FE                              RS_FAIL        EQU 0FEH
= 00F6                              RS_NO_VECTOR   EQU 0F6H

                                    ;*********************************************************************

                                    SUBTTL CODE SEGMENT
                                    PAGE
                                    ;*********************************************************************
                                    ;*********************************************************************
                                    ;                                   CODE SEGMENT
                                    ;*********************************************************************
                                    ;*********************************************************************

0000                                CODE    SEGMENT PUBLIC 'CODE'

                                            ASSUME  CS:CODE, DS:NOTHING
0000                                        ORG     0                     ;Must be org'd at 0 to be a device driver.

0000                                DEV_DRIVER PROC FAR

                                    ;********************** MS-DOS DEVICE DRIVER HEADER ******************

0000  FF FF FF FF                                  DD -1                  ;Link list entry. Must be set to -1
0004  8000                          DRIVER_ATTR    DW ATTR<1,0,0,0,0,0,0,0,0,0> ;Driver attribute.
0006  0265 R                        STRAT_ENT      DW OFFSET DEV_STRATEGY  ;Device strategy entry point.
0008  0270 R                        INT_ENT        DW OFFSET DEV_INTERRUPT ;Device interrupt entry point.
000A  20 32 33 32 4D 53             DRIVER_NAME    DB ' 232MSE '
      45 20

                                    ;********** EX-BIOS DRIVER HEADER AND PHYSICAL DESCRIBE RECORD ********

0020                                               ORG 20H                ;Make sure its paragraph aligned.

= AC18                              DEV_ATTR       EQU HP_ATTR<1,0,1,0,6,0,0,0,1,1,0,0>
0020  AC18                          DEV_HEADER     HP_HEADER<DEV_ATTR,3,0,0,0,V_LHPMOUSE,V_DOLITTLE,0,0>
0022  0003
0024  0000
0026  0000
0028  0000
002A  00CC
```

```
002C  0006
002E  00
002F  00

0030  02                        DEV_DESCRIBE    DESCRIBE <2,0,0,0,0,2,0,20H,0,0,0,0,1,0FFH,200D,0,0,0,0,0,0,0,0>
0031  00
0032  00
0033  00
0034  00
0035  02
0036  00
0037  20
0038  00
0039  00
003A  00
003B  00
003C  01
003D  FF
003E  00C8
0040  0000
0042  0000
0044  0000
0046  0000
0048  0000
004A  0000
004C  0000
004E  0000


                                ;*********************************************************************
                                ;*                        CODE SEGMENT RELATIVE DATA AREA                  *
                                ;*********************************************************************
                                ;
                                ;*****************  DATA AREA FOR MS-DOS DRIVER PORTION  ****************

0050  0000                      REQ_HDR_OFF     DW 0                          ;Storage for offset of device strategy header.
0052  0000                      REQ_HDR_SEG     DW 0                          ;Storage for segment of device strategy header.

0054  52 53 2D 32 33 32         SIGN_ON_MSG     DB 'RS-232 INPUT SYSTEM MOUSE DRIVER  '
      20 49 4E 50 55 54
      20 53 59 53 54 45
      4D 20 4D 4F 55 53
      45 20 44 52 49 56
      45 52 20 20
0076  28 43 29 43 6F 70                         DB '(C)Copyright Hewlett-Packard 1985',CR,LF
      79 72 69 67 68 74
      20 48 65 77 6C 65
      74 74 2D 50 61 63
      6B 61 72 64 20 31
      39 38 35 0D 0A
0099  56 65 72 73 69 6F         VERSION_LAB     DB 'Version A.01.01',CR,LF,'$'
      6E 20 41 2E 30 31
      2E 30 31 0D 0A 24
= 0010                          VERSION_LEN     EQU $-VERSION_LAB-2
00AB  4D 6F 75 73 65 20         OK_MSG          DB 'Mouse installed on COM'
      69 6E 73 74 61 6C
      6C 65 64 20 6F 6E
      20 43 4F 4D
00C1  30 3A 0D 0A 0D 0A         COM_MSG         DB '0:',CR,LF,CR,LF,'$'
      24
00C8  53 70 65 63 69 66         NO_PORT_MSG     DB 'Specified COM port not present.  Driver not installed.',CR,LF,CR,LF,
      69 65 64 20 43 4F
      4D 20 70 6F 72 74
      20 6E 6F 74 20 70
      72 65 73 65 6E 74
      2E 20 20 44 72 69
      76 65 72 20 6E 6F
      74 20 69 6E 73 74
      61 6C 6C 65 64 2E
      0D 0A 0D 0A 24
0103  55 6E 61 62 6C 65         NO_VECTOR       DB 'Unable to install PGID driver.',CR,LF,'$'
      20 74 6F 20 69 6E
      73 74 61 6C 6C 20
      50 47 49 44 20 64
      72 69 76 65 72 2E
      0D 0A 24

0124  0000                      STACK_PTR       DW 0                          ;Storage for existing stack frame.
0126  0000                      STACK_SEG       DW 0

0128  0000                      COM_NUMBER      DW 0                          ;Offset into COM port base address table
                                                                              ;found at 0040:0000H.
012A  0030                      INT_TABLE       DW 0CH * 4                    ;COM1 port interrupt.
012C  002C                                      DW 0BH * 4                    ;COM2 port interrupt.
012E  0030                                      DW 0CH * 4                    ;COM3 port interrupt - set as appropriate.
0130  002C                                      DW 0BH * 4                    ;COM4 port interrupt - set as appropriate.
0132  FFEF                      MASK_TABLE      DW NOT 01H SHL 4              ;COM1 interrupt mask (IRQ4).
0134  FFF7                                      DW NOT 01H SHL 3              ;COM2 interrupt mask (IRQ3)
0136  FFEF                                      DW NOT 01H SHL 4              ;COM3 interrupt mask (IRQ4).
0138  FFF7                                      DW NOT 01H SHL 3              ;COM4 interrupt mask (IRQ3).

013A  0000                      FRAME_COUNT     DW 0                          ;Frame counter for mouse data packet.
013C     05 [                   TEMP_BUFFER     DB 5 DUP (0)                  ;Temporary buffer for mouse data bytes.
                 00
                    ]
```

```

0141  87                      LAST_SYNCH     DB 87H                   ;Copy of last synch byte.

0142     0E [                 HPHIL_TABLE    DB 14 DUP (0)            ;HP-HIL configuration table.
              00
                   ]

0150  00                      HPHIL_ADD      DB 0                     ;HP-HIL 'address' of mouse.
0151  00                      PGID_VECT_NUM  DB 0                     ;HP_VECTOR_TABLE vector address of PGID.

                              ;JUMP TABLE FOR MS-DOS DRIVER COMMANDS

0152  02A7 R                  CMD_TABLE      DW OFFSET INIT_CODE      ;Initialize driver.
0154  0292 R                                 DW OFFSET UNSUPPORT_CMD  ;Media check.
0156  0292 R                                 DW OFFSET UNSUPPORT_CMD  ;Build BPB.
0158  0292 R                                 DW OFFSET UNSUPPORT_CMD  ;IOCTL input.
015A  0292 R                                 DW OFFSET UNSUPPORT_CMD  ;Input.
015C  0292 R                                 DW OFFSET UNSUPPORT_CMD  ;Non-destructive input.
015E  0292 R                                 DW OFFSET UNSUPPORT_CMD  ;Input status.
0160  0292 R                                 DW OFFSET UNSUPPORT_CMD  ;Flush input buffer.
0162  0292 R                                 DW OFFSET UNSUPPORT_CMD  ;Output.
0164  0292 R                                 DW OFFSET UNSUPPORT_CMD  ;Output with verify.
0166  0292 R                                 DW OFFSET UNSUPPORT_CMD  ;Output status.
0168  0292 R                                 DW OFFSET UNSUPPORT_CMD  ;Flush output buffer.
016A  0292 R                                 DW OFFSET UNSUPPORT_CMD  ;IOCTL output.
016C  0292 R                                 DW OFFSET UNSUPPORT_CMD  ;Open device.
016E  0292 R                                 DW OFFSET UNSUPPORT_CMD  ;Close device.
0170  0292 R                                 DW OFFSET UNSUPPORT_CMD  ;Removable media check.

                              ;****************  DATA AREA FOR EX-BIOS DRIVER PORTION  **************

                              PAGE

                              ;*********************************************************************
                              ;*                          MOUSE DRIVER CODE                        *
                              ;*********************************************************************

0172                          MOUSE_INT:

                              ;PRESERVE MACHINE STATE

0172  9C                              PUSHF                           ;Save the registers.
0173  60                              PUSHA
0174  1E                              PUSH   DS
0175  06                              PUSH   ES
0176  8C C8                           MOV    AX,CS                    ;Re-establish data segment addressibility.
0178  8E D8                           MOV    DS,AX

                              ;ISSUE END-OF-INTERRUPT TO 8259A

017A  B0 20                           MOV    AL,20H                   ;EOI
017C  E6 20                           OUT    20H,AL

                              ,GET CHARACTER FROM MOUSE

017E  B8 0040                         MOV    AX,40H                   ;Get base address of COM port from table.
0181  8E C0                           MOV    ES,AX
0183  2E: 8B 1E 0128 R                MOV    BX,COM_NUMBER
0188  26: 8B 17                       MOV    DX,ES:[BX]

018B  EC                              IN     AL,DX                    ;Get character.

                              ;STORE IN TEMPORARY BUFFER UNTIL ENTIRE FRAME HAS BEEN RECEIVED

018C  2E: 8B 1E 013A R                MOV    BX,FRAME_COUNT           ;Get number of characters left in frame.
0191  0B DB                           OR     BX,BX                    ;See if we're looking for synch byte.
0193  75 0D                           JNZ    MSI_1                    ;Jump if not.
0195  8A E0                           MOV    AH,AL                    ;Save a copy of mouse character.
0197  24 F8                           AND    AL,0F8H                  ;Mask off button bits.
0199  3C 80                           CMP    AL,80H                   ;See if this is a synch byte.
019B  8A C4                           MOV    AL,AH                    ;Get the original character back.
019D  74 03                           JZ     MSI_1                    ;Put character in temporary buffer if synch
                                                                      ;byte is valid.
019F  E9 0260 R                       JMP    MSI_5                    ;Otherwise, throw character away.

01A2  2E: 88 87 013C R        MSI_1:  MOV    TEMP_BUFFER[BX],AL       ;Store character away.
01A7  43                              INC    BX                       ,Update the frame counter.
01A8  2E: 89 1E 013A R                MOV    FRAME_COUNT,BX           ;And save it.
01AD  83 FB 05                        CMP    BX,5                     ;Is this the last character in frame?
01B0  74 03                           JZ     MSI_2                    ;Process the frame if so.
01B2  E9 0260 R                       JMP    MSI_5                    ;Otherwise, skip on.

                              ;CHECK FOR A CHANGE IN BUTTON STATE

01B5  BB 0000                 MSI_2:  MOV    BX,0                     ;New character count.
01B8  2E: 89 1E 013A R                MOV    FRAME COUNT,BX           ;Store it.
01BD  2E: 8A 87 013C R                MOV    AL,TEMP_BUFFER[BX]       ;Get synch byte.
01C2  2E  8A 26 0141 R                MOV    AH,LAST_SYNCH            ;Get last synch byte.
01C7  2E  A2 0141 R                   MOV    LAST_SYNCH,AL            ,Update last byte.
01CB  3A E0                           CMP    AH,AL                    ;See if they are the same.
01CD  74 56                           JZ     MSI_3                    ;Skip on if so (no change in button state).

                              ;SEND BUTTON ISR EVENT RECORD(S) TO INPUT SYSTEM
```

```

01CF  53                          PUSH    BX                      ;Save frame counter.
01D0  52                          PUSH    DX                      ;Save
01D1  32 E0                       XOR     AH,AL                   ;AH now holds mask of buttons that have changed.
01D3  B7 01                       MOV     BH,01H                  ;Mask for first button.
01D5  B9 0003                     MOV     CX,3                    ;Number of buttons to process.

01D8                      MBUTTON:
01D8  8A DC                       MOV     BL,AH                   ;Get a copy of change mask.
01DA  22 DF                       AND     BL,BH                   ;See if selected button was the one that changed.
01DC  74 41                       JZ      MNEXT_BUTTON            ;Skip on if not.
01DE  84 F8                       TEST    BH,AL                   ;Determine state (make or break) of selected butt

01E0  74 04                       JZ      MBUTTON_DOWN

01E2                      MBUTTON_UP:

01E2  B3 80                       MOV     BL,80H                  ;Set bit 7 (make/break bit) to 0 (break).
01E4  EB 02                       JMP     SHORT MBUTTON_ISR

01E6                      MBUTTON_DOWN:

01E6  B3 00                       MOV     BL,00H                  ;Set bit 7 (make/break bit) to 1 (make).

01E8                      MBUTTON_ISR:
01E8  53                          PUSH    BX
01E9  8B D9                       MOV     BX, CX
01EB  32 FF                       XOR     BH, BH
01ED  FE CB                       DEC     BL
01EF  2E: 8A 8F 01F7 R            MOV     CL, CS:BUTTON_TAB[BX]
01F4  5B                          POP     BX
01F5  EB 03                       JMP     SHORT BISR2
01F7  00                  BUTTON_TAB      DB      0               ; left button
01F8  02                                  DB      2               ; middle button
01F9  01                                  DB      1               ; right button
01FA                      BISR2:
01FA  0A D9                       OR      BL, CL
01FC  32 FF                       XOR     BH, BH                  ; clear out bh

01FE  50                          PUSH    AX                      ;Save registers.
01FF  53                          PUSH    BX
0200  51                          PUSH    CX
0201  1E                          PUSH    DS

                                  page
                          ;Create ISR Event Record

0202  B6 09                       MOV     DH,T_KC_BUTTON          ;Set data type
0204  2E: 8A 16 0151 R            MOV     DL,PGID_VECT_NUM        ;Get vector number of mouse's PGID.
0209  B9 0000                     MOV     CX,0                    ;Burst length (N/A).
020C  8C C8                       MOV     AX,CS                   ;Point ES:0 to driver header.
020E  40                          INC     AX
020F  40                          INC     AX
0210  8E C0                       MOV     ES,AX
0212  B4 00                       MOV     AH,F_ISR                ;Set ISR function.
0214  BD 002A                     MOV     BP,V_SINPUT             ;We're calling the INPUT driver.
0217  FA                          CLI                             ;Turn off interrupts while we're out.
0218  CD 6F                       INT     HP_ENTRY
021A  FB                          STI                             ;Re-enable interrupts.

021B  1F                          POP     DS
021C  59                          POP     CX
021D  5B                          POP     BX
021E  58                          POP     AX

021F                      MNEXT_BUTTON:

021F  D0 E7                       SHL     BH,1                    ;Move button selector mask to next button.
0221  E2 B5                       LOOP    MBUTTON
0223  5A                          POP     DX                      ;Restore
0224  5B                          POP     BX                      ;Get frame counter back.

