Hewlett-Packard System 45 Desktop Computer

I/O ROM Programming

Manual Changes

HP System 45 Desktop Computer
I/O ROM Programming

(For Manual P/N 09845-90060, dated June 1, 1978)

Page 24:

The last sentence in the third paragraph should read -

The 257th character corresponds to a numeric value of 0.

Page 26:

The program shou d be -

```
10   !
20   !  PROGRAM SEQUENCE TO SET UP AN ASCII-TO-EIA CONVERSION TABLE
30 DATA 48,32,49,1,50,2,51,19,52,4,53,21,54,22,55,7,56,8,57,25,44,59,46,107,10,128
40   FOR I=1 TO 13
45   ! READ THE NEXT ASCII CHARACTER VALUE AND THE NEXT EIA CHARACTER VALUE
50   READ K,L
60   ! THE ASCII CHAR. VALUE IS CONVERTED TO THE EIA CHAR. VALUE ON OUTPUT
61   ! THE EIA CHAR. VALUE IS CONVERTED TO THE ASCII CHAR. VALUE ON INPUT
65   CONVERT Sel_code;"OI",K TO L
70   NEXT I
80   ! THE CONVERSION TABLE FOR "Sel_code" IS NOW SET UP FOR ASCII-EIA NUMERIC
90   ! CHARACTER CONVERSION
100  END
```

Page 27:

The syntax for the CONVERT *statement should be -*

CONVERT sc ; "mode"

Page 70:

The program shou'd be -

```
350  !
360  A=45
370  FOR I=1 TO 16
380  A=SHIFT(A,-1)
390  Z=LASTBIT
400  PRINT Z;
410  NEXT I
420  PRINT
430  END
```

Page 99:

The Output Format Equivalents shɔuld be changed to -

9830A line 20 should be - `20 OUTPUT (6,10) 34,91,93,10.2,97.4,25.3`
9825A line 11 should be - `11wrt6.1,34,91,93,10.2,97.4,25.3`
9845A line 20 should be - `20 OUTPUT 6 USING 10;34,91,93,10.2,97.4,25.3`

This supplement is intended to providə additional information concerning known situations which may arise when using the Systerr 45 I/O ROM.

HP-IB

- HP-IB operations **do not react** tɔ the `SELECT CODE... INACTIVE` statement. Sending an HP-IB operation (suc ı as Trigger, Reset,..etc.) to any HP-IB devices which have been deactivated may result in the stoppage of all processing in the System 45.

OUTPUT

- An `OUTPUT` statement of the forın `OUTPUT` (select code) `BDMA; A$[2,256]` may result in the first byte of the charıcter string being missed, if the total number of bytes being sent is an odd number. Aıı easy way to avoid this problem is to keep the first subscript of A$ from being an eveıı value; make it odd (1,3,5...etc).

- An `OUTPUT` statement like `OUTPUT A$(Any value);` data list may result in an `ERROR 18` if the dimensioned s ze of A$ plus the length of the data list exceeds 32 767 bytes (32K). Note that it is the size of A$ set by the `DIM` statement, rather than the actual length of A$ (`LEN`).

ENTER

- If the length of A$ does not equal () before executing the statement `ENTER` (select code) `BFHS; A$` an error may occur if ٨$ is terminated on an EOI. The error is that the length of A$ may be incorrect. The solıtion is to null out A$ **before** executing the `ENTER` statement.

VAL$

- The `VAL$` function is **not recomınended** for use in I/O statements. For example,

- `SENDBUS VAL(VAL$(Any value))` causes the interface to disable the handshake.

- `STATUS VAL(VAL$(Any value))` causes the interface to disable the handshake.

SENDBUS

- As mentioned previously, `SENDBUS VAL (VAL$ (Anyvalue) )` causes the interface to disable the handshake.

- The `SENDBUS` statement **does not work** in the OVERLAP mode.

Conversions

- If you are having the System 45 perform eight (8) bit conversions (as opposed to seven (7) bit), you **must** specify a separate Input and Output conversion table. Otherwise, the conversion may not be done completely. For example, `CONVERT 2;"I";A$` and `CONVERT 2;"O";B$` rather than `CONVERT 2;"IO";A$`.

- The 0th character in a `CONVERT` is taken from string element 257, instead of 256. The information in the I/O manual on page 24 should be changed to reflect this.

