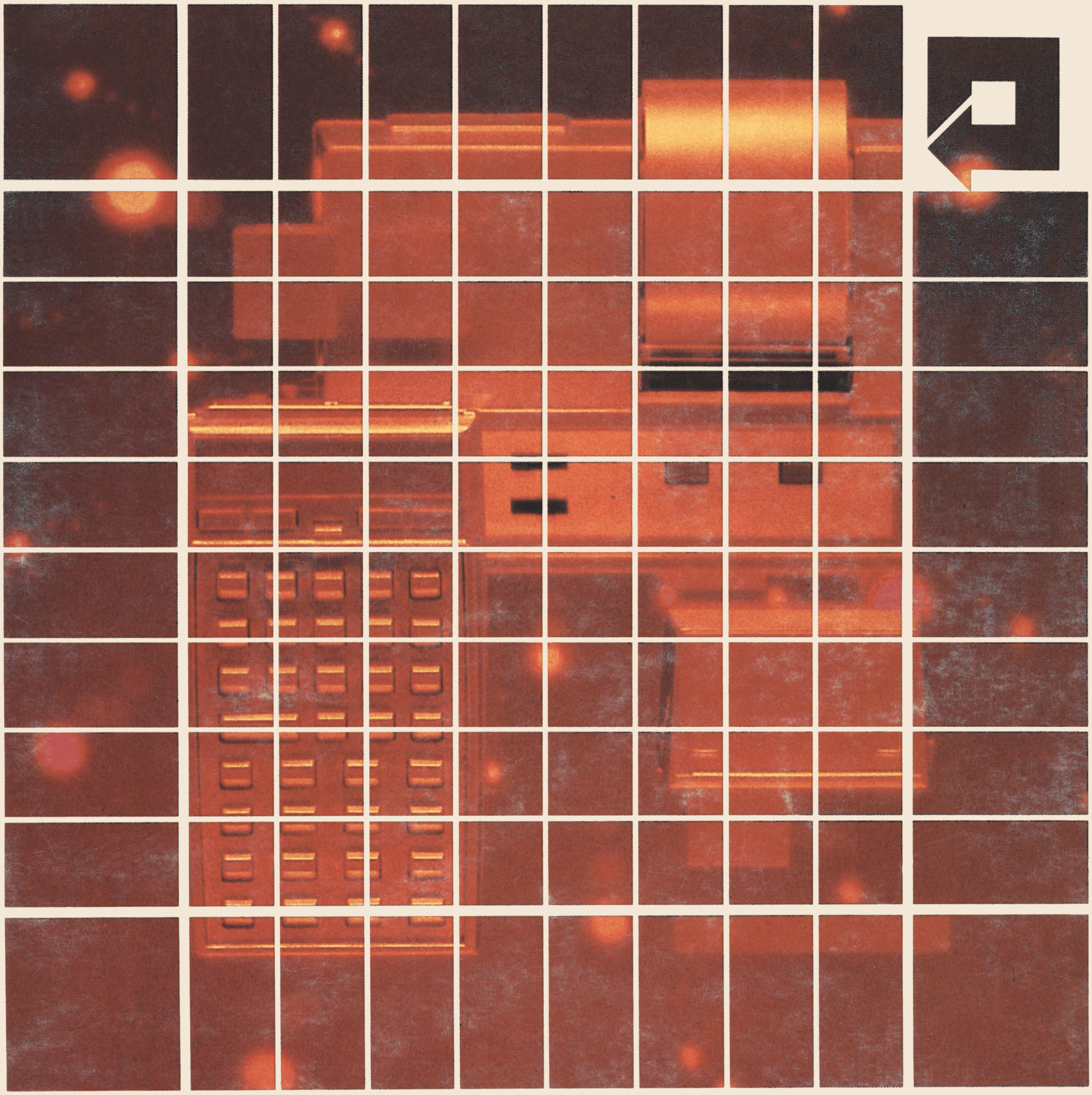
HEWLETT-PACKARD

HP 00041-15043

HP-IL Development Module

OWNER'S MANUAL



Hewlett-Packard

HP 00041-15043
HP-IL Development Module

Owner's Manual

September 1982

00041-90449

Printed in U.S.A.

Contents

Introduction

The Development Module is intended to be used for debugging an HP-IL implementation or performing HP-IL transactions which cannot be performed with other HP products.

With the Development Module you can:

1) Make the HP-41 an HP-IL analyzer.
2) Perform HP-41 internal byte transfers between any two of:
   a) The stack (X or X,Y registers).
   b) The ALPHA register.
   c) A sequence of registers.
   d) The HP-IL loop.
   e) An array of bytes (the buffer).
3) Set up an array of arbitrary bytes (the buffer).
4) Input and output numbers in hexadecimal, octal, or binary.
5) Perform some binary functions.
6) Send most HP-IL messages by specifying their mnemonic.
7) Test any of the status bits in the HP-IL integrated circuit.
8) Read or write any HP-IL message.
9) Read or write any register in the HP-IL integrated circuit.

This manual assumes that you are familiar with the HP-IL protocol as specified in <u>The HP-IL System</u>*. You need to understand HP-IL protocol for many of the module's functions to be useful. This manual does not provide tutorial information about HP-IL. For more detail on HP-IL, see the defining document: "HP-IL Interface Specification"*. The HP-IL integrated circuit used in the HP-41 is very similar to the general purpose integrated circuit described in "The HP-IL Integrated Circuit"*. The differences are detailed in Appendix C.

You do not need to read this manual front to back. In fact, it is possible that one function, or set of functions will do the job for you.

The manual is organized in two sections: an example section, and a reference section.

The example section gives a number of examples of "how to do it." Hopefully, the solution to your problem will be illustrated by one of the example applications of the development module. Refer to the reference section for a detailed description of each of the module's functions.

* <u>The HP-IL System</u>
By Gerry Kane
Steve Harper
David Ushijima
An Osborne/McGraw-Hill book
Berkeley, California

"The HP-IL Interface Specification"
HP 82166-90017
Hewlett-Packard
1000 NE Circle Blvd.
Corvallis, Or. 97330

"The HP-IL Integrated Circuit"
HP 82166-90016
Hewlett-Packard
1000 NE Circle Blvd.
Corvallis, Or. 97330

Section 1

HP-IL Analyzer

[SCOPE] allows the HP-41 to monitor and display the HP-IL messages as they go around the loop. In SCOPE mode, the HP-41 will no longer source frames, but will merely display the mnemonic of a received message and then echo the message on to the next device.

After you execute [SCOPE], the display will read "SCOPE READY". Then, as messages are received from the loop, their mnemonics are placed in the display one at a time. Before the message is retransmitted, a one second delay is inserted. This gives you time to read the display before the next frame comes in. SCOPE mode is exited by pressing [R/S].

[SCOPE] takes control of the HP-41 keyboard. This means that only two keys can be used to exit SCOPE mode - [OFF] and [R/S]. Most of the other keys have no effect. The ones that do have an effect are described below.

The delay between messages can be changed to zero seconds, one half second, one second, or one and a half seconds. These changes are caused by pressing [0], [1], [2], or [3] respectively. If the delay is 1.5 seconds, and an HP 82143A Thermal Printer is plugged into the HP-41, the messages will be printed as they are displayed.

The messages may be stored in a buffer as they are received. The buffer is maintained in the memory not occupied by programs or data. The buffer is created using [BSIZEX]. The buffer also has a pointer associated with it. Refer to the reference section for additional information on the buffer.

Each time [STO] is pressed while the HP-41 is in SCOPE mode, the HP-41 will be toggled into or out of store mode. While in store mode the messages are stored into the buffer.

Once the messages are in the buffer, they may be viewed using [SST]. Each time [SST] is pressed, the SCOPE pointer is advanced and the next message is placed in the display. Pressing [BST] causes the pointer to back up to the previous message. The previous message is then placed in the display. The shift indicator stays on until [SHIFT] is pressed again, thus maintaining the BST function of the key.

An example of SCOPE mode usage:

Keystroke	Frame received	Display
[14]		14
[BSIZEX]		14
[XEQ] "SCOPE"		SCOPE READY

Now that the scope is ready, initate the HP-IL transaction that you wish to observe.

	AAU	AAU
	AAD 1	AAD 1
	TAD 1	TAD 1
	RFC	RFC
	SAI	SAI
	DAB 16	DAB 16
	UNT	UNT
	RFC	RFC

Until now the messages have been staying in the display for .5 seconds. Press [3] and start the sequence again. The messages may be printed on the HP 82143A printer if one is connected.

[3]		1.5 SEC DELAY
	AAU	AAU
	AAD 1	AAD 1
	TAD 1	TAD 1
	RFC	RFC
	SAI	SAI
	DAB 16	DAB 16
	UNT	UNT
	RFC	RFC

So far the messages have not been stored in the buffer. Press [STO] to store the messages in the buffer. If you press [0] so there is no delay while storing messages into the buffer then the loop will respond much faster.

[STO]		STORE MODE
[0]		0 SEC DELAY
	AAU	AAU
	AAD 1	AAD 1
	TAD 1	TAD 1

	RFC	RFC
	SAI	SAI
	DAB 16	DAB 16
	UNT	UNT

The previous message just filled the buffer. The next message will result in the "END OF BUF" message. If you decide to not store any more messages, press [STO] to toggle out of store mode.

[STO]	RFC	RFC

To review the messages that have just been received, press [BACKARROW], which sets the pointer to the start of the sequence.

[backarrow]	00 AAU

You can now single step through the buffer using [SST].

[SST]	02 AAD 1
[SST]	04 TAD 1
[SST]	06 RFC
[SST]	08 SAI
[SST]	10 DAB 16
[SST]	12 UNT

If [SST] is pressed again, the UNT message stays in the display because the pointer is at the end of the buffer. If you want to back up to the SAI message, press [SHIFT] followed by pressing [BST] twice. Each time [BST] is pressed the pointer and display move back by one message. Note that pressing [BST] does not clear the shift indicator.

[BST] (shifted)	10 DAB 16
[BST] (shifted)	08 SAI
[BST] (shifted)	06 RFC
[BST] (shifted)	04 TAD 1
[BST] (shifted)	02 AAD 1

Section 2

Example Programs.

How to Read This Section

The programs in this section have been listed differently from the HP-41 manual. The listing is similar to that produced by trace mode. Line numbers are not used because they are irrelevent to the program. In some cases, multiple program steps are used on a single line. This may cause some confusion as to where one step ends and the next begins. Entering the program will clear all doubts. Details of individual development module functions are given in Section 3.

The following two programs demonstrate the use of the low level message handling functions.