                          ;CHECK FOR MOTION

0225  43                  MSI_3:  INC     BX                      ;Point to first delta X in buffer.
0226  2E: 8A 97 013C R            MOV     DL,TEMP_BUFFER[BX]
022B  43                          INC     BX                      ;Get first delta Y.
022C  2E: 8A B7 013C R            MOV     DH,TEMP_BUFFER[BX]
0231  43                          INC     BX                      ;Add second delta X to first.
0232  2E: 02 97 013C R            ADD     DL,TEMP_BUFFER[BX]
0237  43                          INC     BX                      ;Add second delta Y to first.
0238  2E: 02 B7 013C R            ADD     DH,TEMP_BUFFER[BX]

023D  0B D2               MSI_4:  OR      DX,DX                   ;Check for zero motion.
023F  74 1F                       JZ      MSI_5                   ;Skip on if none detected.

                                  page
                          ;SEND MOTION ISR EVENT RECORD TO INPUT SYSTEM

0241  8A C2                       MOV     AL,DL                   ;Convert delta X to 16 bit value and put
0243  98                          CBW                             ;it in ISR Event Record (BX register).
0244  8B D8                       MOV     BX,AX
0246  8A C6                       MOV     AL,DH                   ;Ditto for delta Y (CX register).
0248  98                          CBW
0249  8B C8                       MOV     CX,AX
```

```

                                ;Create motion ISR event record
0248  B6 41                             MOV     DH,T_REL16              ;Set ISR Event record data type to 16 bit
                                                                        ;relative motion.
024D  2E: 8A 16 0151 R                  MOV     DL,PGID_VECT_NUM        ;Get vector number of mouse's PGID.
0252  8C C8                             MOV     AX,CS                   ;Set ES:0 to driver header.
0254  40                                INC     AX
0255  8E C0                             MOV     ES,AX
0257  B4 00                             MOV     AH,F_ISR                ;Select ISR function.
0259  BD 002A                           MOV     BP,V_SINPUT             ;We're passing this on to the INPUT driver.
025C  FA                                CLI                             ;Interrupts are supposed to be off.
025D  CD 6F                             INT     HP_ENTRY
025F  FB                                STI                             ;Turn interrupts back on now.

                                ;RESTORE MACHINE STATE AND EXIT

0260  07                        MSI_5:  POP     ES
0261  1F                                POP     DS
0262  61                                POPA
0263  9D                                POPF
0264  CF                                IRET

                                PAGE

                                ;***********************************************************************
                                ;*                       MS-DOS DRIVER CODE                            *
                                ;***********************************************************************

                                ;**********************  STRATEGY ENTRY POINT  **********************

0265                            DEV_STRATEGY PROC FAR
0265  2E: 89 1E 0050 R                  MOV     CS:REQ_HDR_OFF,BX       ;Save offset of request header.
026A  2E: 8C 06 0052 R                  MOV     CS:REQ_HDR_SEG,ES       ;Save segment of request header.
026F  CB                                RET                             ;Return to MS-DOS.
0270                            DEV_STRATEGY ENDP

                                ;*********************  INTERRUPT ENTRY POINT  **********************

0270                            DEV_INTERRUPT PROC FAR

                                ;SAVE MACHINE STATE
0270  9C                                PUSHF
0271  FC                                CLD
0272  60                                PUSHA                           ;Save registers.
0273  8C CF                             MOV     DI,CS                   ;Set DS to CS.
0275  8E DF                             MOV     DS,DI

                                ;FETCH COMMAND FROM REQUEST HEADER
0277  2E: C4 3E 0050 R                  LES     DI,DWORD PTR REQ_HDR_OFF ;Move address of request header into ES:DI.
027C  26: 8A 45 02                      MOV     AL,ES:[DI].RH_CMD_CODE  ;Get command byte from header.
0280  3C 00                             CMP     AL,MSD_INIT             ;Perform range check on command byte.
0282  72 0E                             JB      BAD_CMD
0284  3C 0F                             CMP     AL,MSD_REM_MEDIA
0286  77 0A                             JA      BAD_CMD
0288  98                                CBW                             ;Convert command into jump table offset.
0289  D1 E0                             SHL     AX,1
028B  8B D8                             MOV     BX,AX
028D  2E: FF A7 0152 R                  JMP     CMD_TABLE[BX]           ;Dispatch to requested function.

                                ;EXIT POINT FOR BAD OR UNSUPPORTED FUNCTIONS

0292                            BAD_CMD:
0292                            UNSUPPORT_CMD:

0292  26: 81 4D 03 8000                 OR      ES:[DI].RH_STATUS, MASK ERROR ;Set error flag in return status word.
0298  26: 81 4D 03 0003                 OR      ES:[DI].RH_STATUS, MSD_UNKNOWN_CMD ;Set error code.

                                ;COMMON EXIT POINT

029E  26: 81 4D 03 0100         EXIT:   OR      ES:[DI].RH_STATUS, MASK DONE ;Set return status to done.
02A4  61                                POPA                            ;Restore registers.
02A5  9D                                POPF                            ;Restore flags.
02A6  CB                                RET                             ;Return to MS-DOS.


                                ;*********************  END OF RESIDENT CODE  **********************
                                PAGE
                                ;*********************  INITIALIZATION CODE  **********************

02A7                            INIT_CODE:

                                ;SET UP LOCAL STACK
02A7  FA                                CLI                             ;Disable interrupts while we're messing with stack.

02A8  BE 0124 R                         MOV     SI,OFFSET STACK_PTR     ;Store existing stack environment.
02AB  89 24                             MOV     [SI],SP
02AD  83 C6 02                          ADD     SI,2
02B0  8C 14                             MOV     [SI],SS

02B2  BC 0511 R                         MOV     SP,OFFSET CS:STACK_TOP  ;Set up our local stack.
02B5  8C C8                             MOV     AX,CS                   ;Stack segment is same as code (CS).
02B7  8E D0                             MOV     SS,AX

02B9  FB                                STI                             ;Re-enable interrupts.
```

```

                                      ;PRINT SIGN-ON MESSAGE

 02BA  BA 0054 R                              MOV     DX,OFFSET SIGN_ON_MSG
 02BD  B4 09                                  MOV     AH,PRINT_STR
 02BF  CD 21                                  INT     DOS_ENTRY

                                      ;PARSE CONFIG.SYS COMMAND LINE TO DETERMINE WHICH COM PORT THE MOUSE IS ON

 02C1  BB 0000                                MOV     BX,0                    ;Clear BX.  It will be used as index into
                                                                              ;command line.
 02C4  26: C4 7D 12                           LES     DI,ES:[DI].RH_CMD_LINE  ;Load ES:DI with pointer to CONFIG.SYS command
                                                                              ;line.

 02C8  26: 8A 01                      IC_1:   MOV     AL,BYTE PTR ES:[DI+BX]  ;Get next character in command line.
 02CB  3C 2F                                  CMP     AL,'/'                  ;Check for backslash.
 02CD  74 0B                                  JZ      IC_2                    ;If found, indicates start of parameters.
 02CF  3C 0D                                  CMP     AL,CR                   ;Check for carriage return. (Indicates a bogas
                                                                              ;set of parameters).
 02D1  74 1C                                  JZ      IC_3                    ;If found, stop scanning command line.
 02D3  3C 0A                                  CMP     AL,LF                   ;Check for line feed. (Indicates no parameters
                                                                              ;entered in command line.
 02D5  74 18                                  JZ      IC_3                    ;If found, stop scanning command line.
 02D7  43                                     INC     BX                      ;Else, point to next character,
 02D8  EB EE                                  JMP     IC_1                    ;and continue scanning command line.

 02DA  43                             IC_2:   INC     BX                      ;Get next character.  Should indicate COM port
 02DB  26: 8A 01                              MOV     AL,BYTE PTR ES:[DI+BX]  ;to use.  Valid range is 1 - 4.
 02DE  2C 31                                  SUB     AL,'1'                  ;Convert number into offset from 1.
 02E0  72 0D                                  JB      IC_3                    ;Perform range check on results.
 02E2  3C 03                                  CMP     AL,3
 02E4  77 09                                  JA      IC_3
 02E6  98                                     CBW                             ;Convert into offset into STD-BIOS COM port
                                                                              ;base address table at 0040:0000H.
 02E7  D1 E0                                  SHL     AX,1
 02E9  2E: A3 0128 R                          MOV     COM_NUMBER,AX           ;Save it for future use.
 02ED  EB 07                                  JMP     SHORT IC_4

 02EF  2E: C7 06 0128 R 0000          IC_3:   MOV     COM_NUMBER,0            ;If we wind up here, there were no parameters
                                                                              ;specified in the command line, or an invalid
                                                                              ;COM port was specified.  Set COM port COM1
                                                                              ;default.

 02F6  8B D8                          IC_4:   MOV     BX,AX                   ;Convert offset into ASCII COM number (1 - 4).
 02F8  D1 EB                                  SHR     BX,1
 02FA  80 C3 31                               ADD     BL,'1'
 02FD  2E: 88 1E 00C1 R                       MOV     COM_MSG,BL              ;Store in sign-on message.
 0302  FA                                     CLI                             ;Disable interrupts while mouse interrupt
                                                                              ;is being set up.


                                      ;INITIALIZE SERIAL PORT PARAMETERS

 0303  8B F8                                  MOV     DI,AX                   ;Move COM port table offset into DI.
 0305  B8 0040                                MOV     AX,40H                  ;Segment address of COM port base address table.
 0308  8E C0                                  MOV     ES,AX
 030A  26: 8B 15                              MOV     DX,ES:[DI]              ;Get base address of COM port out of table.
 030D  0B D2                                  OR      DX,DX                   ;Make sure port exists.
 030F  75 03                                  JNZ     IC_4A                   ;Continue with initialization if it does,
 0311  E9 03B2 R                              JMP     INIT_NO_PORT            ;otherwise, go to error routine.

                                      ;Clear existing error or character.

 0314  83 C2 05                       IC_4A:  ADD     DX,5
 0317  EC                                     IN      AL,DX
 0318  EB 00                                  JMP     SHORT $+2

                                      ;Set baud rate divisor to 1200 baud.

 031A  83 EA 02                               SUB     DX,2                    ;Point to line control register.
 031D  B0 80                                  MOV     AL,80H                  ;Set line control register to divisor programming
 031F  EE                                     OUT     DX,AL                   ;mode.
 0320  EB 00                                  JMP     SHORT $+2               ;Delay.
 0322  83 EA 03                               SUB     DX,3                    ;Point to divisor LSB register (base).
 0325  B0 60                                  MOV     AL,60H                  ;LSB for 1200 bps.
 0327  EE                                     OUT     DX,AL
 0328  EB 00                                  JMP     SHORT $+2               ;Delay.
 032A  42                                     INC     DX                      ;Point to MSB of divisor (base + 1).
 032B  B0 00                                  MOV     AL,00H                  ;MSB for 1200 bps.
 032D  EE                                     OUT     DX,AL
 032E  EB 00                                  JMP     SHORT $+2               ;Delay.

                                      ;Initialize line control register.

 0330  83 C2 02                               ADD     DX,2                    ;Point to line control register (base +3).
 0333  B0 03                                  MOV     AL,03H                  ;8 data bits, 1 stop bit, no parity.
 0335  EE                                     OUT     DX,AL
 0336  EB 00                                  JMP     SHORT $+2               ;Delay.

                                      ;Initialize modem control register.

 0338  42                                     INC     DX                      ;Point to modem control register (base + 4)
 0339  B0 0B                                  MOV     AL,0BH                  ;DTR and RTS set, OUT2 set to enable interrupts.
```

```
033B  EE                        OUT    DX,AL
033C  EB 00                     JMP    SHORT $+2              ;Delay.

                        ;Initialize interrupt enable register.

033E  83 EA 03                  SUB    DX,3                   ;Point to interrupt enable register (base + 1)
0341  B0 01                     MOV    AL,01                  ;Enable Rx Data Ready interrupt
0343  EE                        OUT    DX,AL

                        ;SET UP COM PORT INTERRUPT VECTOR

0344  2E: 8B 1E 0128 R          MOV    BX,COM_NUMBER          ;Get table offset back.
0349  2E: 8B BF 012A R          MOV    DI,INT_TABLE[BX]       ;Use it as index into interrupt vector table.
034E  B8 0000                   MOV    AX,0                   ;Set ES to interrupt vector segment (0:).
0351  8E C0                     MOV    ES,AX
0353  B8 0172 R                 MOV    AX,OFFSET MOUSE_INT    ;Initialize vector.
0356  AB                        STOSW
0357  8C C8                     MOV    AX,CS
0359  AB                        STOSW

                        ;ENABLE MOUSE INTERRUPT ON 8259A INTERRUPT CONTROLLER

035A  2E: 8B 8F 0132 R          MOV    CX,MASK_TABLE[BX]      ;Get mask from table.
035F  E4 21                     IN     AL,21H                 ;Get current mask.
0361  EB 00                     JMP    SHORT IC_10            ;Delay.
0363  22 C1             IC_10:  AND    AL,CL                  ;Clear mask for mouse interrupt.
0365  E6 21                     OUT    21H,AL                 ;Set new value.

0367  FB                        STI                           ;Re-enable interrupts.

0368  B4 0C                     MOV    AH,F_INQUIRE_ENTRY     ;Return CS:IP of PGID driver function.
036A  BD 002A                   MOV    BP,V_SINPUT
036D  1E                        PUSH   DS
036E  CD 6F                     INT    HP_ENTRY
0370  1F                        POP    DS
0371  80 FC 02                  CMP    AH,RS_UNSUPPORTED      ;See if brute force approach is necessary.
0374  75 06                     JNE    INIT_3
0376  0E                        PUSH   CS                     ;Even the best laid plans of mice and men aft
0377  07                        POP    ES                     ;go awry.  F_INQUIRE_PGID is not implemented in
0378  8D 1E 03FF R              LEA    BX, CS:PGID_DRIVER     ;some early ROM versions.  The PGID CS:IP must be
                                                              ;hard coded for these systems.