I/O Cards

- When using a 98032A Interface Card, terminating a DMA operation with an EIR interrupt causes the System 45 to stop all processing operations.

- After a Reset operation, when using a 98036A Interface Card, check that `IOFLAG` equals 1 before proceeding with a `CARD ENABLE` statement.

Enter on Interrupt

- The practice of entering on an interrupt basis may cause a problem with the System 45 if the entering is from a device which requires operator intervention (i.e...a digitizer or terminal).

  The problem arises out of the System 45's allocation of R/W memory to serve as the I/O buffers. The System 45 allocates buffer space serially through the R/W memory while continuing its processing operations. Eventually, the System 45 reaches the end of the R/W memory. When this occurs, the System 45 recycles itself back to the beginning of the R/W memory to continue allocation of the buffer space. If one of the previously allocated I/O buffers is still busy because an `ENTER` statement has not be completed AND the System 45 needs to reclaim the buffer space, all processing stops. The System 45 remains in this state until the operator completes the `ENTER` by taking the required action.

  The solution to this problem is to periodically complete an `ENTER` statement from this operator controlled device. This has the effect of creating a "window" where the data is accepted. The statement can always be re-executed under program control as needed.

- Another possible problem occurs when an ENTER statement from a device requires an OUTPUT statement to the same device to effect a data transfer. This can be thought of as routing the OUTPUT from one select code device as the input to another select code device. Since the buffers are allocated serially until the memory is exhausted, it is possible for an ENTER statement to take the last available buffer. If the next statement is an OUTPUT, as mentioned previously, the OUTPUT attempts to allocate a buffer as well. Since all buffer space is used the System 45 attempts to allocate the buffer space but cannot because the ENTER statement has not been completed. The end result of this is that all processing stops.

  The solution to this problem is that any type of loop where the data in the ENTER statement results in the OUTPUT statement being fed back to the same device, should be avoided.

Overlap

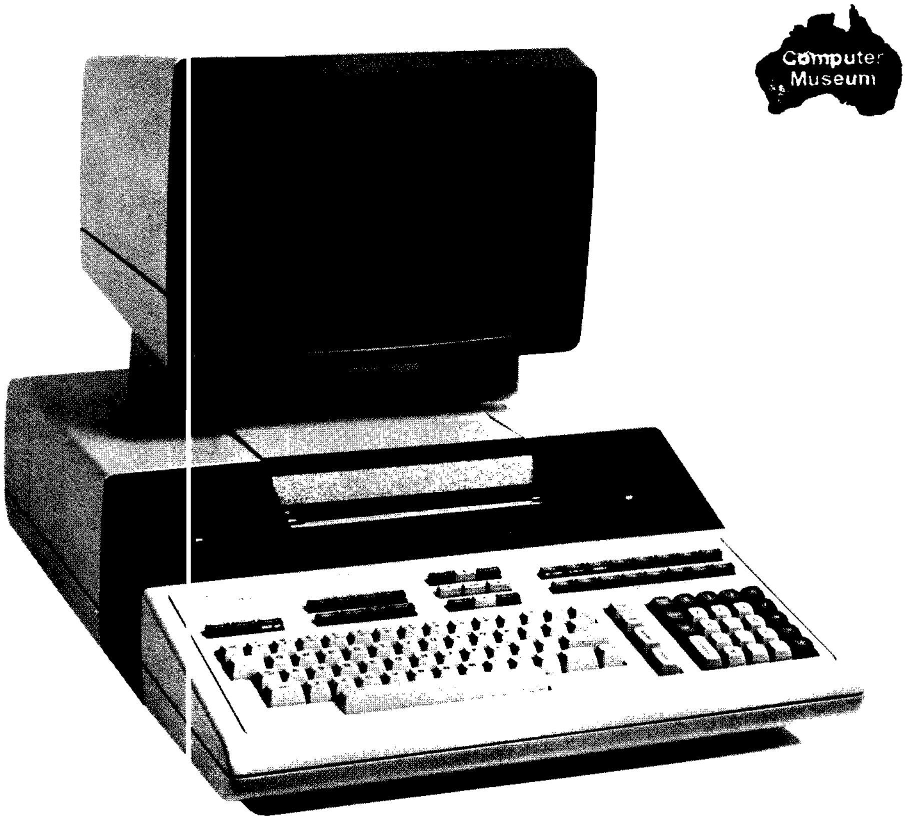
- A problem may arise in the OVERLAP mode of operation. This occurs when using an ENTER statement with DMA.