Program: RGPIO

Description: Reads any control register of an HP-IL to GPIO interface (HP 82165A/HP 82166A.)

Input: The X-register contains the number of the register to be read.

Output: The X-register gets the contents of the register.

Warnings: This program changes the contents of the stack. Uses register 00 to contain the register number. It uses register 02 for the GPIO interface's address, flag 09 to store flag 33, and calls GPADR. You need a buffer of at least 18 bytes for this program to work.

```
*LBL "RGPIO"
 STO 00          Save the desired register.
 XEQ "GPADR"     Find the desired GPIO.
 TAD             Make it talker addressed.
 0 DDT           Tell it to send the registers.
 0 PT=           Start at the beginning of the buffer.
 RCL 00 1 +
 INBUFX          Read only up to the register he wants.
 RCL 00 PT=      Point to the register he wants.
 1 BUF-XB        Get one register from the buffer.
 UNT             Tell the GPIO to stop sending registers.
 FS? 09          Restore flag 33 to its original value.
 CF33
 RTN
```

```
 END
```

Program: WGPIO
Description: This program writes a control register of an HP-IL to GPIO interface (HP 82165A/HP 82166A).
Input: The X-register contains the new contents of the register, the Y-register contains the number of the register to be changed.
Output: none.
Warnings: This program changes the contents of the stack. Register 00 contains the register number, register 10 contains the GPIO register contents, register 02 contains the GPIO interface's address, flag 09 stores flag 33, manual increment mode is entered, and GPADR is called. You need a buffer of at least 18 bytes for this program to work.

```
*LBL"WGPIO"
 STO 01          Save the new register contents.
 RDN             Move the register number into the X-register.
 MIPT            Don't advance the buffer.
 XEQ "RGPIO"     Get the registers into the buffer.
 RCL 01          Put the new contents into the X-register.
 X-BUF           Store it using the PT value from RGPIO.
 RCL 02          Get the GPIO address.
 LAD             Make it listener active.
 0 DDL           Tell it to write its registers.
 RCL 00 1 +      Get the register number again.
 OUTBUFX         Send out this register and all previous ones.
 UNL             Unlisten the GPIO.
 FS? 09          Restore flag 33 to its original value.
 CF33
 END
```

Program: GPADR
Description: This program finds the first GPIO interface on the loop by searching for a GPIO accessory ID (64). A variation of this program, GPADR1, searches for a GPIO device ID.
Input: none.
Output: The X-register contains the interface address.
Warnings: Register 02 stores the device's address, the contents of stack are altered, the ALPHA register is modified, flag 33 is saved in flag 09, and flag 33 is set.

```
*LBL "GPADR"
 SF 09           Save the complement of flag 33 into flag 09.
 FS? 33 CF 09
```

```
 SF33
 0 ENTER^
 193 WREG         Set SC, CA, and MCL (AAU clears MCL).
 FS? 32           AUTOIO or MANIO?
 GTO 00           Go to label 00 if manual IO.
 AAU              Auto address the loop.
 1 AAD
 1 -
 1000 /           Make an index of the form: bbb.eee, where bbb
 +                 is the address of the first device, and eee
 STO 02            is the address of the last device.

*LBL 01
 RCL 02           Get the counter into the X-register,
 TAD               and make it the active talker.
 SAI              Get the Accessory ID of this device.
 64 X=Y?          Is this equal to the Accessory ID of the GPIO?
 GTO 02           Yes! exit with reg 02 = address of the GPIO.
 ISG 02           Any more devices to try?
 GTO 01           Yes - go try them.
 "NO GPIO"        No - tell him that we can't find it.
 PROMPT

*LBL 02           We got it - get the address
 RCL 02            into the X-register,
 UNT               untalk the GPIO, and return
 RTN

*LBL 00           Manual I/O is very simple.
 4                The HP-IL module stores the selected address
 RREG              in register 4 of the HP-IL chip.
 STO 02           Return the address in X and register 02.
 RTN
 END
```

Program: GPADR1
Description: This program finds the first GPIO interface on the loop by searching for a GPIO device ID ("HP82165A" or "HP82166A"). A variation of this program, GPADR, searches for a GPIO accessory ID.
Input: none.
Output: The X-register gets the interface's address.
Warnings: The ALPHA register is modified, flag 33 is saved in flag 09, and flag 33 is set.

```
*LBL"GPADR1"
```

```
 "HP82165A"
 FINDID           Look for the specified Device ID.
 X=0?             Did we find it?
 GTO 01           No, search for other GPIO.
 GTO 03
*LBL 01
 "HP82166A"
 FINDID
 X=0?             Again, did we find it?
 GTO 02           No, put "NO GPIO" in the display
 GTO 03
*LBL 02
 "NO GPIO"
 PROMPT
*LBL 03           We found the GPIO - its address is in X.
 SF 09            Store the complement of flag 33 in flag 9.
 FS? 33 CF 09
 SF33
 RTN
```

Program: ENBINTR
Description: This program sets the HP-IL chip to automatically send IDY messages around the loop while remaining in idle mode.
Input: none.
Output: Sets HP-IL register 0 to 64 (CA), sets HP-IL register 3 to 64 (AIDY), and sets user flag 18, which enables the INTR routine.

```
*LBL "ENBINTR"
 3 ENTER^
 64 WREG          Set HP-IL register 3 to 64 (AIDY).
 0 ENTER^
 64 WREG          Set HP-IL register 0 to 64 (CA).
 SF 18            Enable the INTR program.
 END
```

Program: INTR
Description: This program is executed when the HP-41 enters execution mode with flag 18 set and any one of SRQR, INTR, FRAV, or FRNS true. This program is enabled by executing the program ENBINTR. This routine only executes a "TONE 9", but an interrupt handling routine could be used in place of the "TONE 9".
Input: none
Output: An IDY is sent out to clear the SRQR bit in the

requesting device. The SRQR bit is cleared only when a message is received with the SRQ bit zero. (If no message is sent out then an infinite loop is formed which is difficult to break out of. Pressing a key for a long time usually breaks the infinite loop.)

Warnings: The data bits in the automatically sent IDYs sometimes change, which cause FRNS to be set and start the HP-41 executing. Reading the message will reset FRNS.

```
*LBL "INTR"         Must be "INTR"
 FRNS?              Did we start executing because of FRNS?
 RFRM               Yes, read the message.
 SRQR?              Did we start executing because of SRQR?
 TONE 9             Yes, perform all the work we're going to.
 SRQR?              Did we start executing because of SRQR?
 IDY                Send out an IDY to clear SRQR.
 RTN
 END
```

Program: BINCALC

Description: This program, with the use of some assigned keys, emulates an HP-16C, the Computer Scientist. You may calculate in hexadecimal (HEX), octal (OCT), or binary (BIN) modes. The base is indicated by the number stored in register 00. 16 means HEX, 8 means OCT, and 2 means BIN. The meanings of the keys on the keyboard are kept by this program. Enter still means enter, times still means times, etc. This program is useful because it displays and enters numbers in the desired base. To change bases, the shifted keys 2 (for BIN), 8 (for OCT), and 6 (for HEX) are used.

Input: Numbers typed from the keyboard.

Output: Results in the display.

Warnings: This program is not quite as fast as an HP-16C, nor does it handle word sizes as large as sixty-four bits. Remember that the largest number is thirty-two bits long, and the actual number of bits viewable is dependent upon the current base.

```
*LBL "BINCALC"

*LBL "16"                 Set the current base - HEX.
 16 STO 00 RDN GTO 01
*LBL "2"                  BIN
 2 STO 00 RDN GTO 01
*LBL "8"                  OCT
```

```
 8 STO 00 RDN GTO 01

*LBL "-"        Various arithmetic and logical operations:
 - GTO 01
*LBL "+"
 + GTO 01
*LBL "*"
 * GTO 01
*LBL "/"
 / GTO 01
*LBL "OR "      Note the trailing space.  Also see AND, XOR, NOT.
 OR GTO 01
*LBL "AND "
 AND GTO 01
*LBL "XOR "
 XOR GTO 01
*LBL "NOT "
 NOT GTO 01
*LBL "^"                    Enter
 XEQ 01 GTO 01

*LBL 01                Keep the result within 32 bits.
 4294967296 MOD        Magic number = FFFF,FFFF Hex.

 GTO IND 00            Go to the correct 'VIEW and 'IN function.
*LBL 02
 BINVIEW BININ RTN
*LBL 08
 OCTVIEW OCTIN RTN
*LBL 16
 HEXVIEW HEXIN RTN
 END
```

Key Assignments:

```
 41 "^"
-42 "NOT "      Note the trailing space.  Also see OR, AND, XOR
 51 "-"
-53 "8"
 61 "+"
-61 "OR "
-64 "16"
 71 "*"
-71 "AND "
-73 "2"
 81 "/"
```

-81 "XOR "

Program: DEVICE

Description: This is actually a set of programs designed to work together. The first, DEVICE, sets up for INTR. The second, INTR, handles incoming messages. The third, SENDA, causes the ALPHA register to be sent.

The program INTR gives the HP-41 the following capability: L1,3 T1,2,3,4,6 SR1 AA1. The controller may at any time read the contents of the buffer. The first byte of the buffer is assumed to be the length of the data in the buffer.

Warnings: This program is slow enough that the HP-IL module will time out and give a transmit error. In order to drive this program, a controller program must be written with the development module. This program is given on page 19.