037C  8B FB             INIT_3: MOV    DI,BX                  ;Move IP into DI.
037E  8C CA                     MOV    DX,CS                  ;Get PGID's DS.
0380  83 C2 02                  ADD    DX, 2                  ;account for ORG 20H
0383  B4 0A                     MOV    AH,F_INS_XCHGFREE      ;Exchange fixed vector address function.
0385  BD 0012                   MOV    BP,V_SYSTEM
0388  1E                        PUSH   DS
0389  CD 6F                     INT    HP_ENTRY
038B  1F                        POP    DS
038C  80 FC F6                  CMP    AH, RS_NO_VECTOR       ;Is it installed in vector table
038F  74 18                     JE     INIT_NO_VECTOR

0391  8B C3                     MOV    AX, BX                 ;Set up for the divide
0393  B3 08                     MOV    BL, 8
0395  F6 F3                     DIV    BL                     ;Convert to a vector index
0397  2E: A2 0151 R             MOV    PGID_VECT_NUM, AL      ;Save for ISR Events

039B  B4 04                     MOV    AH, F_IO_CONTROL       ; Now to make sure that the V_LHPMOUSE
039D  B0 02                     MOV    AL, SF_MOUSE_OVERRIDE  ; driver sets up INT 33H.
039F  BD 00CC                   MOV    BP, V_LHPMOUSE
03A2  1E                        PUSH   DS
03A3  CD 6F                     INT    HP_ENTRY
03A5  1F                        POP    DS

03A6  EB 13 90                  JMP    INIT_OK

03A9                    INIT_NO_VECTOR:

03A9  BA 0103 R                 MOV    DX,OFFSET NO_VECTOR    ;Print error message.
03AC  B4 09                     MOV    AH,PRINT_STR
03AE  CD 21                     INT    DOS_ENTRY
03B0  EB 14                     JMP    SHORT INIT_EXIT

03B2                    INIT_NO_PORT:

03B2  BA 00C8 R                 MOV    DX,OFFSET NO_PORT_MSG  ;Print error message.
03B5  B4 09                     MOV    AH,PRINT_STR
03B7  CD 21                     INT    DOS_ENTRY
03B9  EB 0B                     JMP    SHORT INIT_EXIT

03BB                    INIT_OK:

03BB  8C C8                     MOV    AX,CS                  ;Set DS back to proper value.
03BD  8E D8                     MOV    DS,AX
03BF  BA 00AB R                 MOV    DX,OFFSET OK_MSG       ;Print sign-on message.
03C2  B4 09                     MOV    AH,PRINT_STR           ;MS-DOS print string function number.
03C4  CD 21                     INT    DOS_ENTRY

03C6                    INIT_EXIT:

03C6  06                        PUSH   ES                     ; now to set the number of buttons
03C7  50                        PUSH   AX                     ; V_LHPMOUSE has
```

```
03C8  B8 0000                        MOV     AX, 0
03CB  8E C0                          MOV     ES, AX
03CD  26: 8E 06 01BE                 MOV     ES, ES:[HP_ENTRY * 4 + 2]
03D2  26: 8E 06 00D0                 MOV     ES, ES:[V_[HPMOUSE+4]
03D7  26: C6 06 004C 03              MOV     BYTE PTR ES:MSE_NUM_BUTTON,3  ;Define the number of buttons to 3
03DD  58                             POP     AX
03DE  07                             POP     ES

03DF  2E: C4 3E 0050 R               LES     DI,DWORD PTR REQ_HDR_OFF ;Reload ES:DI with address of request header.
03E4  26: C7 45 0E 04D1 R            MOV     ES:[DI].RH_END_OFF,OFFSET END_OF_DRIVER ;Return end of resident code to
03EA  26: 8C 4D 10                   MOV     ES:[DI].RH_END_SEG,CS              ;MS-DOS.

                              ;RESTORE OLD STACK FRAME AND EXIT

03EE  FA                             CLI                              ;Disable interrupts while working on stack frame
03EF  BE 0124 R                      MOV     SI,OFFSET STACK_PTR      ;Get address of old stack storage.
03F2  8B 24                          MOV     SP,[SI]                  ;Restore stack pointer.
03F4  83 C6 02                       ADD     SI,2
03F7  8B 04                          MOV     AX,[SI]                  ;Get old stack segment.
03F9  8E D0                          MOV     SS,AX                    ;And restore it.
03FB  FB                             STI
                                                                      ;Re-enable interrupts.
03FC  E9 029E R                      JMP     EXIT

03FF                          DEV_INTERRUPT  ENDP
03FF                          DEV_DRIVER     ENDP

                                             page
                                             .list
                              ;***DRIVER HEADER*************************************************
                              ;
                              ;  NAME: PGID_DRIVER
                              ;
                              ;  DESCRIPTION:
                              ;
                              ;  LIST OF FUNCTIONS:  (function code in hex)
                              ;    [Those functions not listed are NOT_SUPPORTED.]
                              ;
                              ;       F_ISR
                              ;       F_SYSTEM
                              ;
                              ;  PARAMETERS:
                              ;       See function headers for specific values for other entry and exit
                              ;       parameters
                              ;
                              ; REGISTERS PRESERVED:
                              ;
                              ;  DEFINITION MODIFICATION HISTORY
                              ;
                              ;    VERSION:
                              ;
                              ;    DESCRIPTION OF CHANGES:
                              ;
                              ;*****************************************************************

                                             subttl PGID Main entry point

                                             page
                                             assume  cs:CODE, ds:nothing
                                             public  PGID_DRIVER
                              ;
                              ; NOTE **** No driver header for PGID ****
                              ;  Only 2 functions are supported: F_ISR, F_SYSTEM -- all others are unsupported

03FF                          pgid_driver    proc    near
03FF  80 FC 00                               cmp     ah,F_ISR              ; F_ISR?
0402  75 04                                  jne     check_f_system
0404  E8 0414 R                              call    pgid_Isr
0407  CF                                     iret

0408                          check_f_system:
0408  80 FC 02                               cmp     ah,F_SYSTEM           ; F_SYSTEM?
040B  75 04                                  jne     pgid_opcode_bad
040D  E8 0496 R                              call    pgid_system
0410  CF                                     iret                          ; function has set return code

                              ;
                              ;-----------------------------------------------------
                              ; Main opcode out of range of PGID functions supported
                              ;   just return RS_UNSUPPORTED
                              ;-----------------------------------------------------
0411                          pgid_opcode_bad:
0411  B4 02                                  mov     ah,RS_UNSUPPORTED
0413  CF                                     iret
                              ;
0414                          pgid_driver    endp

                                             page
                              ;***FUNCTION HEADER***********************************************
                              ;
                              ;  NAME:  PGID_ISR
                              ;
                              ;  FUNCTIONAL DESCRIPTION:
```

```
                      ;
                      ;      A graphics input device (GID) physical event has occurred which
                      ;  caused an F_ISR request.  If the event was a button press, then
                      ;  the D_STATE and D_TRANSITION fields will be adjusted and the parent
                      ;  driver will be called immediately.
                      ;      If a the event was a movement, this function will update the
                      ;  absolute position field if the device is a relative device or will
                      ;  update the relative position field if it's an absolute device.  It
                      ;  will then call the PARENT driver to handle the movement event.
                      ;**************************************
                      ;  NOTE: The PGID driver takes HP-HIL 'Y' axis data and translates
                      ;        it into INDUSTRY-STANDARD space data (flips the Y axis).
                      ;        HP-HIL has positive 'Y' in the upward direction, while
                      ;        INDUSTRY-STD. is downward.
                      ;**************************************
                      ;
                      ;  PARAMETERS
                      ;
                      ;    ON ENTRY:
                      ;       AH = F_ISR
                      ;       DH = D_TYPE
                      ;       DL = SOURCE  Vector Index
                      ;       DS:0  = Pointer to Physical device header and describe record
                      ;  For Button Event (Keycode Event Record) : (D_TYPE = T_KC_BUTTON)
                      ;       BX =  Button transition information
                      ;               bits 0..6:  buttons
                      ;               bits 7:   0  up transition
                      ;                         1  down transition
                      ; For Movement Event (GID Event Record, D_TYPE = T_REL08, T_REL16,
                      ;                                          T_ABS08, or T_ABS16):
                      ;       BX =  AXIS-0 (X) Movement in RAW data form (SIGN EXTENDED, if necessary)
                      ;       CX =  AXIS-1 (Y) Movement in RAW data form (SIGN EXTENDED, if necessary)
                      ;
                      ;    ON EXIT:
                      ;       AH = Return Code (SET BY PARENT Driver)
                      ;
                      ;  REGISTERS ALTERED: ax,bx,cx
                      ;
                      ;  DEFINITION MODIFICATION HISTORY
                      ;
                      ;    VERSION:
                      ;
                      ;    DESCRIPTION OF CHANGES:
                      ;
                      ;=====================================================================
                                      page
0414                  pgid_isr        proc    near
                      ;-------------------------------
                      ; See if this was a button event
                      ;-------------------------------
0414  80 FE 09                        cmp     dh,T_KC_BUTTON          ; D_TYPE = T_KC_BUTTON ?
0417  74 57                           je      short button_isr        ; adjust D_STATE & D_TRANSITION
                      ;
                      ;-----------------------------------------------
                      ; A movement occurred.  If this was an absolute device
                      ;   that moved, then adjust the relative location field in the describe record.
                      ;   If it was a relative device, then adjust the absolute location field
                      ;   in the describe record.
                      ;  BX,CX have X,Y movement respectively.
                      ;-----------------------------------------------
0419                  movement_isr:
0419  80 FE 40                        cmp     dh,T_REL08              ; relative 8 bit movement
041C  74 3E                           je      short rel_move
041E  80 FE 41                        cmp     dh,T_REL16              ; relative 16 bit movement
0421  74 39                           je      short rel_move
0423  80 FE 42                        cmp     dh,T_ABS08              ; absolute 8 bit movement
0426  74 08                           je      short abs_move
0428  80 FE 43                        cmp     dh,T_ABS16              ; absolute 16 bit movement
042B  74 03                           je      short abs_move
                      ;
                      ;--------------------------------------
                      ; If none of the above devices, then this is a bad input device
                      ;--------------------------------------
042D  B4 FE                           mov     ah,RS_FAIL              ; return RS_FAIL
042F  C3                              ret                             ; return to main driver
                                      page
                      ;*************************
                      ; Absolute movement
                      ;-------------------------
                      ; We must invert the Y axis to put into INDUSTRY STANDARD coordinate space.
                      ; Must convert 'Y' coordinate such that negative movement is upward (opposite
                      ;       of HP-HIL definition.)
                      ; -- Set BX,CX (x,y ABSOLUTE movement) for event record when done, then pass
                      ;                event record to parent driver.
                      ;
                      ; (BX) is 'X' HP-HIL coordinate.
                      ; (CX) is 'Y'    [ ABS_Y(std) = D_SIZE_Y - ABS_Y(hphil) ]
                      ;*************************
0430                  abs_move:
0430  87 1E 0014                      xchg    bx,ds:D_ABS_X   ; save new x position
0434  2B 1E 0014                      sub     bx,ds:D_ABS_X   ; (OLD - NEW)
0438  F7 DB                           neg     bx              ; Relative move = (NEW - OLD)
043A  89 1E 0018                      mov     ds:D_REL_X,bx   ; save new x relative
```

```

043E  8B 1E 0012                          mov     bx,ds:D_SIZE_Y   ; 'Y' limit
0442  2B D9                               sub     bx,cx            ; invert the axis: bx = (LIMIT - y)
0444  87 1E 0016                          xchg    bx,ds:D_ABS_Y    ; New ABS Y
0448  2B 1E 0016                          sub     bx,ds:D_ABS_Y    ; (OLD - NEW)
044C  F7 DB                               neg     bx               ; Relative move = (NEW - OLD)
044E  89 0E 0018                          mov     ds:D_REL_X,cx    ; save new Y relative

                          ;---------------------------------------------
                          ; GET the X,Y absolute coordinates for the event record
                          ;---------------------------------------------
0452  8B 1E 0014                          mov     bx,ds:D_ABS_X
0456  8B 0E 0016                          mov     cx,ds:D_ABS_Y
045A  EB 31                               jmp     short give_to_parent     ; ok to pass event to parent
                                          page
                          ;*************************
                          ; Relative movement
                          ;-------------------------
                          ; We must invert the Y axis to put into INDUSTRY STANDARD coordinate space.
                          ; Must convert 'Y' coordinate such that negative movement is upward (opposite
                          ;       of HP-HIL definition.)
                          ; -- Set BX,CX (x,y RELATIVE movement) for event record when done, then pass
                          ;               event record to parent driver.
                          ;
                          ; (BX) is 'X' HP-HIL coordinate.
                          ; (CX) is 'Y'  [ REL_Y(std) = -REL_Y(hphil) ]
                          ;*************************
045C                      rel_move:
045C, 89 1E 0018                          mov     ds:D_REL_X,bx    ; save new rel. move (X)
0460  F7 D9                               neg     cx               ; CONVERT TO INDUSTRY STD. SPACE
0462  89 0E 001A                          mov     ds:D_REL_Y,cx    ; save new rel. move (Y)

0466  01 1E 0014                          add     ds:D_ABS_X,bx    ; add new X relative movement
046A  01 0E 0016                          add     ds:D_ABS_Y,cx    ; add new Y relative movement

                          ;---------------------------------------------
                          ;  BX,CX still contain X,Y relative movement information for the event record
                          ;---------------------------------------------
046E  EB 1D                               jmp     short give_to_parent     ; ok to pass event to parent
                                          page
                          ;-------------------------
                          ; Button Press/Release ISR
                          ;   Adjust the D_TRANSITION and D_STATE fields of the physical device's
                          ;   describe record
                          ;
                          ;    Assuming:  1. Only one button can make a transition at a time.
                          ;               2  The button only either goes up or down, not both.
                          ;               3. No strings of buttons are sent (CX register available).
                          ;
                          ; BL is number of button that changed
                          ;       bit 7 is the up/down (1/0) flag
                          ;-------------------------
= 0080                    UP_DOWN_BIT     equu    10000000B        ; bit 7 is up (1), down(0) bit

0470                      button_isr:
                          ;---------------------------------------
                          ; Convert button number to bit mask corresponding
                          ;   to the changed button
                          ;---------------------------------------
0470  8A CB                               mov     cl,bl            ; get button # keycode in CL for shift
0472  80 E1 7F                            and     cl,01111111B     ; keep button #, get rid of up/down flag
0475  B0 01                               mov     al,00000001B     ; put '1' in bit 0 of al
0477  D2 E0                               shl     al,cl            ; set appropriate button bit mask

0479  A2 000C                             mov     ds:D_TRANSITION,al       ; note which button changed

047C  F6 C3 80                            test    bl,UP_DOWN_BIT   ; [bit 7] Was it UP = 1 or down = 0
047F  74 06                               jz      short button_down
0481                      button_up:
0481  08 06 000D                          or      ds:D_STATE,al    ; set the button = 1 (up)
0485  EB 06                               jmp     short give_to_parent     ; ok to pass event to parent
0487                      button_down:
0487  F6 D0                               not     al               ; invert for clearing the bit
0489  20 06 000D                          and     ds:D_STATE,al    ; clear the button to 0 (down)
                          ;
                          ;  fall through to GIVE_TO_PARENT code -- ok to pass event to parent now
                          ;               jmp     give_to_parent   ; (COMMENTED OUT -- jump not necessary)
                                          page
                          ; ok to pass event to parent now
                          ;-----------------------------------
                          ; Call PARENT driver to handle the ISR
                          ; NOTE: HPHIL driver has already adjusted D_SOURCE field, HPHIL_ID and other
                          ;     relevant HPHIL info before passing the event up to here.
                          ;-----------------------------------
048D                      give_to_parent:
048D  B4 00                               mov     ah,F_ISR                 ; tell parent: ISR function
048F  8B 2E 000A                          mov     bp,ds:DH_V_PARENT        ; parent's vector
0493  CD 6F                               INT     HP_ENTRY
0495  C3                                  ret                              ; return to main driver
0496                      pgid_isr        endp

                                          subttl  PGID_SYSTEM function
```

```
                                            page
                            ;***FUNCTION HEADER*********************************************
                            ;
                            ;  NAME:  PGID_SYSTEM
                            ;
                            ;  FUNCTIONAL DESCRIPTION:
                            ;
                            ;       This function supports the HP SYSTEM subfunctions requested of
                            ;  the PGID driver.  The subfunction is checked to make sure that it
                            ;  is in the appropriate range.
                            ;
                            ;
                            ;  PARAMETERS
                            ;
                            ;    ON ENTRY:
                            ;       AH = F_SYSTEM
                            ;       AL = SYSTEM subfunction code
                            ;
                            ;         F_SYSTEM Subfunctions (in hex):
                            ;               (functions not included are UNSUPPORTED)
                            ;         SF_INIT
                            ;         SF_START
                            ;         SF_REPORT_STATE
                            ;         SF_VERSION_DESC
                            ;
                            ;    ON EXIT:
                            ;       See individual system subfunctions for values returned.
                            ;       RS_UNSUPPORTED will be returned if the subfunction is out of range.
                            ;
                            ;  REGISTERS PRESERVED:
                            ;
                            ;  DEFINITION MODIFICATION HISTORY
                            ;
                            ;    VERSION:
                            ;
                            ;    DESCRIPTION OF CHANGES:
                            ;
                            ;****************************************************************
                                            page
0496                        pgid_system     proc    near

0496  3C 06 90 90                           cmp     al,MAX_PGID_SYS_FN      ; check bounds
049A  77 0D                                 ja      short pgid_sys_bad      ; out of range ?