  The problem is that an End of Line Branch does not occur upon completion of the DMA, possibly causing the System 45 to stop all processing. At this point, the System 45 must be reset by a CONTROL STOP sequence.

  The best way to avoid this problem is to place the System 45 in the SERIAL mode rather than OVERLAP prior to performing any DMA transfers.

  This problem does not occur when executing an OUTPUT DMA statement.

I/O ROM Programming



HP System 45 Desktop Computer

Hewlett-Packard Desktop Computer Division
3400 East Harmony Road, Fort Collins, Colorado 80521
(For World-wide Sales and Service Offices see back of manual.)

Table of Contents

Chapter 1: General Information

Chapter 2: General I/O Operations

Chapter 3: Advanced I/O Operations

Appendix A

Appendix B

Appendix C

Appendix D

Appendix E

Chapter 1
General Information

Manual Description

This manual is written with ycu in mind. A beginner's introduction is provided at the start of each chapter explaining the operations covered in that chapter. More advanced users and programmers can skip over the introduction, as the syntax and operation of each program statement are fully explained i1 the associated section.

The chapters and their basic ccntents are summarized here to give you an idea of where to look to find the information you ne3d:

- General I/O Operations Chapter topics include input and output, character conversion and parity checking, and interface control and status.
- Advanced I/O Operaticns. Topics covered are program interrupt branching, device timeouts, use of input and output transfers, unformatted I/O for computer-to-computer communications and overlapped input, and notes on use of overlapped input and output.
- Binary Operations. This chapter includes the functions for manipulating and testing binary values.
- HP-IB Operations. Meta Messages and supporting statements and functions for complete control of the Hewlett-Packard Interface Bus are the topics covered in this chapter.
- Appendices. A syntax summary, technical data section on the HP-IB, keyboard code diagram, and listing of the I/O ROM error messages are among the details included in the manual appendices.

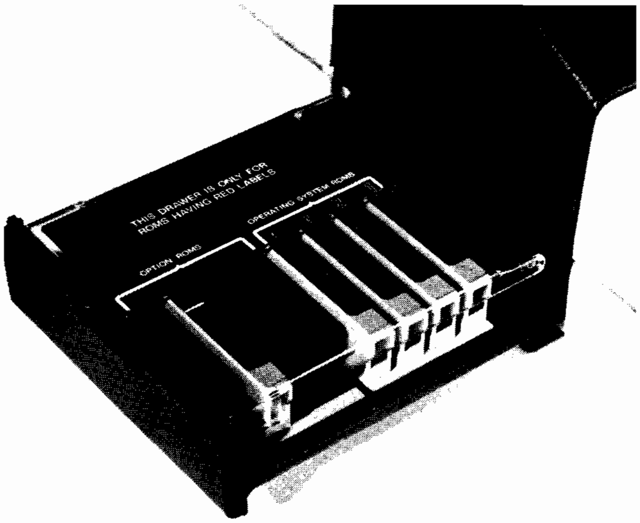
ROM Description

The HP 9845-98432A I/O ROM package adds extensive input, output, and device control facilities to your System 45 desktop computer. The I/O ROM package consists of two ROMs adding a total of 22K bytes to the System 45 operating system. When installed, the I/O ROM package reserves a total of 712 bytes of Read/Write Memory, which cannot be accessed for storing programs or data. During input and output operations, the actual memory requirements of the I/O ROM vary due to data formatting, conversions, transfer buffering, and other functions that are typically performed when any I/O is in progress.

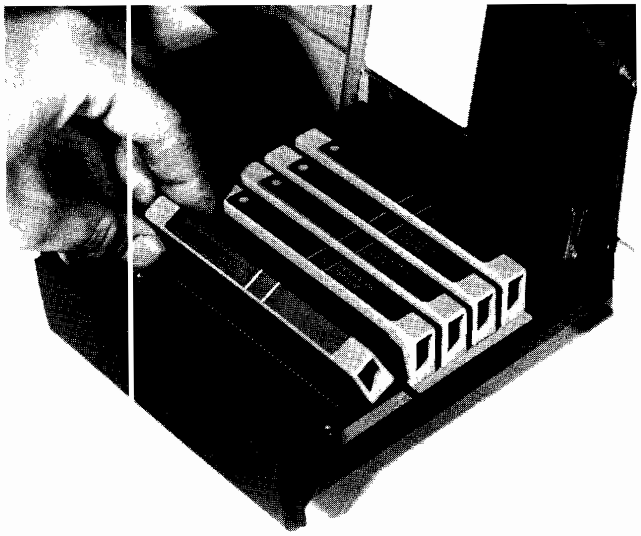
ROM Installation and Checkout

The System 45 I/O ROM is a two-part ROM that installs in the ROM drawers as follows:

1. **Turn off computer power.**
2. Slide the right ROM drawer (brown label ROMs) all the way out.
3. Open the plastic cover by squeezing the sides of the cover to gain access to the connectors as shown.
4. Take the brown-labeled I/O ROM and orient it so the label reads the same as the operating system ROMs –



5. Press the brown-labeled I/O ROM onto an option ROM drawer connector so that it seats all the way down as shown below. The small raised rib on the top of the drawer should fit into the recess on the bottom of the ROM. If it doesn't, make sure you've properly oriented the ROM and that the label color matches the ROM drawer label color.



6. When the brown-labeled ROM has been installed, snap the plastic cover shut and replace the ROM drawer, making sure it is flush with the computer housing.
7. Repeat the above process to place the remaining red-labeled ROM into the left ROM drawer.
8. Double-check the ROM drawers to make sure they are all the way in. They should be flush with the System 45 housing.
9. Turn on power and verify operation of your System 45 by running the System 45 test.

Syntax Guidelines

The following conventions apply to the program statements and functions found in this manual.

DOT MATRIX : All items printed in dot matrix are required exactly as shown.

[ ] : All items in square brackets are optional unless the brackets are printed in dot matrix.

| : A vertical bar is read as "or".

... : Three dots indicate that successive parameters are allowed, when each is separated by a comma.

{ } : When more than one item appears in an item list with no separators, individual items are enclosed in braces.

A complete list of the syntax for the I/O ROM statements and functions is given in the appendix to this manual.

Error Messages

The I/O ROM adds error messages **150-167** to the System 45 error message list. These errors are listed and explained in the last pages of this manual.

Chapter 2
Elementary I/O Operations

This chapter covers the basic topics necessary to perform simple interfacing tasks. Topics covered include:

- Some elementary I/O terminology.
- Some basic peripheral addressing information.
- Elementary input and output statements.
- Formatted input and output using image specifiers.
- Data conversions for non-ASCII communication.
- Limited interface control operations.

Introduction

The System 45 Elementary I/O operations give your computer the capability of sending and receiving data through interface cards on the System 45 interface backplane. Several statements are added to the basic language set of the System 45 when the I/O ROM is present, providing extensive control of the computer's input and output. This chapter covers the operations that can be used for basic applications such as output to printers or tape punches or input from devices such as tape readers. You can easily learn and use the operations discussed in this chapter to "get on board" quickly. More advanced techniques are covered in the chapter on Advanced I/O Operations; if you need high-speed I/O transfers or peripheral device interrupt capabilities, refer to that chapter.

Some I/O Terms

The computer world is full of terms that may or may not have had meanings already assigned to them, and some of the terminology can be confusing; this section describes some of the I/O-related terms used in this manual.

"**Handshake**" is a term describing the sequence of operations that occur when a unit of data is transferred to or from the computer. The basic purpose of a "handshake" is to ensure a proper timing for a transfer of data: 1. the device receiving the data signals when it is ready to accept data, 2. the device sending data signals when the data is available, 3. then the device receiving data signals when it has accepted the data. There are variations in the "handshake" methods actually used, but the techniques used follow basically the same format as described above.

"**Interrupt**" is a single term used to describe two levels of events; on the hardware level, an interrupt is a signal to the computer that some external event has occurred, such as a device malfunction or device ready for data. On the program level, an end-of-line branch[1] causes program execution to vector to a special portion of the program that handles the interrupt in a manner specifically designed by the programmer. The programmer can tailor the service routine to deal with the specific application at hand.

The terms "**input**" and "**output**" are two of the most frequently used terms in this manual, and are defined as follows:

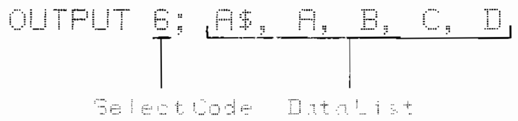
input: a transfer of data or information from an external device to the computer.

output: a transfer of data or information from the computer to an external device.