This program initializes the HP-IL chip, the buffer, and sets flag 18 so that INTR may be executed.

```
*LBL "DEVICE"
 25 BSIZEX AIPT             Create the buffer.
 3 ENTER^ 64 WREG           Clear OSCDIS, set AIDY.
 0 ENTER^ 0 WREG            Clear everything in register zero.
 4 ENTER^ 31 WREG           Initially unaddressed.
 SF 18                      Set flag to enable "INTR".
 FIX 0 CF 28 CF 29          Get rid of radix indicators.
 END
```

This program handles incoming frames.

```
*LBL "INTR"                 Must be named INTR.
 IFCR? GTO 50               Special check for IFC.
 RFRM 256 * OR              Put entire message in the X-register.
 SF 08                      Flag 8 is true if first try.
*LBL 10
 ENTER^                     Put a copy of message on stack.
 CLA ARCL X ASTO X          Put message into X as alpha.
 SF 25 GTO IND X            Search for the message.
 FS?C 08                    If not found on first try
 GTO 11                      then go search again.
*LBL 12                     We didn't recognize this message.
 RFRM WFRM                  Retransmit the message.
 RTN
```

```
*LBL 11                        We didn't recognize this message.
 RDN                           Get it back and ignore
 2016 AND                       the lower five bits to catch
 GTO 10                         TAD, LAD, AAU, etc.

*LBL 50                        Handle IFC by setting IFCR.
 0 ENTER^ 2 WREG
 GTO 03

*LBL "1056" CF 25              ---- LAD ----
 XEQ 01 X=0? GTO 03            If not our address then leave.
 0 RREG 16 OR 223 AND WREG     Set LA and clear TA.
 GTO 03

*LBL "1087" CF 25              ---- UNL ----
 0 RREG 239 AND WREG           Clear LA.
 GTO 03

*LBL "1088" CF 25              ---- TAD ----
 XEQ 01 X=0? GTO "1119"        If not our address then UNT.
 0 RREG 32 OR 239 AND WREG     Set TA and clear LA.
 GTO 03

*LBL "1119" CF 25              ---- UNT ----
 0 RREG 223 AND WREG           Clear TA.
 GTO 03

*LBL "1178" CF 25              ---- AAU ----
 4 ENTER^ 31 WREG              Remember that we don't have
 GTO 03                         an address.

*LBL "1376" CF 25              ---- SDA ----
 0 RREG 247 AND WREG           Clear SSRQ.
 0 PT= MIPT                    Point to the beginning.
 BUF-XB 1 + OUTBUFX            Get the count and output.
 AIPT                          Restore auto-increment and exit.
 RTN

*LBL "1377" CF 25              ---- SST ----
 0 RREG 8 AND                  Get SSRQ.
 28 ROTXY                      Put it into bit 7.
 OUTBIN                        Send 0 (no SRQ) or 128 (SRQ).
 GTO "ETO"

*LBL "1378" CF 25              ---- SDI ----
 1213214004 OUTBIN
```

```
 49 OUTBIN                    Send "HP-41" and ETO.
 GTO "ETO"
 RTN

*LBL "1379" CF 25             ---- SAI ----
 0 OUTBIN                     Send DAB 0 and ETO.
*LBL "ETO"
 64 ENTER^ 5 WFRM
 RTN

*LBL "1408" CF 25             ---- AAD ----
 4 RREG 31 X>Y?               Are we already addressed?
 GTO 12                       Yes - just echo the message.
 RFRM CLX 31 AND 4
 X<>Y WREG                    Set our address.
 RFRM X<>Y 1 + X<>Y WFRM      Increment the address and send it on.
 RTN

*LBL "1439"                   Explicitly ignore IAD.  We need
 GTO 12                        this to catch AAD 31.

*LBL 03                       Exit point for all CMD messages.
 0 RREG 4 OR WREG             Set SLRDY.
 RTN

*LBL 01                       If the incoming message contains our
 2 RREG 4 RREG                 address then return X<>0
 RCL Z 31 AND X=Y? RTN         else return X=0.
 CLX
 END
```

This program must be executed with the string that you want to send in ALPHA. When this program finishes, it must go to idle mode, otherwise INTR will not be executed. This program assumes that the controller is sending IDY's around the loop. The next IDY to go around will have the service request bit set, and the controller will ask the HP-41 for status.

```
*LBL "SENDA"
 0 PT=
 ASIZE?                       Get number of chars in ALPHA.
 X-BUF                        Put count in buffer (AIPT mode).
 A-BUF                        Put ALPHA in buffer.
 0 ENTER^ 8 WREG              Set SSRQ.
 END
```

These two programs are the controller half of the DEVICE mode pair. The first part, READ, initializes for INTR. The second part, INTR, reads the buffer from the device half.

```
*LBL "READ"
 25 BSIZEX AIPT             Create the buffer.
 CF 28 CF 29 FIX 0          Get rid of radix indicators.
 3 ENTER^ 64 WREG           Set AIDY.
 0 ENTER^ 1 WREG            Set MCL - clear HP-IL IC.
 CLX 192 WREG               Set SC, CA.  Clear TA, LA ...
 IFC                        Clear device's interface.
 0 RREG 2 OR WREG           Set CLIFCR.
 AAU                        Unaddress devices.
 1 AAD                      Address devices.
 SF 18                      Enable INTR.
 END

*LBL "INTR"
 FRNS? GTO 01               Go ignore non-SRQRs.
 1 TAD                      Assume that the first device is HP-41.
 SST                        Get its status.
 128 AND                    Did it request service?
 X=0? GTO 08                No - untalk and exit.
 0 PT=                      Point to the beginning.
 BSIZE?                     Don't read more than will fit.
 INBUFX                     Read the buffer from the HP-41.
 0 PT=                      Point to the beginning again.
 CLA                        Prepare to bring in ALPHA.
 BUF-XB                     Get the length of new ALPHA.
 BUF-AX                     Get that many characters.
 AVIEW                      Show him what we got.
*LBL 03
 UNT                        Tell the HP-41 to stop sending.
 IDY                        See if we still have an SRQR.
 SRQR? GTO 02               If we do, tell him and exit.
 RTN
*LBL 02                     Tell him that we got two SRQs
 "SRQR TRUE"                 in a row.
 AVIEW
 RTN
*LBL 01                     Get rid of FRAV? or FRNS?.
 RFRM
 END
```

Section 3

Reference Section

How to Read This Section

This section is split into sub-sections containing logically grouped functions. Some of the sub-sections contain a short introductory paragraph at the beginning containing information required for understanding the functions described in that section.

To properly use the message or integrated circuit level functions, you need to understand how the HP-41, the HP-IL module, and the development module interact. These interactions are described in the paragraphs below.

The HP-41 continually accesses the HP-IL module unless flag 33 is set. (Refer to [SF33] and [CF33] on page 26). The HP-41 tries to print between every function; therefore, the HP-IL module constantly looks for a printer on the loop unless flag 33 is set. Searching for a printer can change the currently addressed device; so flag 33 must be set to maintain an active listener or active talker. Flag 33 must also be set whenever functions from sub-sections G through P are used.

Additionally, the HP-IL integrated circuit is reset whenever the HP-41 goes into idle mode unless the AIDY bit in the HP-IL integrated circuit is set. This bit imposes constraints upon the operation of the loop. To avoid these constraints various functions, such as [SCOPE] and [MONITOR], keep the HP-41 in execution mode. If you wish to use [RREG] and [WREG], you must keep the HP-41 in execution mode, or you must obey the constraints upon the control bits detailed in Appendix C. [ON] will keep the HP-41 in execution mode.

Any time a function expects a number which it will convert to an integer, the sign of that number is ignored. The lone exceptions are [Y-AX] and [A-XX]. Some of these functions will generate "NONEXISTENT" error if the number is greater than 999. Exceptions to this are: buffer input functions, buffer output functions, buffer comparison functions, boolean functions, non-decimal input and output, [OUTBIN], and the X-register argument to [OUTBINY].

The buffer is a collection of registers set aside by using

[BSIZEX]. To conserve memory the buffer will be deleted the first time the HP-41 is turned on without the development module in place. The buffer will also be deleted by a MEMORY LOST condition. The number of registers used by the buffer is determined by the argument to [BSIZEX] divided by seven. The buffer will never use more registers than you give it.

The buffer has an internal pointer whose value can range from zero to the number of bytes in the buffer minus one. Many of the buffer operations start at the buffer pointer.

A. Error Handling

The following list describes the error codes that are placed into the X-register upon the occurrance of an error condition.

Code	Error	Cause
-1	TIME OUT	No message received within ten seconds.
-2	FRNS ERR	A message was received not as sent.
-3	ETO ERR	Message following data was not an ETO.
-4	NO RESPONSE	No response to SST, SAI, SDI, or TCT.
-5	ORAV = 0	ORAV did not go true within ten seconds.

Note that when an "ETO ERR" is given the module is reporting an error in protocol. It is not a violation of protocol for a listener to receive a non-ETO message immediately after a string of DABs. However, when that listener is also the controller it is an error. Remember that ETO is a message from the talker to the controller.

The development module handles some of its errors differently from the way the HP-41 handles its errors. Flag 25 is still used to trap errors, but the flag is not cleared by an error. Instead, an error code is returned to the X-register. The following table describes the effect of an error.

	Flag 25 set	Flag 25 clear
Keyboard Execution	X=negative error code Display error message Flag 25 not cleared	X=negative error code Display error message
Program or SST Execution	X=negative error code Flag 25 not cleared Execution continues	X=negative error code Display error message Execution stops

B. Utility Functions

Function: [SCOPE] - Displays HP-IL messages.

Description: Displays HP-IL message mnemonics and stores them in the buffer if store mode is selected. Messages can be delayed from 0 to 1.5 seconds. The delay for IDY messages is always zero.

Input: When executed from a program, the X-register is the number of messages to be received before exiting. When executed from the keyboard, X is not used.

The keyboard is redefined to:

[STO]: Used to toggle in and out of store mode. When in store mode, every message received is stored in the buffer according to the scope pointer. Upon executing [SCOPE], the scope pointer is initialized to the same value as the buffer pointer. If a message is received which would overflow the buffer, it is not stored or echoed. The message "END OF BUF" is displayed and transmission will halt until you either hit [STO], exiting store mode, or hit [BACKARROW], setting the scope pointer to the buffer pointer.

[Backarrow]: Set the scope pointer to the buffer pointer.

[SST]: Moves the scope pointer to the next message. If the shift annunciator is lit, the pointer moves backwards. The pointer will not go past either end of the buffer.

[SHIFT]: Toggles the shift annunciator. Shift is not cleared by other operations.

[0], [1], [2], [3]: The message delay is set to 0, .5, 1.0, or 1.5 seconds, depending upon whether [0], [1], [2] or [3] is pressed. Messages are placed in the display even with zero delay, to show that a transmission has occurred. If [3] has been pressed and a plug-in printer is present and the mode switch is not set to MAN, then succeeding messages will be printed.

[Off]: Exits scope mode and turns the calculator off.

[R/S]: Exits scope mode.

Output: The display can contain message mnemonics from either the loop or the buffer. If the mnemonic is preceded by a number, the display contains a message taken from the buffer. If the mnemonic stands alone in the display, the display contains the latest message received from HP-IL.

Warnings: On entry, TA and LA are set to one. On exit, TA and LA are set to zero. If ten minutes elapses without a key being pressed or a message being received, scope mode is exited. If the buffer pointer is in auto advance mode, the buffer pointer will be set to the value of the scope pointer on exit. Each message occupies two bytes in the buffer.

[SCOPE] will cause errors when used in a loop with a Series 80 controller with delays other than zero, because of the way the controller tests for loop continuity. The controller sends AAU messages and expects to get them back within a short time. If this does not happen then it will send out another one, which causes messages to back up and errors to occur.

To avoid problems with Series 80 controllers, use a delay of zero and store the messages in the buffer.