049C  87 EB                                 xchg    bp,bx           ; save bx, set bp=subfunction code (al)
049E  8A D8                                 mov     bl,al
04A0  32 FF                                 xor     bh,bh
04A2  87 EB                                 xchg    bp,bx

04A4  2E: FF A6 04AC R                      jmp     cs:word ptr pgid_sys_case[bp]
04A9                        pgid_sys_bad:
04A9  B4 02                                 mov     ah,RS_UNSUPPORTED       ; bad subfunction code
04AB  C3                                    ret                             ; return to main driver

                            ;-----------------------------------
                            ; PGID_SYSTEM subfunction jump table
                            ;-----------------------------------
04AC                        pgid_sys_case:
04AC  04B4 R                                dw      word ptr  pgid_init     ; SF_INIT
04AE  04BC R                                dw      word ptr  pgid_start    ; SF_START
04B0  04BF R                                dw      word ptr  pgid_state    ; SF_REPORT_STATE
04B2  04C2 R                                dw      word ptr  pgid_version  ; SF_VERSION_DESC
= 0006                      MAX_PGID_SYS_FN equ     byte ptr ($ - pgid_sys_case - 2) ; max supported sys fn.
04B4                        pgid_system     endp

                                            subttl  PGID_INIT System Subfunction
                                            page
                            ;***FUNCTION HEADER*********************************************
                            ;
                            ;  NAME:        PGID_INIT
                            ;
                            ;  FUNCTIONAL DESCRIPTION:
                            ;
                            ;       System subfunction SF_INIT -- initialize the physical device
                            ;  header and describe record.  IT IS ASSUMED THAT THE HPHIL DRIVER HAS
                            ;  INITIALIZED ALL APPROPRIATE INFO ALREADY... All position and button
                            ;  data is zeroed out, and relevant HPHIL info is already filled in.
                            ;  Only must set default button states (all off (=1)).
                            ;
                            ;  PARAMETERS
                            ;
                            ;    ON ENTRY:
                            ;       AH = F_SYSTEM
                            ;       AL = SF_INIT
                            ;
                            ;    ON EXIT:
                            ;       AH = Return status (RS_SUCCESSFUL)
                            ;
                            ;  REGISTERS ALTERED: ax
                            ;
                            ;  DEFINITION MODIFICATION HISTORY
                            ;
                            ;    VERSION:
```

RS-232 Mouse Driver

```
                                ;
                                ;       DESCRIPTION OF CHANGES:
                                ;
                                ;=====================================================================
04B4                            pgid_init       proc    near
= 00FF                          INIT_BUTTON_STATE       equu    0FFh                    ; all buttons open
                                ;
04B4  C6 06 000D FF                             mov     ds:D_STATE,INIT_BUTTON_STATE    ; all off

04B9  B4 00                                     mov     ah,RS_SUCCESSFUL; successful initialization
04BB  C3                                        ret                     ; return to main driver
04BC                            pgid_init       endp
                                                subttl  PGID_START System Subfunction
                                                page
                                ;===FUNCTION HEADER===================================================
                                ;
                                ;    NAME:          PGID_START
                                ;
                                ;    FUNCTIONAL DESCRIPTION:
                                ;
                                ;         System subfunction SF_START -- start the driver.  This does nothing
                                ;    but return with RS_SUCCESSFUL.
                                ;
                                ;    PARAMETERS
                                ;
                                ;      ON ENTRY:
                                ;         AH = F_SYSTEM
                                ;         AL = SF_START
                                ;
                                ;      ON EXIT:
                                ;         AH = return status (RS_SUCCESSFUL)
                                ;
                                ;    REGISTERS ALTERED: ah
                                ;
                                ;    DEFINITION MODIFICATION HISTORY
                                ;
                                ;      VERSION:
                                ;
                                ;      DESCRIPTION OF CHANGES:
                                ;
                                ;=====================================================================
04BC                            pgid_start      proc    near
                                ;
04BC  B4 00                                     mov     ah,RS_SUCCESSFUL; successful start up
04BE  C3                                        ret                     ; return to main driver
04BF                            pgid_start      endp
                                                subttl  PGID_STATE System Subfunction
                                                page
                                ;===FUNCTION HEADER===================================================
                                ;
                                ;    NAME:          PGID_STATE
                                ;
                                ;    FUNCTIONAL DESCRIPTION:
                                ;
                                ;         System subfunction PGID_REPORT_STATE -- report the state of this
                                ;    driver.  (NOT SUPPORTED)
                                ;
                                ;    PARAMETERS
                                ;
                                ;      ON ENTRY:
                                ;         AH = F_SYSTEM
                                ;         AL = SF_REPORT_STATE
                                ;
                                ;      ON EXIT:
                                ;         AH = return status (RS_UNSUPPORTED)
                                ;
                                ;    REGISTERS ALTERED: ah,dx
                                ;
                                ;    DEFINITION MODIFICATION HISTORY
                                ;
                                ;      VERSION:
                                ;
                                ;      DESCRIPTION OF CHANGES:
                                ;
                                ;=====================================================================
04BF                            pgid_state      proc    near
04BF  B4 02                                     mov     ah,RS_UNSUPPORTED       ; function not supported
04C1  C3                                        ret                     ; return to main driver
04C2                            pgid_state      endp
                                                subttl  PGID_VERSION System Subfunction
                                                page
                                ;===FUNCTION HEADER===================================================
                                ;
                                ;    NAME:          PGID_VERSION
                                ;
                                ;    FUNCTIONAL DESCRIPTION:
                                ;         System subfunction SF_VERSION_DESC -- Report the version
                                ;    number of the driver. (Use standard system version number)
                                ;
                                ;    PARAMETERS
                                ;
                                ;      ON ENTRY:
                                ;         AH = F_SYSTEM
                                ;         AL = SF_VERSION_DESC
```

```
                                         ;
                                         ;    ON EXIT:
                                         ;       AH = RS_SUCCESSFUL
                                         ;   (others): see hp_system_version function.
                                         ;
                                         ;  REGISTERS ALTERED: ah,es,di
                                         ;
                                         ;  DEFINITION MODIFICATION HISTORY
                                         ;
                                         ;    VERSION:
                                         ;
                                         ;    DESCRIPTION OF CHANGES:
                                         ;
                                         ;=====================================================================
       04C2                              pgid_version    proc    near
       04C2   B4 00                                      mov     ah, RS_SUCCESSFUL
       04C4   BB 5225                                    mov     bx, 5225H
       04C7   B9 0010                                    mov     cx, VERSION_LEN
       04CA   0E                                         push    cs
       04CB   07                                         pop     es
       04CC   8D 3E 0099 R                               lea     di, cs:VERSION_LAB
       04D0   C3                                         ret                     ; return to PGID main driver
       04D1                              pgid_version    endp


                                         ;*************************************************************************
       04D1                              END_OF_DRIVER:

                                         ;LOCAL STACK USED DURING INITIALIZATION

       04D1      40 [                            DB 64 DUP (0)
                       00
                            ]


       0511                              STACK_TOP:
       0511                              CODE        ENDS

                                                     END
```

```
Structures and records

                    N a m e                      Width   # fields
                                                 Shift   Width    Mask     Initial

ATTR . . . . . . . . . . . . . . . . . . . . . . 0010    000B
  DEV  . . . . . . . . . . . . . . . . . . . . . 000F    0001     8000     0000
  IOCTL. . . . . . . . . . . . . . . . . . . . . 000E    0001     4000     0000
  IBM. . . . . . . . . . . . . . . . . . . . . . 000D    0001     2000     0000
  X. . . . . . . . . . . . . . . . . . . . . . . 000C    0001     1000     0000
  OCREM. . . . . . . . . . . . . . . . . . . . . 000B    0001     0800     0000
  Y. . . . . . . . . . . . . . . . . . . . . . . 0005    0006     07E0     0000
  SPEC . . . . . . . . . . . . . . . . . . . . . 0004    0001     0010     0000
  CLK. . . . . . . . . . . . . . . . . . . . . . 0003    0001     0008     0000
  NUL. . . . . . . . . . . . . . . . . . . . . . 0002    0001     0004     0000
  STDO . . . . . . . . . . . . . . . . . . . . . 0001    0001     0002     0000
  STDI . . . . . . . . . . . . . . . . . . . . . 0000    0001     0001     0000
DESCRIBE . . . . . . . . . . . . . . . . . . . . 0020    0017
  D_SOURCE . . . . . . . . . . . . . . . . . . . 0000
  D_HPHIL_ID . . . . . . . . . . . . . . . . . . 0001
  D_DESC_MASK. . . . . . . . . . . . . . . . . . 0002
  D_IO_MASK. . . . . . . . . . . . . . . . . . . 0003
  D_XDESC_MASK . . . . . . . . . . . . . . . . . 0004
  D_MAX_AXIS . . . . . . . . . . . . . . . . . . 0005
  D_CLASS. . . . . . . . . . . . . . . . . . . . 0006
  D_PROMPTS. . . . . . . . . . . . . . . . . . . 0007
  D_RESERVED . . . . . . . . . . . . . . . . . . 0008
  D_BURST_LEN. . . . . . . . . . . . . . . . . . 0009
  D_WR_REG . . . . . . . . . . . . . . . . . . . 000A
  D_RD_REG . . . . . . . . . . . . . . . . . . . 000B
  D_TRANSITION . . . . . . . . . . . . . . . . . 000C
  D_STATE. . . . . . . . . . . . . . . . . . . . 000D
  D_RESOLUTION . . . . . . . . . . . . . . . . . 000E
  D_SIZE_X . . . . . . . . . . . . . . . . . . . 0010
  D_SIZE_Y . . . . . . . . . . . . . . . . . . . 0012
  D_ABS_X. . . . . . . . . . . . . . . . . . . . 0014
  D_ABS_Y. . . . . . . . . . . . . . . . . . . . 0016
  D_REL_X. . . . . . . . . . . . . . . . . . . . 0018
  D_REL_Y. . . . . . . . . . . . . . . . . . . . 001A
  D_ACCUM_X. . . . . . . . . . . . . . . . . . . 001C
  D_ACCUM_Y. . . . . . . . . . . . . . . . . . . 001E
HP_ATTR. . . . . . . . . . . . . . . . . . . . . 0010    000D
  HP . . . . . . . . . . . . . . . . . . . . . . 000F    0001     8000     0000
  DEVCFG . . . . . . . . . . . . . . . . . . . . 000E    0001     4000     0000
  ISR. . . . . . . . . . . . . . . . . . . . . . 000D    0001     2000     0000
  ENTRY. . . . . . . . . . . . . . . . . . . . . 000C    0001     1000     0000
  TYPE . . . . . . . . . . . . . . . . . . . . . 0009    0003     0E00     0000
  STR. . . . . . . . . . . . . . . . . . . . . . 0008    0001     0100     0000
  MAP_CALL . . . . . . . . . . . . . . . . . . . 0007    0001     0080     0000
  A. . . . . . . . . . . . . . . . . . . . . . . 0006    0001     0040     0000
  SUBADD . . . . . . . . . . . . . . . . . . . . 0004    0002     0030     0000
  PSHARE . . . . . . . . . . . . . . . .    0003    0001    0008    0000
  CSHARE . . . . . . . . . . . . . . . .    0002    0001    0004    0000
  ROM  . . . . . . . . . . . . . . . . .    0001    0001    0002    0000
  B  . . . . . . . . . . . . . . . . . .    0000    0001    0001    0000
HP_HEADER  . . . . . . . . . . . . . . .    0010    0009
  DH_ATR . . . . . . . . . . . . . . . .    0000
  DH_NAME_INDEX  . . . . . . . . . . . .    0002
  DH_V_DEFAULT . . . . . . . . . . . . .    0004
  DH_P_CLASS . . . . . . . . . . . . . .    0006
  DH_C_CLASS . . . . . . . . . . . . . .    0008
  DH_V_PARENT  . . . . . . . . . . . . .    000A
  DH_V_CHILD . . . . . . . . . . . . . .    000C
  DH_MAJOR . . . . . . . . . . . . . . .    000E
  DH_MINOR . . . . . . . . . . . . . . .    000F
REQ_HEADER . . . . . . . . . . . . . . .    0017    000A
  RH_LENGTH  . . . . . . . . . . . . . .    0000
  RH_UNIT_CODE . . . . . . . . . . . . .    0001
  RH_CMD_CODE  . . . . . . . . . . . . .    0002
  RH_STATUS  . . . . . . . . . . . . . .    0003
  RH_RESERVED  . . . . . . . . . . . . .    0005
  RH_UNIT_CNT  . . . . . . . . . . . . .    000D
  RH_END_OFF . . . . . . . . . . . . . .    000E
  RH_END_SEG . . . . . . . . . . . . . .    0010
  RH_BPB . . . . . . . . . . . . . . . .    0012
  RH_DRIV  . . . . . . . . . . . . . . .    0016
STATUS . . . . . . . . . . . . . . . . .    0010    0005
  ERROR  . . . . . . . . . . . . . . . .    000F    0001    8000    0000
  Z  . . . . . . . . . . . . . . . . . .    000A    0005    7C00    0000
  BUSY . . . . . . . . . . . . . . . . .    0009    0001    0200    0000
  DONE . . . . . . . . . . . . . . . . .    0008    0001    0100    0000
  ERR_TYPE . . . . . . . . . . . . . . .    0000    0008    00FF    0000