Both terms are used relative to the **computer**, i.e., "input" is data sent **to** the computer while "output" is data sent **from** the computer.

[1] End-of-line branching is covered in more detail in the Advanced I/O Operations chapter.

The external device of an input operation is the data **source**, while the external device of an output operation is the data **destination**. An external device is selected or **addressed** by its **select code**. For example, an OUTPUT statement is used to transfer data from the computer (the System 45) to a printer. As part of the statement itself, it is necessary to address the printer by its select code, and to supply a list of the data to be sent to the printer. A typical statement might be:



```
OUTPUT 6; A$, A, B, C, D
```

In this case, the select code of the printer specifies the **destination** of the data contained in the variables as A$, A, B, C, and D.

Select Codes

There are several considerations to make when deciding what select codes to use, and this section deals with the System 45 interface scheme and the role of select codes.

The System 45 I/O bus has 17 separate and distinct addresses for interfaces, and each interface is addressed by its select code. To prevent the possibility of interference, each interface must be given a unique select code, with no two interfaces having the same select code.

The System 45 has several internal interfaces including the internal printer, the keyboard, the two cartridge drives, the CRT, and the graphics option. Each of these interfaces has a reserved select code that cannot be used for external interfaces. The following chart illustrates the select code assignments of the System 45 computer.

Select Code Assignments

Select Code	Device
0	Keyboard, Printer
1	
•	
•	1-12 for external interfaces
•	
12	
13	Graphics
14	Tape drive
15	Tape drive
16	CRT

Each interface card has a switch that can be set to a select code of 0 through 15, but it must not be set to the same select code as another interface, external or internal.

NOTE

An interface card must not be set to a select code of 0, 13, 14, 15 or to the same select code as another interface card. Erroneous operation and possible loss of data can result from such a configuration.

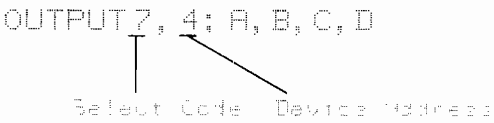
An example of a system select code assignment might be as follows:

Select Code	Device
6	Line Printer
7	HP-IB instrument system
9	Real Time Clock
10	Auxiliary terminal(asynchronous, serial interface)

To address a particular device or interface, the syntax of an input (ENTER or READBIN) statement or output (OUTPUT statement has a parameter for the select code of the interface. The select code can be represented by a numeric expression, a variable, or a constant. For example, each of the following statements addresses a printer on select code 6:

```
 80   OUTPUT 6;"       THIS STATEMENT OUTPUTS TO SELECT CODE 6"
 90   Sel=6
 100  OUTPUT Sel;A$ ! OUTPLT TO Sel, SELECT CODE 6
 110  Sel=4*Sel
 120  OUTPUT Sel/4;"  THIS STATEMENT ALSO OUTPUTS TO SELECT CODE 6"
```

Addressing a particular device on the HP-IB requires both the interface select code and the device's bus address to be specified, as shown below:


OUTPUT 7, 4; A, B, C, D

For HP-IB addressing the select code parameter can be expressed as a numeric expression, a constant or a variable specifying the interface select code, and the device address. The HP-IB select code parameter can be specified by either an interface select code, a comma, and the device address, or as a combined HP-IB address. The form of the combined HP-IB address is the interface select code and a two digit HP-IB device address. The combined HP-IB address form has the advantage that only one variable is needed to specify both the select code and the HP-IB device address, for example, Dvm = 704. Both types of addressing are shown below:

```
OUTPUT 7, 4; Q$
      or
OUTPUT 704; Q$
```

```
 110  OUTPUT 7,3;"         SELECT CODE 7, DEVICE ADDRESS 3"
 120  OUTPUT 7,3,6;A$  !  SELECT CODE 7, DEVICES 3 AND 6
 130  OUTPUT 704,706;A$ ! HP-IB ADDRESSES 704 AND 706
```

There are additional provisions available for extended addressing and secondary commands when communicating on the HP-IB; refer to the chapter on HP-IB operations for further details on HP-IB addressing.