Function: [MONITOR] - Displays HP-IL messages manually.

Description: Displays HP-IL messages, but does not retransmit them. Monitor mode is useful for manually sending and receiving messages. When a message is received, its mnemonic is placed in the display and its value is placed in the X- and Y- registers identically to [RFRM]. The message may be immediately retransmitted using [WFRM], it may be modified before retransmission, or a completely different message may be sent out. [MONITOR] does not let the HP-41 go to idle mode, and it sets register five of the HP-IL integrated circuit to all ones.

Warnings: Monitor mode is exited after ten minutes of no activity, when register five is cleared using [WREG], or whenever the HP-41 is turned off.

Function: [SF33] - Sets user flag 33.

Description: When user flag 33 is set, the HP 82160A HP-IL module functions will not and can not access the loop. Setting this flag will prevent the HP-IL module from searching the loop for a printer. While flag 33 is set, executing any HP-IL module function will result in "TRANSMIT ERR" being displayed. [SF33] exists because user flags 31 through 35 cannot be set or cleared using [SF] or [CF]. These flags are not cleared at turn on. Flags 31 through 35 can be cleared only by "MEMORY LOST" or by some special function such as [CF33].

Output: Flag 33 is set.

Warnings: [SF] with an argument of 33 can be typed into a program without causing an error. When that program is executed, [SF] will generate the "NONEXISTENT" error, so be sure to use [SF33].

Function: [CF33] - Clears user flag 33

Description: Flag 33 is cleared. The HP-41 HP-IL module functions will not work with flag 33 set.

Warnings: [CF] with an argument of 33 can be typed into a program without causing an error. When that program is executed, [CF] will generate the "NONEXISTENT" error, so be sure to use [CF33].

Function: [ON] - Keeps the HP-41 in execution mode.

Description: This is actually a built-in function which prevents the HP-41 from turning itself off after ten minutes of no activity. [ON] also sets flag 44. When flag 44 is set, the development module will keep the HP-41 in execution mode. This prevents the HP-IL integrated circuit from being reset. The only other way to prevent the integrated circuit from being reset is to set the AIDY bit, which then puts constraints on the CA bit. For further details, refer to Appendix C.

Function: [X<>FLAG] - Exchange user flags 0 through 7 with X.

Description: Takes the integer part of the X-register and exchanges it with user flags 0 through 7. User flag 0 is weighted 1, flag 2 is weighted 2, flag 3 is weighted 4, etc. [X<>FLAG] is the same as HP 82180A Extended Functions [X<>F].

Value of X	128	64	32	16	8	4	2	1
User Flags	7	6	5	4	3	2	1	0

Input: X, an integer number from 0 through 255, and user flags 0 through 7.

Output: X, an integer from 0 through 255, and user flags 0 through 7.

Warnings: Any number from 0 through 999 will be converted to an integer; only the low eight bits will be used.

Function: [ROMCHKX] - Performs checksum of the Xth ROM.

Description: Verifies the checksum of a plug-in ROM using the X-register as the ROM ID. The ROM ID is determined from the first number of an XROM pair. Some modules contain two ROMs and hence have two ROM IDs. While computing the ROM checksum, the display will show "DD CC-NN TST" where DD is the ROM ID, and CC-NN is the ROM label. When the checksum computation is finished, the display will change to "DD CC-NN OK" or "DD CC-NN BAD". If [ROMCHKX] was executed from program execution, it will skip the next step if the checksum was bad. If the specified ROM ID is not present, the message "NO ROM DD" will be placed in the display and execution will continue after skipping the next step.

Input: The X-register contains the ROM ID.

Output: A message is left in the display as detailed above.

C. Buffer Utility Functions

Function: [BSIZEX] - Initializes a buffer with X bytes.

Description: Creates a buffer for use by the development module. The X-register contains the buffer size in bytes. If X is 0 then no buffer is created, and any existing one is deleted. The maximum size of the buffer is 1771 bytes. The buffer size will be rounded up to the next larger increment of seven bytes if X is not a multiple of seven. If the HP-41 does not have enough memory for the buffer, it will pack the memory and ask you to try again.

Input: The X-register contains the number of bytes to use for the buffer.

Output: A buffer is created of size 7*INT(X/7+1) bytes.

Function: [BSIZE?] - Returns the buffer size to X.

Description: The number of bytes in the buffer is pushed on the stack. This number will always be a multiple of seven.

Output: The X-register contains the maximum addressable byte plus one; the stack is lifted.

Function: [PT=] - Sets the buffer pointer equal to X.

Description: The absolute value of the X-register contains the pointer value. The buffer pointer may range from 0 to the value returned by [BSIZE?] minus one. The "END OF BUF" error is generated if X is larger than the buffer size less one.

Input: The buffer pointer gets X.

Function: [PT?] - Returns the buffer pointer to X.

Description: The value of the buffer pointer is pushed on the stack. This number can range from zero to the value returned by [BSIZE?]. The only way to get [PT?] to have the value of [BSIZE?] is if a function reached the end of the buffer in automatic increment mode.

Input: The buffer pointer.

Output: The X-register contains the buffer pointer; the stack is lifted.

Function: [AIPT] - Sets auto increment of pointer mode.

Description: Auto increment the buffer pointer after each operation. Every function which operates on the buffer will increment the buffer pointer by the appropriate amount after [AIPT] is executed. [AIPT] is the inverse of [MIPT].

Function: [MIPT] - Sets manual increment of pointer mode.

Description: Manual increment buffer pointer. While in manual increment mode, the buffer pointer will stay where it was after each buffer operation. The pointer will only be moved by [PT=]. When the buffer is first set up, it is set to be in manual advance mode. [MIPT] is the inverse of [AIPT].

Function: [PRBYTES] - Prints bytes from the buffer.

Description: Prints the bytes in the buffer in hexadecimal format. The bytes are printed starting from the buffer pointer and ending at the end of the buffer. If the HP 82143A printer is plugged in, it will be used for printing. If not, the HP-IL printer will be used only if flag 33 is clear and there is no other controller on the loop. The mode switch on either printer must be set to trace or norm, or flag 15 or flag 16 must be set for HP-IL printers other than the HP 82162A. If neither printer is present, the bytes will be displayed at about two bytes per second. Pressing [R/S] will exit [PRBYTES]. Pressing any other key will slow the display rate to about one byte per second.

Function: [PRFRMS] - Prints messages from the buffer.

Description: Identical to [PRBYTES] except that message mnemonics are printed instead of bytes. Each message occupies two bytes in the buffer.

Warnings: The buffer pointer may be pointing to the second byte of the message. This will cause garbage to be displayed in place of the correct mnemonics. The buffer pointer should always be the same as that used to store the messages.

D. Buffer Input

Function: [A-BUF] - Stores the ALPHA register to the buffer.

Description: The number of bytes stored is the same as the number returned by [ASIZE?]. Leading nulls in the ALPHA register are ignored.

Input: ALPHA register.

Output: Buffer, starting at the buffer pointer.

Function: [X-BUF] - Store X to buffer in integer form.

Description: The number of bytes stored is determined by the minimum number of bytes needed to represent the X-register in integer form. If X contains a string, then leading nulls are ignored, one character to a byte.

Input: The X-register

Output: Buffer, starting at the buffer pointer, contains the contents of the X-register.

Warnings: For an arbitrary integer, you don't know how many bytes it will take to store. 0 through 255 will take one byte, and 256 through 65535 will take two, etc.

Function: [RG-BUFX] - Copy registers to buffer using X.

Description: Copies registers to buffer using the X-register as a register index in the form bbb.eee, where bbb is the starting register number and eee is the ending register number. The copy starts at the buffer pointer. If the buffer overflows, the "END OF BUF" error is given. If any of the registers do not exist, the "NONEXISTENT" error is displayed. Each register is seven bytes long. The transfer is done on a byte by byte basis; no translation is performed.

Input: Registers bbb through eee.

Output: Buffer, starting at the buffer pointer, contains the specified registers.

Warnings: Do not [RCL] or [VIEW] any of these registers, because these functions normalize the number and store it back in the register. If the data that was in the register doesn't look like a normalized number, it will be changed.

Function: [INBUFX] - Input data messages into buffer using X.

Description: Inputs data messages into buffer using the X-register as the count of bytes to input. If CA is set, then this function sets LA and sends SDA before reading. If CA is not set, then LA is not set and an SDA is not sent. X bytes are input from HP-IL and stored into the buffer starting at the current pointer. The transfer is terminated by one of the following conditions:

1) ETO. The ETO is not retransmitted.
2) Any non-data message. This generates an "ETO ERROR".
3) X bytes have been received; the NRD handshake is performed if CA=1. If CA=0 then the last frame is held. You have to pull the last frame out of the buffer and echo it when you are finished.
4) Reaching the end of the buffer. The "END OF BUF" error is generated.
5) Sixty seconds after the most recent message is received; the "TIME OUT" error is generated.
6) Pressing any key generates the "TIME OUT" error.

Input: X is the count of bytes to transfer. The bytes are transmitted by the active talker. An active talker is assumed.

Output: Starting at the buffer pointer, the buffer contains the received data.

E. Buffer Output

Function: [BUF-XA] - Convert a string to a number in X.

Description: Converts a string representation of a number into the X-register. Starting at the buffer pointer, X bytes are taken as the ASCII representation of a number, whose value is returned to X. Characters which do not fall into the set of the digits zero through nine, the radix indicator '.', the exponent indicator 'E', and the Line Feed character are ignored. The maximum number of characters used is given in X. Fewer characters will be used if a Line Feed is encountered.

Input: X contains the number of characters to look at. The buffer contains a string representation of a number, starting at the buffer pointer.

Output: X contains the value of the numeric string taken from the buffer starting at the buffer pointer. If Line Feed is encountered, the buffer pointer is left pointing at the character after the Line Feed, otherwise the buffer pointer is incremented by the value of X on entry. Remember that the buffer pointer is changed only in auto increment mode.

Function: [BUF-XB] - Convert bytes in buffer to an integer in X.

Description: The X-register is the count of bytes to convert. The first byte taken from the buffer is the most significant.

Input: X contains the count of bytes to convert from the buffer.

Output: X is an integer. Range is dependent upon the number of bytes converted.

Function: [BUF-AX] - Place X bytes into ALPHA from the buffer.

Description: Loads the ALPHA register from the buffer using the

X-register as a count of bytes. None of the bytes are special; Carriage Return will not terminate the transfer, and nulls are not ignored. The ALPHA register is not cleared before execution.

Input: X contains the count of bytes to transfer from the buffer.

Output: ALPHA register contains the bytes transferred from the buffer.