Segments and Groups:

                N a m e                     Size    Align   Combine Class

CODE . . . . . . . . . . . . . . . . . .    0511    PARA    PUBLIC  'CODE'

Symbols:

                N a m e                     Type    Value   Attr

ABS_MOVE . . . . . . . . . . . . . . . .    L NEAR  0430    CODE
BAD_CMD  . . . . . . . . . . . . . . . .    L NEAR  0292    CODE
BISR2  . . . . . . . . . . . . . . . . .    L NEAR  01FA    CODE
BUTTON_DOWN  . . . . . . . . . . . . . .    L NEAR  0487    CODE
BUTTON_ISR . . . . . . . . . . . . . . .    L NEAR  0470    CODE
BUTTON_TAB . . . . . . . . . . . . . . .    L BYTE  01F7    CODE
BUTTON_UP  . . . . . . . . . . . . . . .    L NEAR  0481    CODE
CHECK_F_SYSTEM . . . . . . . . . . . . .    L NEAR  0408    CODE
CMD_TABLE  . . . . . . . . . . . . . . .    L WORD  0152    CODE
COM_MSG  . . . . . . . . . . . . . . . .    L BYTE  00C1    CODE
COM_NUMBER . . . . . . . . . . . . . . .    L WORD  0128    CODE
CR . . . . . . . . . . . . . . . . . . .    Number  000D
DEBUG  . . . . . . . . . . . . . . . . .    Alias   TRUE
DEV_ATTR . . . . . . . . . . . . . . . .    Number  AC18
DEV_DESCRIBE . . . . . . . . . . . . . .    L 0020  0030    CODE
DEV_DRIVER . . . . . . . . . . . . . . .    F PROC  0000    CODE    Length =03FF
DEV_HEADER . . . . . . . . . . . . . . .    L 0010  0020    CODE
DEV_INTERRUPT  . . . . . . . . . . . . .    F PROC  0270    CODE    Length =018F

DEV_STRATEGY . . . . . . . . . . . . . .    F PROC  0265    CODE    Length =000B
DOS_ENTRY  . . . . . . . . . . . . . . .    Number  0021
DRIVER_ATTR  . . . . . . . . . . . . . .    L WORD  0004    CODE
DRIVER_NAME  . . . . . . . . . . . . . .    L BYTE  000A    CODE
END_OF_DRIVER  . . . . . . . . . . . . .    L NEAR  04D1    CODE
EXIT . . . . . . . . . . . . . . . . . .    L NEAR  029E    CODE
FALSE  . . . . . . . . . . . . . . . . .    Number  0000
FRAME_COUNT  . . . . . . . . . . . . . .    L WORD  013A    CODE
F_INQUIRE_ENTRY  . . . . . . . . . . . .    Number  000C
F_INS_BASEHPVT . . . . . . . . . . . . .    Number  0004
F_INS_XCHGFREE . . . . . . . . . . . . .    Number  000A
F_IO_CONTROL . . . . . . . . . . . . . .    Number  0004
F_ISR  . . . . . . . . . . . . . . . . .    Number  0000
F_SYSTEM . . . . . . . . . . . . . . . .    Number  0002
GIVE_TO_PARENT . . . . . . . . . . . . .    L NEAR  048D    CODE
HPHIL_ADD  . . . . . . . . . . . . . . .    L BYTE  0150    CODE
HPHIL_TABLE  . . . . . . . . . . . . . .    L BYTE  0142    CODE    Length =000E
HP_ENTRY . . . . . . . . . . . . . . . .    Number  006F
IC_1 . . . . . . . . . . . . . . . . . .    L NEAR  02C8    CODE
IC_10  . . . . . . . . . . . . . . . . .    L NEAR  0363    CODE
IC_2 . . . . . . . . . . . . . . . . . .    L NEAR  02DA    CODE
IC_3 . . . . . . . . . . . . . . . . . .    L NEAR  02EF    CODE
IC_4 . . . . . . . . . . . . . . . . . .    L NEAR  02F6    CODE
IC_4A  . . . . . . . . . . . . . . . . .    L NEAR  0314    CODE
INIT_3 . . . . . . . . . . . . . . . . .    L NEAR  037C    CODE
INIT_BUTTON_STATE  . . . . . . . . . . .    Number  00FF
INIT_CODE  . . . . . . . . . . . . . . .    L NEAR  02A7    CODE
INIT_EXIT  . . . . . . . . . . . . . . .    L NEAR  03C6    CODE
INIT_NO_PORT . . . . . . . . . . . . . .    L NEAR  03B2    CODE
INIT_NO_VECTOR . . . . . . . . . . . . .    L NEAR  03A9    CODE
```

RS-232 Mouse Driver

```
INIT_OK . . . . . . . . . . . . . . .   L NEAR   03BB    CODE
INT_ENT . . . . . . . . . . . . . . .   L WORD   0008    CODE
INT_TABLE . . . . . . . . . . . . . .   L WORD   012A    CODE
LAST_SYNCH  . . . . . . . . . . . . .   L BYTE   0141    CODE
LF  . . . . . . . . . . . . . . . . .   Number   000A
MASK_TABLE  . . . . . . . . . . . . .   L WORD   0132    CODE
MAX_PGID_SYS_FN . . . . . . . . . . .   E BYTE   0006
MBUTTON . . . . . . . . . . . . . . .   L NEAR   01D8    CODE
MBUTTON_DOWN  . . . . . . . . . . . .   L NEAR   01E6    CODE
MBUTTON_ISR . . . . . . . . . . . . .   L NEAR   01E8    CODE
MBUTTON_UP  . . . . . . . . . . . . .   L NEAR   01E2    CODE
MNEXT_BUTTON  . . . . . . . . . . . .   L NEAR   021F    CODE
MOUSE_INT . . . . . . . . . . . . . .   L NEAR   0172    CODE
MOVEMENT_ISR  . . . . . . . . . . . .   L NEAR   0419    CODE
MSD_BAD_LENGTH  . . . . . . . . . . .   Number   0005
MSD_BLD_BPB . . . . . . . . . . . . .   Number   0002
MSD_CRC_ERROR . . . . . . . . . . . .   Number   0004
MSD_DEV_CLOSE . . . . . . . . . . . .   Number   000E
MSD_DEV_OPEN  . . . . . . . . . . . .   Number   000D
MSD_GEN_FAILURE . . . . . . . . . . .   Number   000C
MSD_INIT  . . . . . . . . . . . . . .   Number   0000
MSD_INPUT . . . . . . . . . . . . . .   Number   0004
MSD_IN_FLUSH  . . . . . . . . . . . .   Number   0007
MSD_IN_NOWAIT . . . . . . . . . . . .   Number   0005
MSD_IN_STATUS . . . . . . . . . . . .   Number   0006
MSD_IOCTL_IN  . . . . . . . . . . . .   Number   0003
MSD_IOCTL_OUT . . . . . . . . . . . .   Number   000C
MSD_MEDIA_CHK . . . . . . . . . . . .   Number   0001
MSD_NOT_RDY . . . . . . . . . . . . .   Number   0002
MSD_OUTPUT  . . . . . . . . . . . . .   Number   0008
MSD_OUT_FLUSH . . . . . . . . . . . .   Number   000B
MSD_OUT_STATUS  . . . . . . . . . . .   Number   000A
MSD_OUT_VERIFY  . . . . . . . . . . .   Number   0009
MSD_PAPER_OUT . . . . . . . . . . . .   Number   0009
MSD_READ_FAULT  . . . . . . . . . . .   Number   000B
MSD_REM_MEDIA . . . . . . . . . . . .   Number   000F
MSD_SEC_NOT_FOUND . . . . . . . . . .   Number   0008
MSD_SEEK_ERROR  . . . . . . . . . . .   Number   0006
MSD_UNKNOWN_CMD . . . . . . . . . . .   Number   0003
MSD_UNKNOWN_MEDIA . . . . . . . . . .   Number   0007
MSD_UNKNOWN_UNIT  . . . . . . . . . .   Number   0001
MSD_WRITE_FAULT . . . . . . . . . . .   Number   000A
MSD_WRITE_PROT  . . . . . . . . . . .   Number   0000
MSE_NUM_BUTTON  . . . . . . . . . . .   Number   004C
MSI_1 . . . . . . . . . . . . . . . .   L NEAR   01A2    CODE
MSI_2 . . . . . . . . . . . . . . . .   L NEAR   01B5    CODE
MSI_3 . . . . . . . . . . . . . . . .   L NEAR   0225    CODE
MSI_4 . . . . . . . . . . . . . . . .   L NEAR   023D    CODE
MSI_5 . . . . . . . . . . . . . . . .   L NEAR   0260    CODE
NO_PORT_MSG . . . . . . . . . . . . .   L BYTE   00C8    CODE
NO_VECTOR . . . . . . . . . . . . . .   L BYTE   0103    CODE
OK_MSG  . . . . . . . . . . . . . . .   L BYTE   00AB    CODE
PGID_DRIVER . . . . . . . . . . . . .   N PROC   03FF    CODE     Global  Length =0015
PGID_INIT . . . . . . . . . . . . . .   N PROC   04B4    CODE     Length =0008
PGID_ISR  . . . . . . . . . . . . . .   N PROC   0414    CODE     Length =0082
PGID_OPCODE_BAD . . . . . . . . . . .   L NEAR   0411    CODE
PGID_START  . . . . . . . . . . . . .   N PROC   04BC    CODE     Length =0003
PGID_STATE  . . . . . . . . . . . . .   N PROC   04BF    CODE     Length =0003
PGID_SYSTEM . . . . . . . . . . . . .   N PROC   0496    CODE     Length =001E
PGID_SYS_BAD  . . . . . . . . . . . .   L NEAR   04A9    CODE
PGID_SYS_CASE . . . . . . . . . . . .   L NEAR   04AC    CODE
PGID_VECT_NUM . . . . . . . . . . . .   L BYTE   0151    CODE
PGID_VERSION  . . . . . . . . . . . .   N PROC   04C2    CODE     Length =000F
PRINT_STR . . . . . . . . . . . . . .   Number   0009
REL_MOVE  . . . . . . . . . . . . . .   L NEAR   045C    CODE
REQ_HDR_OFF . . . . . . . . . . . . .   L WORD   0050    CODE
REQ_HDR_SEG . . . . . . . . . . . . .   L WORD   0052    CODE
RH_CMD_LINE . . . . . . . . . . . . .   E DWORD  0012
RS_DONE . . . . . . . . . . . . . . .   Number   0006
RS_FAIL . . . . . . . . . . . . . . .   Number   00FE
RS_NO_VECTOR  . . . . . . . . . . . .   Number   00F8
RS_SUCCESSFUL . . . . . . . . . . . .   Number   0000
RS_UNSUPPORTED  . . . . . . . . . . .   Number   0002
SF_MOUSE_OVERRIDE . . . . . . . . . .   Number   0002
SIGN_ON_MSG . . . . . . . . . . . . .   L BYTE   0054    CODE
STACK_PTR . . . . . . . . . . . . . .   L WORD   0124    CODE
STACK_SEG . . . . . . . . . . . . . .   L WORD   0126    CODE
STACK_TOP . . . . . . . . . . . . . .   L NEAR   0511    CODE
STRAT_ENT . . . . . . . . . . . . . .   L WORD   0006    CODE
TEMP_BUFFER . . . . . . . . . . . . .   L BYTE   013C    CODE     Length =0005
TRUE  . . . . . . . . . . . . . . . .   Number   - 0001
T_ABS08 . . . . . . . . . . . . . . .   Number   0042
T_ABS16 . . . . . . . . . . . . . . .   Number   0043
T_KC_BUTTON . . . . . . . . . . . . .   Number   0009
T_REL08 . . . . . . . . . . . . . . .   Number   0040
T_REL16 . . . . . . . . . . . . . . .   Number   0041
UNSUPPORT_CMD . . . . . . . . . . . .   L NEAR   0292    CODE
UP_DOWN_BIT . . . . . . . . . . . . .   Number   0080
VERSION_LAB . . . . . . . . . . . . .   L BYTE   0099    CODE
VERSION_LEN . . . . . . . . . . . . .   Number   0010
V_DOLITTLE  . . . . . . . . . . . . .   Number   0006
V_LHPMOUSE  . . . . . . . . . . . . .   Number   00CC
```

RS-232 Mouse Driver

```
V_SINPUT . . . . . . . . . . . . .           Number  002A
V_SYSTEM . . . . . . . . . . . . .           Number  0012

40380 Bytes free

Warning Severe
Errors  Errors
0       0


A. . . . . . . . . . . . . . . . .   203#
ABS_MOVE . . . . . . . . . . . . .   999    1001    1021#
ATTR . . . . . . . . . . . . . . .    76#    263

B. . . . . . . . . . . . . . . . .   203#
BAD_CMD. . . . . . . . . . . . . .   613     615     623#
BISR2. . . . . . . . . . . . . . .   498     502#
BUSY . . . . . . . . . . . . . . .    92#
BUTTON_DOWN. . . . . . . . . . . .  1093    1097#
BUTTON_ISR . . . . . . . . . . . .   984    1080#
BUTTON_TAB . . . . . . . . . . . .   496     499#
BUTTON_UP. . . . . . . . . . . . .  1094#

CHECK_F_SYSTEM . . . . . . . . . .   907     911#
CLK. . . . . . . . . . . . . . . .    76#
CMD_TABLE. . . . . . . . . . . . .   387#    619
CODE . . . . . . . . . . . . . . .   253#    253     255     898    1341
COM_MSG. . . . . . . . . . . . . .   338#    702
COM_NUMBER . . . . . . . . . . . .   360#    432     691     694     759
CR . . . . . . . . . . . . . . . .   145#    329     332     338     338     348     349     354     673
CSHARE . . . . . . . . . . . . . .   203#