Output Operations

There are several output-related syntaxes available with the I/O ROM. Two of these syntaxes are covered in this section. The OUTPUT statement initiates a data output operation that transfers data from the System 45 to an external device, to the CRT, or to the optional internal printer. The output transfer can be formatted by an IMAGE statement. The new image specifiers available with the I'O ROM can also be used to format a PRINT image reference. Additional OUTPUT capabilities are provided which are covered in detail in the Advanced I/O Operations chapter.

The second OUTPUT operation covered in this section is the EOL statement. The EOL statement replaces the normal carriage-return/line-feed sequence of the "L" image specifier with a user defined sequence for purposes such as CRT image formatting.

The OUTPUT Statement

The OUTPUT statement transfers data from the System 45 computer to an external device or to the System 45's CRT or internal printer. Access to external devices is through interface cards plugged into the I/O backplane of the System 45; each device is addressed by the select code of the interface card associated with the device. In addition to the OUTPUT syntaxes listed here, there are specialized syntaxes for high-speed and overlapped output transfers; these are covered in the chapter on Advanced I/O Operations.

Syntax:

OUTPUT sc [WHS][USING image]; data list

sc: The sc parameter specifies the select code (0-12,16) of the interface or the HP-IB address(es) of one or more devices. Refer to the section covering select codes for details on select code specifications.

WHS: Word Handshake specifies a 16 bit output rather than the default 8 bit output.

image: The image parameter is either a string expression specifying a valid format string or the line identifier of an IMAGE statement. If no reference is made to an image specification, then the default mode of freefield string and numeric output is used.

data list: The data list may consist of any of the following, separated by commas or semicolons:

variable names

array identifiers

numeric expressions

string expressions

If an image reference is not specified, the output format defaults to the freefield format as described for the PRINT statement in the System 45 Operating and Programming Manual. At the completion of each line of output, a carriage-return/line-feed sequence is output. (See the next section for details on specifying a different end-of-line sequence.)

The OUTPUT statement can be used in conjunction with a conversion table for changing output characters to a different character code. This is covered in the section on Conversions later in this chapter.

The following statement outputs the concatenation of A$ and B$, then outputs the result of the expression (A↑2+B/4) to select code 16 (the CRT):

```
OUTPUT 16 ; A$&B$ ; A↑2+B/4
```

Formatted output is accomplished using the OUTPUT image specifiers, which include the PRINT USING image specifiers and B, Y, L, and W. If the image parameter of the OUTPUT statement is a string expression, the string expression must be a valid format string at the time of execution. If the image parameter is a line identifier, it must refer to an IMAGE statement.

The format string is a list of field specifiers separated by delimiters. It specifies numeric and string fields, blanks, and carriage control. Each numeric or string field specifier must correspond to an appropriate item in the data list.

The allowable delimiters used to separate field specifiers are:

, The comma is used only to separate two specifiers.

/ The slash separates two specifiers and causes a C/R-LF sequence to be output.

@ The commercial "at" sign separates two specifiers and causes a top-of-form to be output.

The allowable field specifiers are:

[n]X: The X specification causes a blank to be output. It can be replicated [n] times, and can be imbedded within any other field specifier without delimiters.

[n]L: The L specification causes [n] EOL (end-of-line) character sequences to be output.

"...": Text enclosed within quotes may be imbedded without delimiters within any other field specifier. A literal specifier (text within quotes) may only appear in an IMAGE statement.

[n]A: The A specifier causes one character of the associated output string to be output. [n]A specifies [n] characters of output for the corresponding string variable.

[n]D: The D specifier sets [n] number of digit positions. Leading zeroes are replaced with a blank space as a fill character.

[n]Z: The Z specifier sets [n] number of digit positions. Leading zeroes are printed.

[n]*: The * specifier sets [n] number of digit positions. Leading zeroes are replaced with * as a fill character.

.: Specifies a decimal point radix symbol to be output in this position.

R: Specifies a comma radix symbol to be output in this position.

S: Specifies output of a sign:
+ if the number is positive,
− if the number is negative.

M: Specifies output of a sign:
a blank if the number is positive,
− if the number is negative.

C: Specifies a comma as a separator in the specified position.

P: Specifies a period as a separator in the specified position.

E: Specifies that the numeric field in which it is contained is to be output in scientific notation. An E, exponent sign (+ or −), and a two digit exponent are output. At least one digit symbol must precede the E symbol.