Function: [BUF-RGX] - Copy buffer bytes to registers.

Description: Copies bytes from the buffer to registers. The X-register contains a register index in the form bbb.eee, where bbb is the starting register number and eee is the ending register number. The copy starts at the buffer pointer. If the buffer is overflowed, "END OF BUF" is displayed. If any of the registers do not exist, the "NONEXISTENT" error is given. Each register is seven bytes long. The transfer is done on a byte by byte basis. No translation is performed, and nulls are not ignored.

Input: Buffer contents, and the X-register contains the register index.

Output: Registers bbb through eee contain the bytes from the buffer.

Warnings: Do not [RCL] or [VIEW] any of these registers, because these functions normalize the number and store it back in the register. If the data that was in the register does't look like a normalized number, it will be changed.

Function: [OUTBUFX] - Output data messages from buffer using X.

Description: Outputs data messages from buffer using the X-register as a count of bytes to be transferred. If CA set, sets TA before execution. If CA is not set then CA is not set. The bytes are taken from the buffer starting at the buffer pointer. If an NRD is received from the listener then the transfer will be halted and the NRD handshake performed. An ETO will be sent after the desired number of bytes has been sent.

Input: The X-register contains the count of bytes.

Output: The data messages are sent to the active listeners on HP-IL.

Warnings: Does not wait for an SDA.

F. Buffer Comparisons

All of these comparison functions act just like the comparison functions in the HP-41. If the question being asked is true, the display will contain "YES", or execution will continue with the next step. If the question is false, the display will contain "NO", or execution will continue after skipping a step.

Function: [X=BUF?] - Compare X to bytes in buffer.

Description: Compares the X-register to the buffer. If X contains a number, the integer part is used. The number of bytes needed to represent X will be compared to a like number of bytes in the buffer starting at the pointer. If X contains a string, leading nulls will be ignored.

Input: The X-register is a number or string.

Output: Yes or No.

Function: [A=BUFX?] - Compare X bytes from ALPHA to the buffer.

Description: Compares X bytes of the ALPHA register to the buffer starting at the buffer pointer. If the X-register is greater than the length of the ALPHA register, then [A=BUFX?] is identical to [A=BUF?]. Null bytes are ignored.

Input: The X-register contains the count of bytes, ALPHA register contains the data to be compared.

Output: Yes or No.

Function: [A=BUF?] - Compare ALPHA register to the buffer.

Description: Compares the contents of the ALPHA register to the contents of the buffer starting at the buffer pointer. The number of bytes compared is equal to the number of characters in the ALPHA register. Leading null bytes in the ALPHA register are ignored. If the ALPHA register is empty, the test returns "NO".

Input: The ALPHA register contains a string.

Output: Yes or No.

Function: [RG=BUF?] - Compare registers to the buffer using X.

Description: The contents of the X-register is taken as a register index in the form bbb.eee. The block of registers will be compared with the same number of bytes in the buffer starting from the buffer pointer. Each register contains seven bytes. The comparison is done on a byte by byte basis. No conversion is performed.

Input: X contains the register index; the buffer.

Output: Yes or No.

Warnings: Do not [RCL] or [VIEW] any of these registers, because these functions normalize the number and store it back in the register. If the data that was in the register does't look like a normalized number, it will be changed.

G. ALPHA Register Functions

Function: [ASIZE?] - X gets the number of characters in ALPHA.

Description: Returns the number of characters contained in the ALPHA register. Leading nulls are ignored.

Input: ALPHA register.

Output: The X-register contains the number of characters in the ALPHA register; the stack is lifted.

Function: [A-XL] - Removes the leftmost ALPHA and puts it in X.

Description: The decimal value of the leftmost character in the ALPHA register is placed in the X-register. The character is removed from the ALPHA register. If the leftmost character was followed by any nulls, those nulls will be lost.

Input: The leftmost character of the ALPHA register.

Output: The X-register contains the ASCII equivalent of the character.

Function: [X-AL] - ASCII number in X is put to left of ALPHA.

Description: The ASCII character equivalent to the integer in the X-register is appended to the left of the ALPHA register. If there are already 24 characters in the ALPHA register, the rightmost character will be lost and the ALPHA register will be shifted right once.

Input: X contains the decimal value of ASCII equivalent of the character.

Output: Leftmost character of the ALPHA register.

Function: [A-XR] - Removes the rightmost ALPHA and puts it in X.

Description: The decimal value of the rightmost ASCII character in the ALPHA register is placed in the X-register. The character is deleted from the ALPHA register.

Input: The rightmost character of the ALPHA register.

Output: The X-register contains the decimal equivalent of the ASCII character.

Function: [X-AR] - ASCII number in X is put to right of ALPHA

Description: The ASCII character specified by the integer in the X-register is appended to the right of the ALPHA register. If there are already 24 characters in the ALPHA register, the leftmost character will be lost and the ALPHA register shifted left once.

Input: The X-register contains the decimal equivalent of the ASCII character.

Output: Rightmost character of the ALPHA register.

Function: [A-XX] - The Xth ALPHA character's value is placed in X.

Description: The decimal value of the Xth ASCII character in the ALPHA register is placed in the X-register. The character in the ALPHA register is left untouched. The original position in X is saved in the LASTX-register.

The usage of the X-register depends upon the sign of X. If X is positive, X is a count of characters from the left end of ALPHA, starting from zero and ignoring nulls. If X is negative, X is a count of bytes from the right end of ALPHA, starting from one and using absolute position. The X-register is interpreted as follows, given that N is the number of characters in ALPHA:

X >= N or X > 24	not valid - "DATA ERROR"
0 <= X < N	X counted from left end of string
X = 0	leftmost character in string
-24 <= X < 0	N counted from right end of ALPHA
X < -24	not valid - "DATA ERROR"

Input: The Xth ALPHA register character.

Output: X contains the decimal equivalent of the ASCII character; the stack is lifted. The LASTX register gets the old X-register.

Function: [Y-AX] - ASCII number in Y is placed in ALPHA at X.

Description: The ASCII character specified by the integer in the Y-register is placed into the ALPHA register at the position given by the X-register. The new character replaces the old one at the same position. Refer to the table under [A-XX] for the way this function will interpret the value specified in the X-register.

Input: A position in the X-register, the ASCII character's decimal value in Y.

Output: A character placed in Alpha.

H. Stack Input and Output

The next set of functions all use the same output routine. This routine waits for ORAV to become true within ten seconds. If this does not happen, the "ORAV=0" error is generated. Otherwise, the message is sent and the routine waits for it to return. If it does not return within ten seconds, the "TIME OUT" error is generated.

Function: [OUTBIN] - Output bytes from X.

Description: Outputs bytes from the X-register. The number of bytes sent is the minimum number of bytes needed to represent X. If X contains a string, leading nulls will be ignored. TA is set before execution. An ETO is not sent after execution. At least one and no more than seven data bytes will be output by [OUTBIN].

Input: The integer part of X is output.

Output: Data bytes to the active listeners on HP-IL.

Function: [OUTBINY] - Output bytes from X, using Y as the number of bytes to send.

Description: Same as [OUTBIN] except that the number of bytes transmitted is determined by the Y-register. No matter what the value of Y, at least one and no more than seven bytes will be output by [OUTBINY].

Input: The integer part of the X-register is output, using Y as a byte count.

Output: Data bytes to the active listeners on HP-IL.

Function: [INBIN] - Input bytes to X.

Description: Inputs bytes to the X-register. If CA set, sets LA and sends SDA before execution. If CA clear, doesn't set LA and doesn't send SDA. [INBIN] will read data messages until a non-data message is received. The seven most recently received data bytes are treated as a seven byte long integer which will be placed in X. If the non-data message was not an ETO, an "ETO ERROR" will be generated. If no messages are received within ten seconds, a "TIME OUT" error will be generated. If an error was generated, X will contain the error number and Y will contain the integer.

Input: The active talker on HP-IL.

Output: An integer in X, or an integer in Y and an error code in X.

I. Sending Command Messages

These functions use the same output routine as that in section H. All of these functions will set CA before execution, as it is a violation of protocol for a non-controller to send any of these messages.

Function: [AAU] - Sends AAU.

Description: Sends the Auto Address Unconfigure command.

Function: [CMD] - Sends arbitrary CMD from X.

Description: Sends an arbitrary command message. The eight bits, D7 through D0, of the message are taken from the X-register.

Input: X contains an integer from zero to 255.

Function: [DDL] - Sends the DDL message specified in X.

Description: Sends a Device Dependent Listener command. The X-register specifies which DDL message will be sent. If X is greater than 31 then the "ADR ERR" error will be generated.

Input: X contains the DDL command number.

Function: [DDT] - Sends the DDT message specified in X.

Description: Sends a Device Dependent Talker command. The X-register specifies which DDT message will be sent. If X is greater than 31 then the "ADR ERR" error will be generated.

Input: X contains the DDT command number.

Function: [GET] - Sends GET.

Description: Sends the Group Execute Trigger command.

Function: [GTL] - Sends GTL.

Description: Sends the Go To Local command.

Function: [IFC] - Sends IFC.

Description: Sends the Interface Clear command. Only the system controller can source IFC, so SC will be set before execution.

Function: [LAD] - Sends the LAD message specified in X.

Description: Sends a Listener Address command. The X-register specifies which LAD message will be sent. If X is greater than 31 then the "ADR ERR" error will be generated.

Input: X contains the LAD command number.

Function: [LPD] - Sends LPD.

Description: Sends the Loop Power Down command.

Function: [NRE] - Sends NRE.

Description: Sends the Not Remote Enable command.

Function: [REN] - Sends REN.

Description: Sends the Remote Enable command.

Function: [SDC] - Sends SDC.

Description: Sends the Selected Device Clear command.

Function: [TAD] - Sends the TAD message specified in X.

Description: Sends a Talker Address command. The X-register specifies which TAD message will be sent. If X is greater than 31 then the "ADR ERR" error will be generated.

Input: X contains the TAD command number.

Function: [UNL] - Sends UNL.

Description: Sends the Unlisten command, which is the same as a LAD 31.

Function: [UNT] - Sends UNT.

Description: Sends the Untalk command, which is the same as a TAD 31.

J. Sending Ready Messages

Sending ready messages is performed in the same manner as sending command messages. In certain cases the time out and error checking are not performed because a message is not returned. Those functions which do not expect a message to return will explicitly state this.

Function: [AAD] - Sends the AAD message specified in X.

Description: Sends a Auto Address message. The X-register specifies which AAD message will be sent. If X is greater than 31 then the "ADR ERR" error will be generated. The value of the AAD which returns is pushed on the stack.