DEBUG. . . . . . . . . . . . . . .    50#
DESCRIBE . . . . . . . . . . . . .   166#    197
DEV. . . . . . . . . . . . . . . .    76#
DEVCFG . . . . . . . . . . . . . .   203#
DEV_ATTR . . . . . . . . . . . . .   273#    274
DEV_DESCRIBE . . . . . . . . . . .   284#
DEV_DRIVER . . . . . . . . . . . .   258#    866
DEV_HEADER . . . . . . . . . . . .   274#
DEV_INTERRUPT. . . . . . . . . . .   265     800#    865
DEV_STRATEGY . . . . . . . . . . .   264     592#    596
DH_ATR . . . . . . . . . . . . . .   153#
DH_C_CLASS . . . . . . . . . . . .   157#
DH_MAJOR . . . . . . . . . . . . .   160#
DH_MINOR . . . . . . . . . . . . .   161#
DH_NAME_INDEX. . . . . . . . . . .   154#
DH_P_CLASS . . . . . . . . . . . .   156#
DH_V_CHILD . . . . . . . . . . . .   159#
DH_V_DEFAULT . . . . . . . . . . .   155#
DH_V_PARENT. . . . . . . . . . . .   158#   1112
DONE . . . . . . . . . . . . . . .    92#    631
DOS_ENTRY. . . . . . . . . . . . .   140#    661     819     826     835
DRIVER_ATTR. . . . . . . . . . . .   263#
DRIVER_NAME. . . . . . . . . . . .   266#
D_ABS_X. . . . . . . . . . . . . .   190#   1022    1023    1037    1058
D_ABS_Y. . . . . . . . . . . . . .   191#   1029    1030    1038    1059
D_ACCUM_X. . . . . . . . . . . . .   194#
D_ACCUM_Y. . . . . . . . . . . . .   195#
D_BURST_LEN. . . . . . . . . . . .   182#
D_CLASS. . . . . . . . . . . . . .   175#
D_DESC_MASK. . . . . . . . . . . .   171#
D_HPHIL_ID . . . . . . . . . . . .   170#
D_IO_MASK. . . . . . . . . . . . .   172#
D_MAX_AXIS . . . . . . . . . . . .   174#
D_PROMPTS. . . . . . . . . . . . .   178#
D_RD_REG . . . . . . . . . . . . .   184#
D_REL_X. . . . . . . . . . . . . .   192#   1025    1032    1054
D_REL_Y. . . . . . . . . . . . . .   193#   1056
D_RESERVED . . . . . . . . . . . .   181#
D_RESOLUTION . . . . . . . . . . .   187#
D_SIZE_X . . . . . . . . . . . . .   188#
D_SIZE_Y . . . . . . . . . . . . .   189#   1027
D_SOURCE . . . . . . . . . . . . .   168#
D_STATE. . . . . . . . . . . . . .   186#   1095    1099    1218
D_TRANSITION . . . . . . . . . . .   185#   1090
D_WR_REG . . . . . . . . . . . . .   183#
D_XDESC_MASK . . . . . . . . . . .   173#

END_OF_DRIVER. . . . . . . . . . .   850    1331#
ENTRY. . . . . . . . . . . . . . .   203#
ERROR. . . . . . . . . . . . . . .    92#    626
ERR_TYPE . . . . . . . . . . . . .    92#
EXIT . . . . . . . . . . . . . . .   631#    863

FALSE. . . . . . . . . . . . . . .    48#     49
FRAME_COUNT. . . . . . . . . . . .   371#    439     452     460
F_INQUIRE_ENTRY. . . . . . . . . .   220#    778
F_INS_BASEHPVT . . . . . . . . . .   212#
F_INS_XCHGFREE . . . . . . . . . .   213#    793
F_IO_CONTROL . . . . . . . . . . .   218#    806
F_ISR. . . . . . . . . . . . . . .   218#    521     570     906    1111
F_SYSTEM . . . . . . . . . . . . .   217#    912
```

RS-232 Mouse Driver

```
GIVE_TO_PARENT . . . . . . . . . .  1039    1064    1096    1110#

HP . . . . . . . . . . . . . . . .   203#
HPHIL_ADD. . . . . . . . . . . . .   382#
HPHIL_TABLE. . . . . . . . . . . .   378#
HP_ATTR. . . . . . . . . . . . . .   203#    273
HP_ENTRY . . . . . . . . . . . . .   225#    524     573     781     796     810     843     1113
HP_HEADER. . . . . . . . . . . . .   151#    163

IBM. . . . . . . . . . . . . . . .    76#
IC_1 . . . . . . . . . . . . . . .   670#    680
IC_10. . . . . . . . . . . . . . .   772     773#
IC_2 . . . . . . . . . . . . . . .   672     682#
IC_3 . . . . . . . . . . . . . . .   675     678     685     687     694#
IC_4 . . . . . . . . . . . . . . .   692     699#
IC_4A. . . . . . . . . . . . . . .   713     718#
INIT_3 . . . . . . . . . . . . . .   784     790#
INIT_BUTTON_STATE. . . . . . . . .  1216#   1218
INIT_CODE. . . . . . . . . . . . .   387     641#
INIT_EXIT. . . . . . . . . . . . .   820     827     837#
INIT_NO_PORT . . . . . . . . . . .   714     822#
INIT_NO_VECTOR . . . . . . . . . .   799     815#
INIT_OK. . . . . . . . . . . . . .   813     829#
INT_ENT. . . . . . . . . . . . . .   265#
INT_TABLE. . . . . . . . . . . . .   362#    760
IOCTL. . . . . . . . . . . . . . .    76#

ISR. . . . . . . . . . . . . . . .   203#

LAST_SYNCH . . . . . . . . . . . .   376#    462     463
LF . . . . . . . . . . . . . . . .   144#    329     332     338     338     349     349     355     676

MAP_CALL . . . . . . . . . . . . .   203#
MASK_TABLE . . . . . . . . . . . .   366#    770
MAX_PGID_SYS_FN. . . . . . . . . .  1159    1180#
MBUTTON. . . . . . . . . . . . . .   475#    535
MBUTTON_DOWN . . . . . . . . . . .   480     487#
MBUTTON_ISR. . . . . . . . . . . .   485     491#
MBUTTON_UP . . . . . . . . . . . .   482#
MNEXT_BUTTON . . . . . . . . . . .   478     532#
MOUSE_INT. . . . . . . . . . . . .   412#    763
MOVEMENT_ISR . . . . . . . . . . .   993#
MSD_BAD_LENGTH . . . . . . . . . .   109#
MSD_BLD_BPB. . . . . . . . . . . .   122#
MSD_CRC_ERROR. . . . . . . . . . .   108#
MSD_DEV_CLOSE. . . . . . . . . . .   134#
MSD_DEV_OPEN . . . . . . . . . . .   133#
MSD_GEN_FAILURE. . . . . . . . . .   116#
MSD_INIT . . . . . . . . . . . . .   120#    612
MSD_INPUT. . . . . . . . . . . . .   124#
MSD_IN_FLUSH . . . . . . . . . . .   127#
MSD_IN_NOWAIT. . . . . . . . . . .   125#
MSD_IN_STATUS. . . . . . . . . . .   126#
MSD_IOCTL_IN . . . . . . . . . . .   123#
MSD_IOCTL_OUT. . . . . . . . . . .   132#
MSD_MEDIA_CHK. . . . . . . . . . .   121#
MSD_NOT_RDY. . . . . . . . . . . .   106#
MSD_OUTPUT . . . . . . . . . . . .   128#
MSD_OUT_FLUSH. . . . . . . . . . .   131#
MSD_OUT_STATUS . . . . . . . . . .   130#
MSD_OUT_VERIFY . . . . . . . . . .   129#
MSD_PAPER_OUT. . . . . . . . . . .   113#
MSD_READ_FAULT . . . . . . . . . .   115#
MSD_REM_MEDIA. . . . . . . . . . .   135#    614
MSD_SEC_NOT_FOUND. . . . . . . . .   112#
MSD_SEEK_ERROR . . . . . . . . . .   110#
MSD_UNKNOWN_CMD. . . . . . . . . .   107#    627
MSD_UNKNOWN_MEDIA. . . . . . . . .   111#
MSD_UNKNOWN_UNIT . . . . . . . . .   105#
MSD_WRITE_FAULT. . . . . . . . . .   114#
MSD_WRITE_PROT . . . . . . . . . .   104#
MSE_NUM_BUTTON . . . . . . . . . .   199#    845
MSI_1. . . . . . . . . . . . . . .   441     446     450#
MSI_2. . . . . . . . . . . . . . .   454     459#
MSI_3. . . . . . . . . . . . . . .   465     541#
MSI_4. . . . . . . . . . . . . . .   550#
MSI_5. . . . . . . . . . . . . . .   448     455     551     578#

NO_PORT_MSG. . . . . . . . . . . .   340#    824
NO_VECTOR. . . . . . . . . . . . .   350#    817
NUL. . . . . . . . . . . . . . . .    76#

OCREM. . . . . . . . . . . . . . .    76#
OK_MSG . . . . . . . . . . . . . .   334#    833

PGID_DRIVER. . . . . . . . . . . .   787     899     905#    926
PGID_INIT. . . . . . . . . . . . .  1176    1215#   1222
PGID_ISR . . . . . . . . . . . . .   908     979#   1115
PGID_OPCODE_BAD. . . . . . . . . .   913     922#
PGID_START . . . . . . . . . . . .  1177    1252#   1256
PGID_STATE . . . . . . . . . . . .  1178    1286#   1289
PGID_SYSTEM. . . . . . . . . . . .   914    1157#   1181
PGID_SYS_BAD . . . . . . . . . . .  1160    1168#
PGID_SYS_CASE. . . . . . . . . . .  1167    1175#   1180
PGID_VECT_NUM. . . . . . . . . . .   383#    515     566     804
PGID_VERSION . . . . . . . . . . .  1179    1319#   1327
```

RS-232 Mouse Driver

```
PRINT_STR. . . . . . . . . . . . .  139#   660    818    825    834
PSHARE . . . . . . . . . . . . . .  203#

REL_MOVE . . . . . . . . . . . . .  995    997    1053#
REQ_HDR_OFF. . . . . . . . . . . .  315#   593    610    849
REQ_HDR_SEG. . . . . . . . . . . .  316#   594
REQ_HEADER . . . . . . . . . . . .   56#    69
RH_BPB . . . . . . . . . . . . . .   66#    71
RH_CMD_CODE. . . . . . . . . . . .   60#   611
RH_CMD_LINE. . . . . . . . . . . .   71#   667
RH_DRIV. . . . . . . . . . . . . .   67#
RH_END_OFF . . . . . . . . . . . .   64#   850
RH_END_SEG . . . . . . . . . . . .   65#   851
RH_LENGTH. . . . . . . . . . . . .   58#
RH_RESERVED. . . . . . . . . . . .   62#
RH_STATUS. . . . . . . . . . . . .   61#   626    627    631
RH_UNIT_CNT. . . . . . . . . . . .   63#
RH_UNIT_CODE . . . . . . . . . . .   59#
ROM. . . . . . . . . . . . . . . .  203#
RS_DONE. . . . . . . . . . . . . .  239#
RS_FAIL. . . . . . . . . . . . . .  240#  1006
RS_NO_VECTOR . . . . . . . . . . .  241#   798
RS_SUCCESSFUL. . . . . . . . . . .  237#  1220   1254   1320
RS_UNSUPPORTED . . . . . . . . . .  238#   783    923   1169   1287

SF_MOUSE_OVERRIDE. . . . . . . . .  223#   807
SIGN_ON_MSG. . . . . . . . . . . .  318#   659
SPEC . . . . . . . . . . . . . . .   76#
STACK_PTR. . . . . . . . . . . . .  357#   646    856
STACK_SEG. . . . . . . . . . . . .  358#
STACK_TOP. . . . . . . . . . . . .  651   1340#
STATUS . . . . . . . . . . . . . .   92#
STDI . . . . . . . . . . . . . . .   76#
STDO . . . . . . . . . . . . . . .   76#
STR. . . . . . . . . . . . . . . .  203#
STRAT_ENT. . . . . . . . . . . . .  264#
SUBADD . . . . . . . . . . . . . .  203#

TEMP_BUFFER. . . . . . . . . . . .  372#   450    461    542    544    546    548
TRUE . . . . . . . . . . . . . . .   49#    50
TYPE . . . . . . . . . . . . . . .  203#
T_ABS08. . . . . . . . . . . . . .  232#   998
T_ABS16. . . . . . . . . . . . . .  233#  1000
T_KC_BUTTON. . . . . . . . . . . .  229#   514    983
T_REL08. . . . . . . . . . . . . .  230#   994
T_REL16. . . . . . . . . . . . . .  231#   564    996

UNSUPPORT_CMD. . .             . .  388    389    390    391    392    393    394    395    396    397    398    399    400    401
                                    402    624#

UP_DOWN_BIT. . . . . . . . . . . . 1078#  1092

VERSION_LAB. . . . . . . . . . . .  330#   333   1325
VERSION_LEN. . . . . . . . . . . .  333#  1322
V_DOLITTLE . . . . . . . . . . . .  209#   274
V_LHPMOUSE . . . . . . . . . . . .  222#   274    808    844
V_SINPUT . . . . . . . . . . . . .  215#   522    571    779
V_SYSTEM . . . . . . . . . . . . .  211#   794