K: Specifies the corresponding data item is to be output as follows:

- numeric items are output in STANDARD format, with no leading or trailing blanks.
- string items are output with no leading or trailing blanks, and the entire string is output.

B : Specifies one byte of two's complement binary information to be output. The associated variable or expression in the output list is output as a value between − 128 (1000 0000) and 255 (1111 1111).

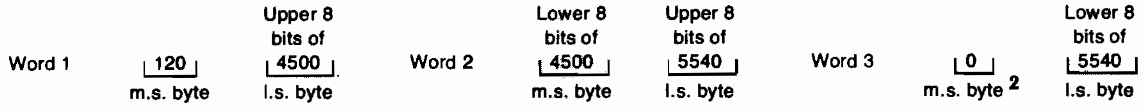
W : Specifies two bytes of two's complement binary information to be output. The associated variable or expression in the output list must have a value between − 32768 and 32767. Both bytes are output simultaneously on a 16 bit interface by specifying a word handshake transfer (WHS) in the OUTPUT statement.

Y: Specifies two bytes of two's complement binary information to be output. The associated variable or expression in the output list must have a value between − 32768 and 32767. Because this value is packed for output, both bytes may not appear at the same time on a 16 bit interface.

For example, with a 16 bit interface on select code 6:

```
OUTPUT 6WHS USING "B,Y,Y";120,4500,5540
```

outputs three 16 bit words.
The output looks like this:



[n] (...): Image specifiers can be replicated [n] times by enclosing them in parentheses. One or more image specifiers can be replicated within the parentheses.

+: The + image specifier suppresses the line-feed normally output at the end of the output list.

[2] This is a zero-fill character to fill the remaining 8 bits of the 16 bit word.

-: The - image specifier suppress the carriage-return normally output at the end of the output list.

#: The # image specifier suppresses both the carriage-return and the line-feed normally output at the end of the output list.

The statements below illustrate the syntax necessary to utilize an image reference:

```
152   !
160  OPTION BASE 1
170  DIM A(10,10),A$[50]
180  A$="TEST STRING FOR OUTPUT EXAMPLES"
190  MAT A=CON
200  Img$="10(10(DXX)/),K"       ! SET UP AN IMAGE STRING
220   !  FOR 10 LINES OF 10 SINGLE DIGIT NUMBERS PER LINE
230   !  FOLLOWED BY A STRING OF OUTPUT.
240   !  REFERENCE THE IMAGE STRING Img$ IN THE FOLLOWING OUTPUT.
250   !
260  OUTPUT 16 USING Img$;A(*);A$
270   !
280   !  NOW OUTPUT USING AN IMAGE LITERAL.
290   !  THE FORMAT IS IDENTICAL TO Img$ ABOVE.
300   !
310  OUTPUT 16 USING "10(10(DXX)/),K";A(*);A$
320   !
```

The OUTPUT statement can be used in conjunction with data conversions and parity generation as well. This information is presented in the Data Conversions section of this chapter.

The End-of-Line Statement

The EOL statement provides the capability of specifying the end-of-line sequence and a line-to-line output delay to a selected device. During normal output operations such as output to an impact printer, a carriage-return/ line-feed sequence is sent at the end of each line of output. Under certain circumstances, however (such as display formatting), it may be desirable to output a different end-of-line sequence. For instance, the HP 2640 CRT terminal provides a special display enhancement mode requiring special characters after a carriage-return/line-feed in order to maintain the enhancement mode. Other devices such as teleprinters have physical limitations on the speed of a carriage-return: after a carriage-return it is necessary to delay a certain length of time before initiating another line of output.

Syntax:

EOL sc; seq[, del]

EOL sc ; del

sc: The select code parameter specifies the select code (0-12, 16) or the HP-IB address of the device for which the EOL statement is executed.

seq: The {seq} parameter is a string variable or expression that specifies the end-of-line sequence to be output for the "L" image specifier. The default sequence is a CR-LF.

del: The {del} parameter is a numeric expression specifying the minimum number of milliseconds to delay between successive lines of output.

Use the # image specifier to disable the normal CR-LF sequence that is sent at the end of the data list by the OUTPUT statement if you wish to replace CR-LF with your own end-of-line sequence.

Syntax:

```
EOL sc
```

Executing EOL sc with no other parameters cancels the end-of-line sequence previously set up by an EOL statement.