Input: X contains the AAD message number.

Output: X contains the address of the highest addressable device plus one.

Warnings: No check is made to ensure that the returned message is actually an AAD. The lower five bits of whatever message is received are returned in the X-register.

Function: [SDA] - Sends SDA.

Description: Sends the Send Data ready message. Does not wait for a message to return. CA is set before execution.

Function: [SAI] - Sends SAI, returns ID to X.

Description: Sends the Send Accessory ID message. Sets CA and LA before execution. The sequence of bytes (assuming that more than one returns) is treated as an integer, MSB (most significant byte) first. This integer is pushed on the stack. If the last non-data message is not an ETO, an "ETO ERR" error will be generated. If the SAI is retransmitted by the device, the "NO RESPONSE" error will be generated. If no message returns within ten seconds, the

"TIME OUT" error will be generated.

Output: X contains an Accessory ID, or Y contains an Accessory ID and X contains an error number.

Function: [SST] - Sends SST, returns status to X.

Description: Sends the Send Status message. Sets CA and LA before execution. The sequence of bytes (assuming that more than one returns) is treated as an integer, MSB (most significant byte) first. This integer is pushed on the stack. If the last non-data message is not an ETO, an "ETO ERR" error will be generated. If the SST is retransmitted by the device, the "NO RESPONSE" error will be generated. If no message returns within ten seconds, the "TIME OUT" error will be generated.

Output: X contains the status, or Y contains the status and X contains an error number.

Function: [SDI] - Sends SDI, returns data to ALPHA.

Description: Sends the Send Device ID message. Sets CA and LA before execution. The data bytes that return are placed in the ALPHA register. CR and LF are ignored. If the last non-data message is not an ETO, an "ETO ERR" error will be generated. If the SDI is retransmitted by the device, the "NO RESPONSE" error will be generated. If no message returns within ten seconds, the "TIME OUT" error will be generated.

Output: The ALPHA register contains a string of characters representing the device ID. X may contain an error number.

Function: [TCT] - Sends TCT, waits for incoming message.

Description: Sends the Take Control ready message. [TCT] then waits for a message to return. If the active talker cannot take control, it will return the TCT. If it can take control, it will start sending commands. If the message which returns is a command message, it is put into the X- and Y-registers by [RFRM]. If the TCT returns, or no message returns within ten seconds, the "NO RESPONSE" error is generated.

Warnings: The test for TCT returned is satisfied by any ready message.

Function: [NRD] - Performs NRD handshake on current data message.

Description: Performs the NRD handshake. The message which is currently sitting in the HP-IL IC registers one and two is read and saved. CA is set. The Not Ready for Data (NRD) ready message is sent on the loop. The HP-41 waits for the NRD message to return. If anything else returns, a "FRNS ERR" error is generated. The saved data message is sent out on the loop. The HP-41 waits for an ETO to return. If anything else returns, a "ETO ERR" is generated.

K. Sending Identify Messages.

Function: [IDY] - Sends an IDY message, returns data bits to X.

Description: Sends out an Identify message with the data bits set to zero. CA is set before execution. When the message returns, the eight data bits are pushed on the stack. If the message does not return within ten seconds, a "TIME OUT" error is generated and -1 is pushed on the stack.

Input: none (X is not used).

Output: X contains the parallel poll bits, or X contains an error number.

L. Sending Arbitrary Messages.

Function: [RFRM] - Reads the already present message into X,Y.

Description: Reads registers one and two of the HP-IL integrated circuit and puts them on the stack where the X-register contains the three control bits and Y-register contains the eight data bits of the message. [RFRM] does not wait for FRAV or FRNS before reading the message.

Input: Registers one and two of the HP-IL integrated circuit.

Output: X and Y contain the message's value.

Function: [WFRM] - Writes a message using X,Y; waits for ORAV.

Description: The X-register contains an integer giving the three control bits of a message, and Y-register contains the eight data bits. ORAV and FRNS are tested before writing the message. If FRNS is set, the "FRNS ERR" error message is generated. If ORAV is not set within ten seconds, the "ORAV = 0" error message will be generated. [WFRM] exits after writing the message and does not wait for its return.

Input: X and Y are integers representing the message.

Output: Registers one and two of the HP-IL integrated circuit.

Function: [IFCR?] - Tests for the IFCR bit true.

Description: [IFCR?] tests bit 4 of register 1 in the HP-IL integrated circuit. If executed from the keyboard, it will display "YES" or "NO". If executed from a program, it will skip a step if the answer is no.

Function: [SRQR?] - Tests for the SRQR bit true.

Description: [SRQR?] tests bit 3 of register 1 in the HP-IL integrated circuit. If executed from the keyboard, it will display "YES" or "NO". If executed from a program, it will skip a step if the answer is no.

Function: [FRAV?] - Tests for the FRAV bit true.

Description: [FRAV?] tests bit 2 of register 1 in the HP-IL integrated circuit. If executed from the keyboard, it will display "YES" or "NO". If executed from a program, it will skip a step if the answer is no.

Function: [FRNS?] - Tests for the FRNS bit true.

Description: [FRNS?] tests bit 1 of register 1 in the HP-IL integrated circuit. If executed from the keyboard, it will display "YES" or "NO". If executed from a program, it will skip a step if the answer is no.

Function: [ORAV?] - Tests for the ORAV bit true.

Description: [ORAV?] tests bit 0 of register 1 in the HP-IL integrated circuit. If executed from the keyboard, it will display "YES" or "NO". If executed from a program, it will skip a step if the answer is no.

M. Boolean Functions

The following functions operate upon thirty-two bit unsigned integers. If an argument to the function requires greater than thirty-two bits to be represented as an integer, the error "OUT OF RANGE" is given. Note that this error does not return an error number, and behaves exactly like an HP-41 error.

Function: [AND] - ANDs the X- and Y-registers and returns to X.

Description: Performs a boolean AND between the X-register and the Y-register. Returns the result to the X-register with stack lift enabled. The operands are dropped from the stack. Saves the X argument in LASTX.

Input: X and Y are integers.

Output: X is an integer.

Function: [OR] - ORs the X- and Y-registers and returns to X.

Description: Performs a boolean OR between the X-register and the Y-register. Returns the result to the X-register with stack lift enabled. The operands are dropped from the stack. Saves the X argument in LASTX.

Input: X and Y are integers.

Output: X is an integer.

Function: [XOR] - exclusive ORs X and Y and returns to X.

Description: Performs a boolean exclusive OR between the X-register and the Y-register. Returns the result to the X-register with stack lift enabled. The operands are dropped from the stack. Saves the X argument in LASTX.

Input: X and Y are integers.

Output: X is an integer.

Function: [NOT] - X gets the one's complement of X.

Description: The X-register gets the one's complement of itself. The stack lift is enabled, and the original X is removed from the stack and placed in LASTX.

Input: X is an integer.

Output: X is an integer.

Function: [ROTXY] - Rotates Y to the right by X bits.

Description: The Y-register is rotated to the right by X bits. The bits which fall off the right end reappear at the left end of Y. For example, 1 rotated right by 1 will become 80000000 hexadecimal. The operands are not removed, but are pushed onto the stack. Does not set LastX. The sign of X is ignored. Rotating 32-X bits effectively rotates to the left by X.

Input: X is an integer count, Y is an integer.

Output: Y is pushed to Z, X is pushed to Y, and X contains the rotated value from Y.

Function: [BIT?] - Tests to see if the Xth bit of Y is set.

Description: [BIT?] works like all the rest of the HP-41 test instructions. The test is for the Xth bit of the number in the Y-register to see if it is set. If [BIT?] is being executed from the keyboard it will return "YES" or "NO". If executed from a program, it will skip a step if the specified bit is set.

Input: X is a bit position number, Y is an integer.

Output: Yes or No.

N. Non-Decimal Input and Output

Function: [BININ] - Inputs a number in binary.

Description: The keyboard is redefined so that the only digit keys which are active are [0] and [1]. Any other key terminates numeric entry and is executed.

Input: [0] and [1].

Output: An integer in the X-register.

Warnings: Because of the size of the display, the largest number enterable is ten digits (10 bits). If you press a terminating key too quickly, the key may be lost after [BININ] terminates.

Function: [OCTIN] - Inputs a number in octal.

Description: The keyboard is redefined so that the only digit keys which are active are the keys [0] through [7]. Any other key terminates numeric entry and is executed.

Input: [0] through [7].

Output: An integer in the X-register.

Warnings: Because of the size of the display, the largest number enterable is ten digits (30 bits). If you press a terminating key too quickly, the key may be lost after [OCTIN] terminates.

Function: [HEXIN] - Inputs a number in hexadecimal.

Description: The keyboard is redefined so that all the digit keys and the letters [A] through [F] are active. Any other key terminates numeric entry and is executed.

Input: [0] through [9], [A] through [F].

Output: An integer in the X-register.

Warnings: If you press a terminating key too quickly, the key may be lost after [HEXIN] terminates.

Function: [BINVIEW] - Shows the value of X in binary.

Description: The display shows the value of the X-register in binary. If X is longer than ten bits, then the "OUT OF RANGE" error is given. The distance between the commas in the display is given by the current FIX value. The normal display of the X-register is restored using [backarrow].

Input: X is an integer.

Output: The display shows X in binary.

Function: [OCTVIEW] - Shows the value of X in octal.

Description: The display shows the value of the X-register in octal. If X is longer than thirty bits, then the "OUT OF RANGE" error is given. The distance between the commas in the display is given by the current FIX value. The normal display of the X-register is restored using [backarrow].

Input: X is an integer.

Output: The display shows X in octal.

Function: [HEXVIEW] - Shows the value of X in hexadecimal.

Description: The display shows the value of the X-register in hexadecimal. The distance between the commas in the display is given by the current FIX value. The normal display of the X-register is restored using [backarrow].

Input: X is an integer.

Output: The display shows X in hexadecimal.

O. Reading and Writing of the HP-IL IC Registers.

Function: [WREG] - Writes X to HP-IL register number in Y.

Description: Writes to a given register. Y-register contains the register number, the X-register contains the new contents of the register.

Input: Y is an integer from 0 to 7, X is an integer from 0 to 255.

Function: [RREG] - Reads HP-IL IC register specified by X.

Description: Read from a given register. The X-register contains the register number to read. The contents of the register are pushed onto the stack. After execution Y contains the register number and X contains the contents of the register.

Input: X is an integer from 0 to 7.

Output: X is an integer from 0 to 255, Y is the old X.