X. . . . . . . . . . . . . . . . .   76#

Y. . . . . . . . . . . . . . . . .   76#

Z. . . . . . . . . . . . . . . . .   92#

220 Symbols

50960 Bytes Free
```

APPENDIX H

H. ASCII AND SCANCODE CONVERSION TABLES

The following tables provide information for decimal-hexadecimal-ASCII conversions and Keystroke-scancode-Keycode conversions.

Table H.1

Decimal-Hexadecimal-ASCII Conversion

Dec	Hex	ASCII	Dec	Hex	ASCII	Dec	Hex	ASCII	Dec	Hex	ASCII
0	00	NUL	32	20	SP	64	40	@	96	60	'
1	01	SOH	33	21	!	65	41	A	97	61	a
2	02	STX	34	22	"	66	42	B	98	62	b
3	03	ETX	35	23	#	67	43	C	99	63	c
4	04	EOT	36	24	$	68	44	D	100	64	d
5	05	ENQ	37	25	%	69	45	E	101	65	e
6	06	ACK	38	26	&	70	46	F	102	66	f
7	07	BEL	39	27	'	71	47	G	103	67	g
8	08	BS	40	28	(	72	48	H	104	68	h
9	09	HT	41	29	)	73	49	I	105	69	i
10	0A	LF	42	2A	*	74	4A	J	106	6A	j
11	0B	VT	43	2B	+	75	4B	K	107	6B	k
12	0C	FF	44	2C	,	76	4C	L	108	6C	l
13	0D	CR	45	2D	—	77	4D	M	109	6D	m
14	0E	SO	46	2E	.	78	4E	N	110	6E	n
15	0F	SI	47	2F	/	79	4F	O	111	6F	o
16	10	DLE	48	30	0	80	50	P	112	70	p
17	11	DC1	49	31	1	81	51	Q	113	71	q
18	12	DC2	50	32	2	82	52	R	114	72	r
19	13	DC3	51	33	3	83	53	S	115	73	s

Dec	Hex	ASCII	Dec	Hex	ASCII	Dec	Hex	ASCII	Dec	Hex	ASCII
20	14	DC4	52	34	4	84	54	T	116	74	t
21	15	NAK	53	35	5	85	55	U	117	75	u
22	16	SYN	54	36	6	86	56	V	118	76	v
23	17	ETB	55	37	7	87	57	W	119	77	w
24	18	CAN	56	38	8	88	58	X	120	78	x
25	19	EM	57	39	9	89	59	Y	121	79	y
26	1A	SUB	58	3A	:	90	5A	Z	122	7A	z
27	1B	ESC	59	3B	;	91	5B	[	123	7B	{
28	1C	FS	60	3C	<	92	5C	\	124	7C	|
29	1D	GS	61	3D	=	93	5D	]	125	7D	}
30	1E	RS	62	3E	>	94	5E	^	126	7E	~
31	1F	VS	63	3F	?	95	5F	—	127	7F	DEL ∧

Table H.2

Scancode Conversion Table

Key Number	AT Scancode	Hp Scancode	Key Cap	Unshifted ASCII	Unshifted Hex	Shifted ASCII	Shifted Hex	Control	Alt
90	076H	001H	ESC	esc	1BH	esc	1BH	1BH	—
02	016H	002H	1	'1'	31H	'!'	21H	—	00/78H
03	01EH	003H	2	'2'	32H	'@'	40H	00H	00/79H
04	026H	004H	3	'3'	33H	'#'	23H	—	00/7AH
05	025H	005H	4	'4'	34H	'$'	24H	—	00/7BH
06	02EH	006H	5	'5'	35H	'%'	25H	—	00/7CH
07	036H	007H	6	'6'	36H	'^'	5EH	1EH	00/7DH
08	03DH	008H	7	'7'	37H	'&'	26H	—	00/7EH
09	03EH	009H	8	'8'	38H	'*'	2AH	—	00/7FH
10	046H	00AH	9	'9'	39H	'('	28H	—	00/80H
11	045H	00BH	0	'0'	30H	')'	29H	—	00/81H
12	04EH	00CH	—	'-'	2DH	'_'	5FH	1FH	00/82H
13	055H	00DH	=	'='	3DH	'+'	2BH	—	00/83H
15	066H	00EH	backspace	bs	08H	bs	08H	7FH	—
16	00DH	00FH	Tab	tab	09H	si	0FH	—	—
17	015H	010H	Q	'q'	71H	'Q'	51H	11H	00/10H
18	01DH	011H	W	'w'	77H	'W'	57H	17H	00/11H
19	024H	012H	E	'e'	65H	'E'	45H	05H	00/12H
20	02DH	013H	R	'r'	72H	'R'	52H	12H	00/13H
21	02CH	014H	T	't'	74H	'T'	54H	14H	00/14H
22	035H	015H	Y	'y'	79H	'Y'	59H	19H	00/15H
23	03CH	016H	U	'u'	75H	'U'	55H	15H	00/16H
24	043H	017H	I	'i'	69H	'I'	49H	09H	00/17H
25	044H	018H	O	'o'	6FH	'O'	4FH	0FH	00/18H

Key Number	AT Scancode	Hp Scancode	Key Cap	Unshifted ASCII	Hex	Shifted ASCII	Hex	Control	Alt
26	04DH	019H	P	'p'	70H	'P'	50H	10H	00/19H
27	054H	01AH	[	'['	5BH	'{'	7BH	1BH	—
28	05BH	01BH	]	']'	5DH	'}'	7DH	1DH	—
43	05AH	01CH	Enter	cr	0DH	cr	0DH	0AH	—
30	014H	01DH	CTRL	–	–	–	–	—	—
31	01CH	01EH	A	'a'	61H	'A'	41H	01H	00/1EH
32	01BH	01FH	S	's'	73H	'S'	53H	13H	00/1FH
33	023H	020H	D	'd'	64H	'D'	44H	04H	00/20H
34	02BH	021H	F	'f'	66H	'F'	46H	06H	00/21H
35	034H	022H	G	'g'	67H	'G'	47H	07H	00/22H
36	033H	023H	H	'h'	68H	'H'	48H	08H	00/23H
37	03BH	024H	J	'j'	6AH	'J'	4AH	0AH	00/24H
38	042H	025H	K	'k'	6BH	'K'	4BH	0BH	00/25H
39	04BH	026H	L	'l'	6CH	'L'	4CH	0CH	00/26H
40	04CH	027H	;	';'	3BH	':'	3AH	—	—
41	052H	028H	'	'''	27H	''''	22H	—	—
01	00EH	029H	'	'''	60H	'~'	7EH	—	—
44	012H	02AH	Left Shift	–	–	–	–	—	—
14	05DH	02BH	\	'\'	5CH	'|'	7CH	1CH	—
46	01AH	02CH	Z	'z'	7AH	'Z'	5AH	1AH	00/2CH
47	022H	02DH	X	'x'	78H	'X'	58H	18H	00/2DH
48	021H	02EH	C	'c'	63H	'C'	43H	03H	00/2EH
49	02AH	02FH	V	'v'	76H	'V'	56H	16H	00/2FH
50	032H	030H	B	'b'	62H	'B'	42H	02H	00/30H
51	031H	031H	N	'n'	6EH	'N'	4EH	0EH	00/31H
52	03AH	032H	M	'm'	6DH	'M'	4DH	0DH	00/32H
53	041H	033H	,	','	2CH	'<'	3CH	—	—
54	049H	034H	.	'.'	2EH	'>'	3EH	—	—
55	04AH	035H	/	'/'	2FH	'?'	3FH	—	—
57	059H	036H	Right Shift	–	–	–	–	—	—
106	07CH	037H	Prt Sc	'*'	2AH	–	–	00/72H	—
58	011H	038H	Alt	–	–	–	–	—	—
61	029H	039H	Space	' '	20H	' '	20H	20H	20H
64	058H	03AH	Caps lock	–	–	–	–	—	—
70	005H	03BH	F1	–	3BH	–	54H	00/5EH	00/68H
65	006H	03CH	F2	–	3CH	–	55H	00/5FH	00/69H
71	004H	03DH	F3	–	3DH	–	56H	00/60H	00/6AH
66	00CH	03EH	F4	–	3EH	–	57H	00/61H	00/6BH
72	003H	03FH	F5	–	3FH	–	58H	00/62H	00/6CH
67	00BH	040H	F6	–	40H	–	59H	00/63H	00/6DH
73	083H	041H	F7	–	41H	–	5AH	00/64H	00/6EH
68	00AH	042H	F8	–	42H	–	5BH	00/65H	00/6FH
74	001H	043H	F9	–	43H	–	5CH	00/66H	00/70H
69	009H	044H	F10	–	44H	–	5DH	00/67H	00/71H

Key Number	AT Scancode	Hp Scancode	Key Cap	NumLock or Shift		None Or NumLock and Shift	Control
95	077H	045H	Num lock	–	45H	—	—
100	07EH	046H	ScrLck	–	46H	—	—
91	06CH	047H	Home	'7'	37H	00/47H	0077H
96	075H	048H	↑	'8'	38H	00/48H	—
101	07DH	049H	Pg Up	'9'	39H	00/49H	00/84H
107	07BH	04AH	—	'-'	3AH	3AH	—
92	06BH	04BH	←	'4'	34H	00/4BH	00/73H
97	073H	04CH	5	'5'	35H	—	—
102	074H	04DH	→	'6'	36H	00/4DH	00/74H
108	079H	04EH	+	'+'	2BH	2BH	—
93	069H	04FH	End	'1'	31H	00/4FH	00/75H
98	072H	050H	↓	'2'	32H	00/50H	—
108	07AH	051H	Pg Dn	'3'	33H	00/51H	00/76H
99	070H	052H	Ins	'0'	30H	00/52H	—
104	071H	053H	DEL	'.'	2EH	00/53H	—
105	084H	054H	Sys req	–	–	—	—

Key Number	AT Scancode	Hp Scancode	Key Cap	Unshifted ASCII Hex	Shifted ASCII Hex	Control	Alt
		055H	- undef.				
		056H	- undef.				
		057H	- undef.				
		058H	- undef.				
		059H	- undef.				
		05AH	- undef.				
		05BH	- undef.				
		05CH	- undef.				
		05DH	- undef.				
59		05EH	Unlabled-L	00/D7H	00/BDH	00/A3H	00/89H
62		05FH	Unlabled-R	00/D8H	00/BEH	00/A4H	00/8AH
113		060H	CCP-Up	00/D9H	00/BFH	00/A5H	00/8BH
111		061H	CCP-Lft	00/DAH	00/C0H	00/A6H	00/8CH
115		062H	CCP-Dn	00/DBH	00/C1H	00/A7H	00/8DH
118		063H	CCP-Rht	00/DCH	00/C2H	00/A8H	00/8EH
110		064H	CCP-Home	00/DDH	00/C3H	00/A9H	00/8FH
117		065H	CCP-PgUp	00/DEH	00/C4H	00/AAH	00/90H
112		066H	CCP-End	00/DFH	00/C5H	00/ABH	00/91H
119		067H	CCP-PgDn	00/E0H	00/C6H	00/ACH	00/92H
116		068H	CCP-Ins	00/E1H	00/C7H	00/ADH	00/93H
120		069H	CCP-Del	00/E2H	00/C8H	00/AEH	00/94H
114		06AH	CCP-CNTR	00/E3H	00/C9H	00/AFH	00/95H
		06BH	- undef.	00/E4H	00/CAH	00/B0H	00/96H
		06CH	- undef.	00/E5H	00/CBH	00/B1H	00/97H
		06DH	- undef.	00/E6H	00/CCH	00/B2H	00/98H
		06EH	- undef.	00/E7H	00/CDH	00/B3H	00/99H
		06FH	- undef.	00/E8H	00/CEH	00/B4H	00/9AH
121		070H	f1	00/E9H	00/CFH	00/B5H	00/9BH
122		071H	f2	00/EAH	00/D0H	00/B6H	00/9CH
123		072H	f3	00/EBH	00/D1H	00/B7H	00/9DH
124		073H	f4	00/ECH	00/D2H	00/B8H	00/9EH
125		074H	f5	00/EDH	00/D3H	00/B9H	00/9FH
126		075H	f6	00/EEH	00/D4H	00/BAH	00/A0H
127		076H	f7	00/EFH	00/D5H	00/BBH	00/A1H
128		077H	f8	00/F0H	00/D6H	00/BCH	00/A2H
		078H through 7FH—undef.					

APPENDIX I

I. HEXADECIMAL ARITHMETIC

For use as a quick reference, the following tables are provided. Table I.1 shows the conversion from decimal-hexadecimal. Table I.2 is a simple hexadecimal addition table and table I.3 is a simple hexadecimal multiplication table.

Table I.1 converts from hexadecimal to/from decimal for the first 256 decimal numbers.

Table I.1

Decimal to Hexadecimal Conversion Chart

Dec	Hex	Dec	Hex	Dec	Hex	Dec	Hex
0	00	21	15	42	2A	63	3F
1	01	22	16	43	2B	64	40
2	02	23	17	44	2C	65	41
3	03	24	18	45	2D	66	42
4	04	25	19	46	2E	67	43
5	05	26	1A	47	2F	68	44
6	06	27	1B	48	30	69	45
7	07	28	1C	49	31	70	46
8	08	29	1D	50	32	71	47
9	09	30	1E	51	33	72	48
10	0A	31	1F	52	34	73	49
11	0B	32	20	53	35	74	4A
12	0C	33	21	54	36	75	4B
13	0D	34	22	55	37	76	4C
14	0E	35	23	56	38	77	4D
15	0F	36	24	57	39	78	4E
16	10	37	25	58	3A	79	4F
17	11	38	26	59	3B	80	50
18	12	39	27	60	3C	81	51
19	13	40	28	61	3D	82	52
20	14	41	29	62	3E	83	53

Dec	Hex	Dec	Hex	Dec	Hex	Dec	Hex
84	54	127	7F	170	AA	213	D5
85	55	128	80	171	AB	214	D6
86	56	129	81	172	AC	215	D7
87	57	130	82	173	AD	216	D8
88	58	131	83	174	AE	217	D9
89	59	132	84	175	AF	218	DA
90	5A	133	85	176	B0	219	DB
91	5B	134	86	177	B1	220	DC
92	5C	135	87	178	B2	221	DD
93	5D	136	88	179	B3	222	DE
94	5E	137	89	180	B4	223	DF
95	5F	138	8A	181	B5	224	E0
96	60	139	8B	182	B6	225	E1
97	61	140	8C	183	B7	226	E2
98	62	141	8D	184	B8	227	E3
99	63	142	8E	185	B9	228	E4
100	64	143	8F	186	BA	229	E5
101	65	144	90	187	BB	230	E6
102	66	145	91	188	BC	231	E7
103	67	146	92	189	BD	232	E8
104	68	147	93	190	BE	233	E9
105	69	148	94	191	BF	234	EA
106	6A	149	95	192	C0	235	EB
107	6B	150	96	193	C1	236	EC
108	6C	151	97	194	C2	237	ED
109	6D	152	98	195	C3	238	EE
110	6E	153	99	196	C4	239	EF
111	6F	154	9A	197	C5	240	F0
112	70	155	9B	198	C6	241	F1
113	71	156	9C	199	C7	242	F2
114	72	157	9D	200	C8	243	F3
115	73	158	9E	201	C9	244	F4
116	74	159	9F	202	CA	245	F5
117	75	160	A0	203	CB	246	F6
118	76	161	A1	204	CC	247	F7
119	77	162	A2	205	CD	248	F8
120	78	163	A3	206	CE	249	F9
121	79	164	A4	207	CF	250	FA
122	7A	165	A5	208	D0	251	FB
123	7B	166	A6	209	D1	252	FC
124	7C	167	A7	210	D2	253	FD
125	7D	168	A8	211	D3	254	FE
126	7E	169	A9	212	D4	255	FF

Table I.2

Hexadecimal Addition

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
0	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
1	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10
2	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11
3	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12
4	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13
5	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14
6	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15
7	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16
8	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17
9	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18
A	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19
B	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A
C	C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B
D	D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C
E	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D
F	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E

Table I.3

Hexadecimal Multiplication

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
2	0	2	4	6	8	A	C	E	10	12	14	16	18	1A	1C	1E
3	0	3	6	9	C	F	12	15	18	1B	1E	21	24	27	2A	2D
4	0	4	8	C	10	14	18	1C	20	24	28	2C	30	34	38	3C
5	0	5	A	F	14	19	1E	23	28	2D	32	37	3C	41	46	4B
6	0	6	C	12	18	1E	24	2A	30	36	3C	42	48	4E	54	5A
7	0	7	E	15	1C	23	2A	31	38	3F	46	4D	54	5B	62	69
8	0	8	10	18	20	28	30	38	40	48	50	58	60	68	70	78
9	0	9	12	1B	24	2D	36	3F	48	51	5A	63	6C	75	7E	87
A	0	A	14	1E	28	32	3C	46	50	5A	64	6E	78	82	8C	96
B	0	B	16	21	2C	37	42	4D	58	63	6E	79	84	8F	9A	A5
C	0	C	18	24	30	3C	48	54	60	6C	78	84	90	9C	A8	B4
D	0	D	1A	27	34	41	4E	5B	68	75	82	8F	9C	A9	B6	C3
E	0	E	1C	2A	38	46	54	62	70	7E	8C	9A	A8	B6	C4	D2
F	0	F	1E	2D	3C	4B	5A	69	78	87	96	A5	B4	C3	D2	E1

Glossary

Adapter: A circuit board containing electronic circuitry that interfaces a peripheral to the system processor board.

Adapter Card: See ADAPTER

Alphanumeric Display Mode: One of the Video Display Adapter modes. When this mode is selected, data is displayed in character cells, organized in rows and columns on the screen.

Application Programs: Software that performs application specific tasks. Word processors, spreadsheets, and data bases are examples of application programs.

Barcode Reader: An input device that is used to scan surfaces containing barcodes. The barcode reader converts barcodes into scancode data format, and transmits the scancodes to an input interface.

Baud Rate: The rate a signal changes state. When used with relationship to RS-232 ports, it is synonymous with the data transfer rate, expressed in bits-per-second (BPS).

BIOS: Basic Input/Output System. The BIOS is the code module that contains the drivers that constitute the software interface between the hardware, and system software and application programs.

Bootstrap: The process of initializing the system and loading system software after a reset.

Bucket: A data structure used by the EX-BIOS string functions for alphanumeric string management.

Character Code: A word returned by the keyboard driver indicating a key stroke. The character code consists of a keyboard scancode, and either an Extended (00H) or ASCII character.

Checksum: An error-checking protocol used to verify the integrity of a block of data or code. Each byte or word in the block is summed, then added to a Checksum Byte. The block of data or code is presumed valid if this sum equals a predefined value, usually 0.

Checksum Byte: A byte added to the sum of a block of code or data to produce a valid sum.

Child Driver: A child driver is called by another driver when it is unable to perform a function requested of it. Child drivers perform lower level or more hardware specific tasks than their calling drivers.

Clipping: The process utilized when dealing with graphics coordinates outside of the logical coordinate space. The Input System clips coordinates so that they don't exceed the boundaries of the logical coordinate space.

CMOS Memory: RAM memory on the Processor Board that is powered by both the system power supply and battery. When the system power is turned off, the contents of the RAM memory are preserved by the battery.

Code Module: A group of related processor instructions.

Code Segment (CS): The segment address of the code module currently being executed.

Coprocessor: An add-on processor that works with the 80286 processor that is found on the SPU. The 80287 is an example of a specialized coprocessor for floating point arithmetic.

Cursor Control Pad: The keypad containing cursor control keys.

Cylinder: A term used with multi-platter disc mechanisms, a cylinder is a group of sectors having the same track number on each of the platters.

Daisy Chain: A method of linking devices together in a serial configuration. Input devices on the HP-HIL loop are connected in a daisy chain.

Data Segment (DS): The segment address of the data currently being accessed.

Data Structures: A related group of data fields.