P. Receiving Messages in Idle Mode

The HP-IL integrated circuit has the ability to wake up the HP-41 from idle mode. This ability is detailed in Appendix C. If flag eighteen is set, then the receipt of a message that causes IFCR, SRQR, FRAV, or FRNS to be set will cause the execution of the program "INTR". The INTR program must cause the interrupting bit to be cleared, otherwise the subroutine will be reexecuted just as soon as it exits.

The IFCR bit is cleared by setting the CLIFCR bit, which will automatically clear itself also. The SRQR bit is cleared only upon receipt of a message with the SRQ bit false, such as a DAB, END, or IDY message. Both the FRAV and FRNS bits are cleared by reading register two using either [RFRM] or [RREG].

Since the HP-41 cannot be both editing and running a program, never enter program mode with flag 18 set. If you do, the first digit in the first digit string in your program will be inserted between every instruction step. The easiest way to exit this mode is to pull the HP-IL module out of the HP-41. To repair the damage, delete the extraneous digit strings.

Appendix A

Care, Warranty, and Service Information

Module Care

CAUTION
Always turn off the HP-41 before connecting or disconnecting any module or peripheral. Failure to do so could result in damage to the HP-41 or disruption of the system's operation.

- Keep the contact area of the module free of obstructions. Should the contacts become dirty, carefully brush or blow the dirt out of the contact area. Do not use any liquid to clear the contacts.

- Store the module in a clean, dry place.

- Always turn off the HP-41 before installing or removing any module or peripherals.

- Observe the following temperature specifications:

  Operating: 0 deg C to 45 deg C (32 deg F to 113 deg F)

  Storage: -40 deg C to 75 deg C (-40 deg F to 167 deg F)

Limited One-Year Warranty

What We Will Do

The HP 00041-15043 HP-IL Development Module is warranted by Hewlett-Packard against defects in materials and workmanship affecting electronic and mechanical performance, but not software content, for one year from the date of original purchase. If you sell your unit or give it as a gift, the warranty is transferred to the new owner and remains in effect for the original one-year period. During the warranty period, we will repair or, at our option, replace at no charge a product that proves to be defective, provided you return the product, shipping prepaid, to a Hewlett-Packard service center.

What Is Not Covered

This warranty does not apply if the product has been damaged by accident or misuse or as the result of service or modification by other than an authorized Hewlett-Packard service center.

No other express warranty is given. The repair or replacement of a product is your exclusive remedy. ANY OTHER IMPLIED WARRANTY OF MERCHANTABILITY OR FITNESS IS LIMITED TO THE ONE-YEAR DURATION OF THIS WRITTEN WARRANTY. Some states, provinces, or countries do not allow limitations on how long an implied warranty lasts, so the above limitation may not apply to you. IN NO EVENT SHALL HEWLETT-PACKARD COMPANY BE LIABLE FOR CONSEQUENTIAL DAMAGES. Some states, provinces, or countries do not allow the exclusion or limitation of incidental or consequential damages, so the above limitation or exclusion may not apply to you.

This warranty gives you specific legal rights, and you may also have other rights which vary from state to state, province to province, or country to country.

Warranty for Consumer Transactions in the United Kingdom

This warranty shall not apply to consumer transactions and shall not affect the statutory rights of a consumer. In relation to such transactions, the rights and obligations of Seller and Buyer shall be determined by statute.

Obligation to Make Changes

Products are sold on the basis of specifications applicable at the time of manufacture. Hewlett-Packard shall have no obligation to modify or update products once sold.

Warranty Information

If you have any questions concerning this warranty, please contact an authorized Hewlett-Packard dealer or a Hewlett-Packard sales and service office. Should you be unable to contact them, please contact:

- In the United States:

Hewlett-Packard
Corvallis Division
1000 N.E. Circle Blvd.
Corvallis, OR 97330
Telephone: (503) 758-1010
Toll-Free Number: (800) 547-3400 (except in Oregon, Hawaii, and Alaska)

- In Europe:

Hewlett-Packard S.A.
7, rue du Bois-du-lan
P.O. Box
CH-1217 Meyrin 2
Geneva
Switzerland
Telephone: (022) 83 81 11

Note: Do not send units to this address for repair.

- In other countries:

Hewlett-Packard Intercontinental
3495 Deer Creek Rd.
Palo Alto, California 94304
U.S.A.
Telephone: (415) 857-1501

Note: Do not send units to this address for repair.

Service

Hewlett-Packard maintains service centers in most major countries throughout the world. You may have your unit repaired at a Hewlett-Packard service center any time it needs service, whether the unit is under warranty or not. There is a charge for repairs after the one-year warranty period.

Hewlett-Packard products are normally repaired and reshipped within five (5) working days of receipt at any service center. This is an average time and could vary depending upon the time of year and the work load at the service center. The total time you are without your unit will depend largely on the shipping time.

Obtaining Repair Service in the United States

The Hewlett-Packard United States Service Center for battery-powered computational products is located in Corvallis, Oregon:

Hewlett-Packard Company
Corvallis Division Service Department
P.O. Box 999/1000 N.E. Circle Blvd.
Corvallis, Oregon 97339/97330, U.S.A.
Telephone: (503) 757-2000

Obtaining Repair Service in Europe

Service centers are maintained at the following locations. For countries not listed, contact the dealer where you purchased your unit.

AUSTRIA
HEWLETT-PACKARD Ges.m.b.H.
Kleinrechner-Service
Wagramerstrasse-Lieblgasse 1
A-1220 Wien (Vienna)
Telephone: (0222) 23 65 11

BELGIUM
HEWLETT-PACKARD BELGIUM SA/NV
Woluwedal 100
B-1200 Brussels
Telephone: (02) 762 32 00

DENMARK
HEWLETT-PACKARD A/S
Datavej 52
DK-3460 Birkerod (Copenhagen)
Telephone: (02) 81 66 40

EASTERN EUROPE
Refer to the address listed under Austria

FINLAND
HEWLETT-PACKARD OY
Revontulentie 7
SF-02100 Espoo 10 (Helsinki)
Telephone: (90) 455 02 11

FRANCE
HEWLETT-PACKARD FRANCE
Division Informatique Personnelle
S.A.V. Calculateurs de Poche
F-91947 Les Ulis Cedex
Telephone: (6) 907 78 25

GERMANY
HEWLETT-PACKARD GmbH
Kleinrechner-Service
Vertriebszentrale
Berner Strasse 117
Postfach 560 140
D-6000 Frankfurt 56
Telephone: (611) 50041

ITALY
HEWLETT-PACKARD ITALIANA S.P.A.
Casella postale 3645 (Milano)
Via G. Di Vittorio, 9
I-20063 Cernusco Sul Naviglio (Milan)
Telephone: (2) 90 36 91

NETHERLANDS
HEWLETT-PACKARD NEDERLAND B.V.
Van Heuven Goedhartlaan 121
N-1181 KK Amstelveen (Amsterdam)
P.O. Box 667
Telephone: (020) 472021

NORWAY
HEWLETT-PACKARD NORGE A/S
P.O. Box 34
Oesterndalen 18
N-1345 Oesteraas (Oslo)
Telephone: (2) 17 11 80

SPAIN
HEWLETT-PACKARD ESPANOLA S.A.
Calle Jerez 3
E-Madrid 16
Telephone: (1) 458 2600

SWEDEN
HEWLETT-PACKARD SVERIGE AB
Skalholtsgatan 9, Kista
Box 19
S-163 93 Spanga (Stockholm)
Telephone: (08) 750 20 00

SWITZERLAND
HEWLETT-PACKARD (SCHWEIZ) AG
Kleinrechner-Service
Allmend 2
CH-8967 Widen
Telephone: (057) 31 21 11

UNITED KINGDOM
HEWLETT-PACKARD Ltd
King Street Lane
GB-Winnersh, Wokingham
Berkshire RG11 5AR
Telephone: (0734) 784 774

International Service Information

Not all Hewlett-Packard service centers offer service for all models of HP products. However, if you bought your product from an authorized Hewlett-Packard dealer, you can be sure that service is available in the country where you bought it.

If you happen to be outside of the country where you bought your unit, you can contact the local Hewlett-Packard service center to see if service is available for it. If service is unavailable, please ship the unit to the address listed above under Obtaining Repair Service in the United States. A list of service centers for other countries can be obtained by writing to that address.

All shipping, reimportation arrangements, and customs costs are your responsibility.

Service Repair Charge

There is a standard repair charge for out-of-warranty repairs. The repair charges include all labor and materials. In the United States, the full charge is subject to the customer's local sales tax. In European countries, the full charge is subject to Value Added Tax (VAT) and similar taxes wherever applicable. All such taxes will appear as separate items on invoiced amounts.

Calculator products damaged by accident or misuse are not covered by the fixed repair charges. In these situations, repair charges will be individually determined based on time and materials.

Service Warranty

Any out-of-warranty repairs are warranted against defects in materials and workmanship for a period of 90 days from date of service.

Shipping Instructions

Should your unit require service, return it with the following items:

- o A completed Service Card, including a description of the problem.
- o A sales receipt or other proof of purchase date if the one-year warranty has not expired.

The product, the Service Card, a brief description of the problem, and (if required) the proof of purchase date should be packaged in adequate protective packaging to prevent in-transit damage. Such damage is not covered by the one-year limited warranty; Hewlett-Packard suggests that you insure the shipment to the service center. The packaged unit should be shipped to the nearest Hewlett-Packard designated collection point or service center. Contact your dealer for assistance. (If you are not in the country where you originally purchased the unit, refer to "International Service Information" above.)

Whether the unit is under warranty or not, it is your responsibility to pay shipping charges for delivery to the Hewlett-Packard service center.

After warranty repairs are completed, the service center returns the unit with postage prepaid. On out-of-warranty repairs in the United States and some other countries, the unit is returned C.O.D. (covering shipping costs and the service charge).

Further Information

Service contracts are not available. Circuitry and designs are proprietary to Hewlett-Packard, and service manuals are not available to customers. Should other problems or questions arise regarding repairs, please call your nearest Hewlett-Packard service center.

Potential for Radio/Television Interference (for U.S.A. Only)

The HP-41 generates and uses radio frequency energy and, if not installed and used properly--that is, in strict accordance with the instructions in this manual-- may cause interference to radio and television reception. It has been tested and found to comply with the limits for a Class B computing device in accordance with the specifications in Subpart J of Part 15 of FCC rules, which are designed to provide reasonable protection against such interference in a residential installation. However, there is no guarantee that interference will not occur in a particular installation. In the unlikely event that your HP-41 does cause interference to radio or television reception (which can be determined by turning off the HP-41 and then turning it on) you are encouraged to try to correct the interference by one or more of the following measures:

- Reorient the receiving antenna.
- Relocate the HP-41 with respect to the receiver.
- Move the HP-41 away from the receiver.
- Plug the HP-41 into a different outlet so that the HP-41 and the receiver are on different branch circuits.