Describe Record: A data structure utilized by the Input System which contains information characterizing an input event.

Device: A physical piece of hardware, e.g. a touch screen, mouse, keyboard, dot matrix printer, ThinkJet, or LaserJet.

Disc Partitions: A group of cylinders within a hard disc volume allocated to a specified operating system, and its associated programs and data.

Disc Volumes: A group of cylinders comprising a logical disc. The optional 40 Mbyte hard disc is divided into two disc volumes containing 20 Mbytes each. The optional 20 Mbyte hard disc contains a single volume.

Divide By Zero Interrupt: The 80286 executes this interrupt any time a divide by zero operation is attempted. The vector to the service routine for this interrupt must be stored in memory locations 0000:0000H–0000:0003H.

DOS: Disc Operating System.

DOS Installable Device Driver: A device driver designed to be dynamically installed by DOS. DOS installable device drivers may be used to add EX-BIOS drivers to the system.

Driver: Code that interfaces to either a physical 'device' or another driver.

Driver Header: A data structure contained in the data area of each EX-BIOS driver. The driver header contains data fields that specify the attributes, mapping, and other parameters of the driver.

EX-BIOS: Extended BIOS. A set of proprietary HP drivers that provide support for various system features.

Extra Segment (ES): The segment address of the extra data segment currently being accessed.

Functions: Code modules within a driver that perform specific tasks. Individual driver functions are selected when a driver is called.

Function Keys: The ten industry standard keys labeled F1–F10 on the keyboard. See also HP SOFTKEYS.

GID: see GRAPHIC INPUT DEVICE.

Graphic Display Mode: A video display adapter mode in which all positions on the screen are addressable as pixels.

Graphic Input Device: An input device that generates positional and/or button state data. A mouse, tablet, and touch screen are examples of graphic input devices.

Graphics Sprite: See SPRITE.

Hardware Interrupts: Requests for interrupt service generated by the hardware components.

Head: The magnetic device that reads and writes data from a disc drive. Disc drives have a head for each recording surface in the mechanism. A flexible disc has two heads, while a hard disc head count can vary depending on the drive being used. The optional 20MB disc has two platters and four heads.

Hexadecimal: Numbers expressed in base 16. Hexadecimal notation is used throughout this manual to represent binary data. hexadecimal digits are represented with the numbers 0–9 and letters A–F. The hexadecimal numbers are indicated with an uppercase 'H' as their last character (i.e., 17H).

HP Extensions: Additional functions added to industry standard drivers that support EX-BIOS features and/or provide additional flexibility in programming industry standard system capabilities.

HP Global Data Area: A data structure located in the EX-BIOS Data Area containing variables common to two or more EX-BIOS drivers. In addition, the stack used by the EX-BIOS drivers is located here.

HP Softkeys: 8 function keys labeled f1–f8 on the keyboard. These keys can be mapped to return their own scancode, or they may emulate their respective industry standard function keys (F1–F8). See also FUNCTION KEYS.

HP__ENTRY__CODE: The code module that dispatches the EX-BIOS interrupt (6FH) to the selected driver.

HP__ENTRY: The symbolic reference for the EX-BIOS interrupt, 6FH.

HP-HIL Controller: The hardware that provides the electrical interface to the HP-HIL link and supervises the communication protocol.

HP-HIL Link: The electrical interface and communication protocol utilized to connect HP-HIL input devices.

HP-HIL Major Address: The primary address of an HP-HIL device. This is typically the link address of the device.

HP-HIL Minor Address: The secondary address of an HP-HIL device.

HP-HIL Universal Address: Used to broadcast commands to all HP-HIL devices. The Universal Address is implemented as Address 0 in the HP-HIL protocol.

HP__VECTOR__TABLE: A data structure containing the IP, CS, and DS of all EX-BIOS drivers. This data structure is utilized by the HP__ENTRY__CODE to branch to the selected EX-BIOS driver.

Input System: A set of EX-BIOS drivers that service the input devices. The Input System supports the keyboard, HP Mouse, HP touch screen, and other HP-HIL input devices. It can be expanded to encompass non-HP-HIL input devices.

Instruction Pointer (IP): The offset from the base of the code segment of the next instruction to be executed.

Interrupt Service Routine: A code module, and its associated data structure(s) that responds to a hardware interrupt.

ISR Event Record: A data structure used by the Input System which contains information characterizing an input event.

Interleave: The number of physical sectors on a disc drive skipped when reading consecutive logical sectors on the same track. See also STAGGER.

Interrupt Vector: A data structure used by the 80286 to branch to a service routine or an interrupt. Interrupt vectors are located in the first 1024 bytes of system memory. Each interrupt vector occupies 2 words of memory and contains the IP and CS of the interrupt service routine.

KB: KiloBytes. 1024 bytes.

Keyboard: The physical keyboard.

Keyboard Controller (8041): The 8041 keyboard controller. The 8041 provides industry standard keyboard compatibility, and serves as a buffer between the STD-BIOS keyboard drivers and the Input System.

Keyboard Modifier: One of the special keyboard keys that modifies the interpretation of the other keys. The keyboard modifiers are the CTRL, ALT, SHIFT, CAPS LOCK, NUM LOCK, and SCROLL LOCK keys.

LED Mode Indicators: The LEDs located on the keyboard that indicate the state of the CAPS LOCK, NUM LOCK, and SCROLL LOCK keyboard modifiers.

Logical Driver: A driver responsible for interfacing with the Operating System or application.

Logical Keyboard: A set of drivers within the Input System that service the physical keyboard.

MB: MegaByte. 1,048,576 bytes.

MICKIES: The number of physical coordinates per inch reported by a mouse or other relative GID device.

Mouse: A GID device that reports relative motion coordinates based on its motion. A mouse will also report the state of its buttons.

MS-DOS: See DOS.

Multi-Tasking: The ability of a CPU to perform multiple jobs or tasks simultaneously. Multitasking is accomplished by dividing CPU execution time between the different tasks. If this task-switching is performed quickly enough, the illusion of simultaneous execution occurs.

Numeric Keypad: The keypad containing numeric and modifier keys.

NMI: Non-Maskable Interrupt. This is an 80286 interrupt line used to report system error conditions. This interrupt is mapped by the 80286 to Interrupt vector 02H.

Operating System: The system software that provides access to system resources for application programs. The operating system manages input and output, data and program files, and system memory.

Palette: The set of all possible colors the Video Display Adapter can produce. The Multimode Video Display Adapter has a palette of 16 colors.

Parallel Port: An I/O port that transmits and receives data a byte at a time. The parallel ports are typically used to interface to printers.

Parent Driver: A parent driver is called by another driver when the second is unable to perform a function requested of it. Parent drivers perform higher level or more system software oriented tasks than their calling drivers.

Physical Driver: A driver responsible for interfacing with the physical hardware.

Pixel: A dot on the screen in the graphics modes.

Polling: The process of periodically determining the status of a device. Polling is used to determine if peripheral devices have data or are ready to accept data in non-interrupt driven systems.

Post: Power On Self Test. The POST process is executed each time the system is powered on or a hard reset occurs.

Processor Interrupts: Interrupts generated by the 80286 processor in response to error conditions or processor exceptions.

Protected Mode: One of the two modes that the 80286 can operate in. The Protected mode provides virtual memory addressing, on-board memory management and protection, and task switching to support multi-user, multi-tasking system software.

RAM BIOS: The interface between DOS and the ROM BIOS. It is dynamically loaded at system boot with DOS.

Real Mode: One of the two modes that the 80286 can operate in. The Real mode provides compatibility with the 8086 family of microprocessors.

Real-Time Clock: A clock circuit that maintains the correct time whether the system is on or off. The real-time clock is powered by both the system power supply and battery. When the system power is turned off, the clock continues to operate from the battery.

Return Status Code: A code returned by the EX-BIOS drivers that indicates the status of the function requested.

ROM BIOS: The set of EX-BIOS and STD-BIOS drivers. These code modules are contained in ROM modules on the Processor Extension Card.

ROM Module: Code and/or data stored in an EPROM or ROM.

RS-232C: An EIA standard for a serial data interface. Often used as a synonym for serial when referring to system ports.

Scaling: The process of adjusting physical graphics coordinates to fit in a proportionately larger or smaller logical space. The Input System scales the coordinates received from a tablet to fit into its logical space.

Scancodes: Codes returned by the physical keyboard to indicate key makes and breaks.

Sector: A physical location on the disc where a block of data is stored. Disc surfaces are divided into concentric rings called tracks. These rings are in turn divided into sectors.

Serial: To transmit data one bit at a time, serially. Used to indicate system ports that transmit data in this fashion. See also RS-232C.

Single Step Interrupt: A processor interrupt generated after each instruction if the Single Step flag is set. This interrupt is mapped by the 80286 to Interrupt vector 01H.

Software Interrupts: Interrupts generated by the 80286 INT 'n' instruction where 'n' is the interrupt number.

Sprite: A graphics cursor. The sprite is controlled by the Input System V__STRACK and V__LHPMOUSE drivers.

Stagger: Disc stagger is the track to track offset between logical sectors. Stagger increases disc performance during sequential read operations by adjusting for track to track access time. See also INTERLEAVE.

STD-BIOS: The set of drivers that execute the industry standard BIOS functions.

System Software: See Operating System.

System Strings: Character strings stored in memory. Each EX-BIOS driver has a system string associated with it. System strings are designed to provide a simple method for system software to access them. In addition, their implementation provides a simple and effective method of localization.

Tablet: A Graphics Input Device (GID) that generates absolute graphics coordinates.

Timeout: An indication (for example an interrupt) that indicates that a predetermined time has elapsed waiting for an event to occur. Timeouts are used to prevent the system from hanging up waiting for an event to happen that doesn't. For example, a timeout can be used to abort a print operation if the printer does not return a ready status.

Timer Tick: An interrupt generated by the system timer. It is initialized to produce approximately 18.2 timer ticks per second.

Touch Screen: An HP Graphic Input Device (GID). allows a user to input data by physically touching the display screen.

Track: An Input System driver that moves a Sprite on the display screen in response to graphics motion received from GID devices.

Tracking: The process of moving a Sprite on the display screen in response to graphic motion received from GID devices.

Typematic Delay: The amount of time a key must remain depressed before the keyboard enters the typematic or repeat mode.

Typematic Rate: The rate at which make scancodes are transmitted by the keyboard when it is in the typematic or repeat mode.

Video Attributes: Video characteristics of characters displayed on the Video Display Adapter. Video attributes include reverse video, blinking, underline, and high intensity. Video attributes only apply to characters displayed in the alphanumeric modes.

References

HP Vectra MS-DOS User's Reference Manual

HP Vectra MS-DOS Programmer's Reference Manual
—Discusses programming of the 80286 using MS-DOS.

HP-HIL Technical Reference Manual
—Discusses the HP-HIL controller.
—Discusses the HP-HIL link.

HP Vectra MS-DOS Macro Assembler
—Reference for the assembler.

INTEL iAPX286 Programmer's Reference Manual
—Reference for 80286 instruction set and architecture.
—Reference for the 80287 numeric processor.

INTEL iAPX286/10 Hardware Reference Manual
—Discusses the 80286 processor.

INTEL Microcontroller Handbook
—Discusses the 8041 keyboard controller chip.

INTEL Microsystem Components Handbook, Volume II
—Discusses the 8254 timer chip.
—Discusses the 8237A DMA controller chip.

Motorola Single Chip Microcomputer Data, Section C
—Discusses the MC146818 real time clock/ CMOS chip.

Motorola 8-Bit Microprocessor & Peripheral Data
—Discusses the 6845A video controller chip.

INTEL Microprocessor and Peripheral Handbook
—Discusses the 8237A DMA controller.
—Discusses the 82284 clock chip.
—Discusses the 8041 keyboard controller chip.
—Discusses the 8254 timer chip.

INTEL the 8086 Family User's Manual

NEC Electronics Microcomputer Products Data Book
—Discusses the 765A flexible disc controller chip.

The Peter Norton Guide to the IBM PC
by Peter Norton, Microsoft Press.

INDEX

HEWLETT
PACKARD

Printed in U.S.A.
Part Number 45961-90002