If necessary, you should consult your dealer or an experienced radio/television technician for additional suggestions. You may find the following booklet, prepared by the Federal Communications Commission, helpful: How to Identify and Resolve Radio-TV Interference Problems. This booklet is available from the U.S. Government Printing Office, Washington, D.C. 20402, Stock Number 004-000-00345-4.

Programming and applications Assistance

Should you need technical assistance concerning programming, applications, etc., call Hewlett-Packard Customer Support at (503) 757-2000. This is not a toll-free number and we regret that we cannot accept collect calls. As an alternative, you may write to:

Hewlett-Packard
Corvallis Division Customer Support
1000 N.E. Circle Blvd.
Corvallis, OR 97330

Dealer and Product Information

For dealer locations, product information, and prices, please call (800) 547-3400. In Oregon, Alaska, and Hawaii, call (503) 758-1010.

Appendix B

Null Characters

1) Null Characters and the ALPHA Register

The null character is the ~ (overbar) and corresponds to character code 0. Normally the HP-41 does not display null characters. However, under certain conditions, using ALPHA register functions, you can place null characters in the ALPHA register.

Since the null character doesn't normally appear in the display, the HP-41 uses the null character as a special indicator. As a result, nulls in the ALPHA register occasionally cause unexpected displays, as described in this appendix.

2) Treatment of Null Characters

The distinction between the ALPHA register and the ALPHA display is important when considering the treatment of nulls.

> The ALPHA register is always 24 characters long; when it is empty, it actually contains 24 null characters. As characters enter the ALPHA register from the right side, they displace nulls. Any leading nulls (either that you enter or that were already there) remain, but they are ignored by HP-41 operations.
>
> The ALPHA display consists of the characters in the ALPHA register after the leading nulls. It starts with the first (leftmost) non-null character and displays all others to the right, including any embedded or trailing nulls.

The HP-41 and its functions always consider that an ALPHA string starts at the first non-null character, ignoring leading nulls. Nulls embedded between non-null characters are retained. Embedded nulls can be lost if the ALPHA string is rotated until a null character is leftmost.

3) Appending Characters.

If you append a character to the ALPHA register (using X-AR), the display will differ from the actual contents of the ALPHA register if the last character (before appending) was a null.

If the last character in the ALPHA register is a null, then--while you enter characters to append--the HP-41 acts like the register is empty, and displays only the characters that you are appending. (The cursor sign (_) is present in the display while you append characters.) However, the ALPHA register itself properly retains the original string and combines it with the appended string.

You can view the full, appended contents by pressing f[AVIEW] or [ALPHA][ALPHA]. (Remember that leading nulls are never displayed.

4) Deleting Characters While Appending.

If you use f[APPEND] or f[ARCL] and the last character in the ALPHA string is a null, using backarrow to delete the rightmost character will clear the entire ALPHA register. This is because when a null character gets deleted the HP-41 figures that it has encountered the leading nulls that precede a string, and it concludes that the register is empty--so it clears everything.

5) ALPHA Strings in Data or Stack Registers.

If you store an ALPHA string containing nulls in a data or stack register, none of the nulls will be displayed when you view (or print) the contents of that register (as with f[VIEW] or [RCL]). However, if you recall those contents to the ALPHA register and then view them (f[ARCL]), all the characters in the ALPHA data string will be displayed (except, of course, leading nulls).

(If you print out the ALPHA string contents of a data or stack register, the results are different and incomplete.)

Appendix C

HP-41 HP-IL Integrated Circuit

This appendix documents the registers of the HP-41 HP-IL integrated circuit. Some of the registers have different meanings depending upon whether they are being read from or written to. For further information, see the "HP-IL Integrated Circuit" manual.

The HP-41 HP-IL integrated circuit is slightly different from the general purpose integrated circuit. These differences are mostly concerned with the HP-41 bus interface. All differences are indicated by a star ("*").

Generally speaking, the only registers that you will need to directly read or write are registers zero, three, and four. Registers one and two are adequately serviced by RFRM, WFRM, and the five tests IFCR?, SRQR?, etc. Registers five, six, and seven are used by the HP-IL module at various times. Should you wish to write register one, always write a zero to FLGEN.

Status Register	7	6	5	4	3	2	1	0
0 Read	SC	CA	TA	LA	SSRQ	RFCR	CLIFCR	MCL
0 Write	SC	CA	TA	LA	SSRQ	SLRDY	CLIFCR	MCL

SC - System Controller
CA - Controller Active
TA - Talker Active
LA - Listener Actve
SSRQ - Send Service Request
RFCR - RFC Received
SLRDY - Set Local Ready
CLIFCR - Clear IFC Recv
MCL - Master Clear

Note: SLRDY and CLIFCR are self-resetting bits (resetting occurs 1-2 usec after end of write pulse.) Reading R0 returns the value of CLIFCR, which will always be a logic zero by the time the HP-41 reads it.

Control Interrupt Register	7	6	5	4	3	2	1	0
1 Read	CI2	CI1	CI0	IFCR	SRQR	FRAV	FRNS	ORAV
1 Write	CO2	CO1	CO0	*-	*-	*-	*-	*FLGEN

CI2-CI0 - Input Control Bits
CO2-CO0 - Output Control Bits
IFCR - Interface Clear Received
SRQR - Service Request Received
FRAV - Frame Available
FRNS - Frame Received Not as Sent.
ORAV - Output Register Available.
*FLGEN - Enable FI Line

Data bits	7	6	5	4	3	2	1	0
Register 2 Read	DI7	DI6	DI5	DI4	DI3	DI2	DI1	DI0
2 Write	DO7	DO6	DO5	DO4	DO3	DO2	DO1	DO0

DI7-DI0 - Input Data Bits
DO7-DO0 - Output Data Bits

Parallel Poll	7	6	5	4	3	2	1	0
Register 3 R/W	*OSCDIS	*AIDY	PPIST	PPEN	PPOL	P2	P1	P0

*OSCDIS - Oscillator Disable
*AIDY - Automatic IDY Sourcing in Idle Mode
PPIST - Parallel Poll Individual Status
PPEN - Parallel Poll Enable
PPOL - Parallel Poll Polarity
P2-P0 - Parallel Poll Response Bit Designation

Loop Address	7	6	5	4	3	2	1	0
Register 4 R/W	S	S	S	ADR4	ADR3	ADR2	ADR1	ADR0

S - Scratch Bits
ADR4-ADR0 - Address Bits

Scratch	7	6	5	4	3	2	1	0
Register 5 R/W	S	S	S	S	S	S	S	S

Scratch Register 6 R/W	7	6	5	4	3	2	1	0
	S	S	S	S	S	S	S	S

Scratch Register 7 R/W	7	6	5	4	3	2	1	0
	*S	*S	*S	*S	*S	*S	*S	*S

Integrated Circuit Initialization

The HP-41 HP-IL integrated circuit provides two levels of initialization, RESET and MCL. RESET occurs whenever the HP-41 is turned off or whenever the HP-41 goes to idle mode and is not in AIDY mode. MCL occurs whenever RESET occurs or whenever a one is written into the MCL bit in register zero.

RESET turns off the internal oscillator, and causes MCL to be set.

MCL resets IFCR, SRQR, FRNS, and FRAV, sets ORAV, resets *FLGEN, and sets SC (*SC is tied high).

There are three initialization states possible: oscillator off, cleared; oscillator on, cleared; oscillator on, running. The transition between these states is indicated in Figure 1.

Automatic IDY mode

The HP-IL integrated circuit will not be reset in idle mode if the AIDY bit is set, which allows for two possibilities:

1) If CA is true, then IDYs will be automatically generated. No handshaking of the IDYs will be performed. The eight data bits of the IDY are undefined. The HP-41 will start executing if SRQR goes high, indicating that some device wants service.

2) If CA if false, then the HP-41 will start executing if IFCR, FRAV, or FRNS is set.

Neither RESET nor MCL affect AIDY; however, AIDY will power on low when the HP-IL module is first plugged into the HP-41.

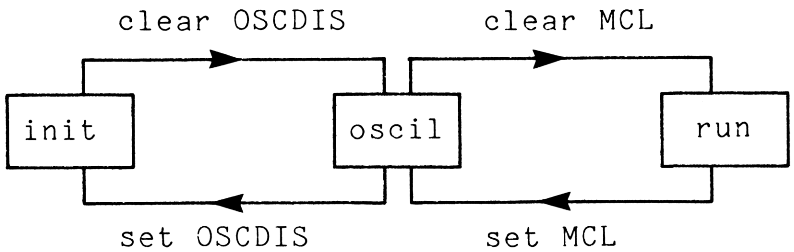


Figure 1. - Initialization Transitions

Appendix D

Error Messages

This appendix contains a list of messages and errors that are related to module operations.

Display	Functions	Meaning
ADR ERR	[LAD] [TAD] [DDL] [DDT]	The address or device dependent function did not fall into the range of zero through 31, inclusive.
ALPHA DATA	-all-	ALPHA characters are in a register where a number is required - either a stack register or a data storage register.
BSIZE>1771	[BSIZEX]	The desired buffer size is too large.
DATA ERROR	-all-	The specified number is out of range.
END OF BUF	-all buffer functions-	The end of the buffer has been reached. The buffer pointer (if in auto-increment mode) is left pointing at the end of the buffer plus one.
ETO ERR	-all-	The message following the data messages was not an ETO.
FRNS ERR	-all-	A message was not received as sent.
INVALID REG	[WREG] [RREG]	The HP-IL register specified does not exist.
NO BUFFER	-all-	A buffer has not been created with [BSIZEX].
NO RESPONSE	[SST] [SAI] [SDI] [TCT]	No response to SST, SAI, SDI, or TCT. The listener did not respond to the specified message.

NONEXISTENT	-all-	The specified number is out of range or the specified register does not exist.
ORAV=0	-all-	ORAV did not go true within ten seconds.
OUT OF RANGE	-all-	The specified number is out of range.
PACKING TRY AGAIN	[BSIZEX]	Executed from the keyboard, memory is too small for the specified buffer size. Memory is packed. Try the operation again, or reallocate registers or add memory module(s).
TIME OUT	-all-	A message was not received after a time specified by the function specified.
TRANSMIT ERR	-none-	If flag 33 is set, execution of any HP-IL module functions will cause this error. The development module does not generate this error.

Appendix E

Function Index

HP 00041-15043 HP-IL Developement Module
Owner's Manual

HEWLETT
PACKARD

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

00041-90449

Printed in U.S.A. 10/82