As an example using the EOL statement to cause double spacing to a printer on select code 6, and allowing for a carriage-return time of 150 milliseconds, the following loop outputs the string array A$:

```
350    !
351    S=6   ! SELECT CODE 6.
360   DIM A$[80]
361    ! Seq$ IS THE EOL SEQUENCE STRING: CARRIAGE-RETURN,2 LINE-FEEDS.
370   Seq$=CHR$(13)&CHR$(10)&CHR$(10)
380   A$="THIS IS THE TEST STRING TO DEMONSTRATE THE EOL SEQUENCE"
381    !  THE EOL STATEMENT  BELOW ENABLES DOUBLE-SPACED OUTPUT
382    !  AND A 150 MILLISECOND DELAY PER LINE.
390   EOL S;Seq$,150
400 Loop:  REM
401    !
410    !  NOTE THE FOLLOWING OUTPUT STATEMENT REFERENCES AN IMAGE
420    !  OF DISABLED C/R-L/F, COMPACTED STRING OUTPUT, AND EOL.
430    !  THE C/R-L/F IS DISABLED (#) SO ONLY THE EOL SEQUENCE IS SENT.
440    !
450   OUTPUT S USING "#,K,L";A$
460   GOTO Loop
470   END
480    !
```

Input Operations

The I/O ROM provides several statements for reading in data from external sources. Two basic input statements are covered here. The ENTER statement has the capability of formatting and converting incoming data, while the READBIN statement provides a means of inputting binary data from an interface.

The ENTER Statement

The ENTER statement initiates a transfer of data from an external device to the System 45. The data can be entered into numeric variables, strings, and numeric and string arrays. The external device is referenced by the select code parameter of the ENTER statement. High-speed input is also available using the ENTER statement; details on high-speed input and overlapped data entry are covered in the Advanced I/O Operations chapter.

Syntax:

ENTER sc[WHS][USING image]; enter list

sc: The select code parameter specifies the select code (1-12) of the interface or the HP-IB address of the device that is the source of data.

WHS: Specifying word handshake causes 16 bits of data per handshake to be input from the interface rather than the default of 8 bits.

enter list: The enter list may consist of any combination of numeric or string variables, string arrays, or array identifiers. Items in the list are separated by commas or semicolons.

image: The image parameter is either a string expression containing a valid ENTER image specification, or the line identifier of a valid IMAGE statement. If an image specifier is not referenced by the ENTER statement, the format defaults to a freefield format. The freefield formats are explained under the F (for numeric freefield) and the T (for string variable freefield) image specifiers.

ENTER IMAGE Specifiers:

Image specifiers are combined into a string expression by separating individual items with delimiters. The delimiters usec to separate items are the comma and the slash (/). The slash can also be used as an image specifier, its use is described below.
An example of a typical IMAGE statement is

```
10 IMAGE F, B/F
```

F: The F image when specified sets a numeric freefield format with the decimal point as the radix symbol. Leading blanks and non-numeric characters are ignored, while a non-numeric character following a numeric data item is treated as a delimiter. Numeric data characters include +, −, e, E, the radix symbol, and digits 0-9. A solitary e or E is not considered a numeric data character. For example, using the F image specifier to enter the string "ZV1.43f1.8432e6D" from select code 8, the following statement –

```
ENTER 8 USING "F"; A, Q
```

produces the results:
A = 1.43
Q = 1.8432 E6

H: The H image specifier is identical to the F specifier above except that the comma is recognized as the radix symbol. The H specifier is used for European freefield numeric data input.

[n]N: The N image specifier sets the number of digits [n] received per numeric data item. Non-numeric characters are counted as a digit but not entered, and a decimal point is interpreted as a radix point. For example, the input data string "$3.5796,$3,467.88, $0.20" can be entered as follows –

```
ENTER 8 USING "8N, 10N, 5N"; G [1], G [2], G [3]
```

producing the results:
G [1] = 3.5796
G [2] = 3467.88
G [3] = .20

[n]G: The G image specifier is identical to the N specifier, except a comma rather than a decimal point is interpreted as the radix symbol. The G specifier is used for European fixed-field numeric data input.

B: The B image specifier causes one byte of data to be entered into a numeric array variable. The incoming byte is converted to the corresponding variable type of the ENTER statement. The form [n] B is not allowed.

Y: The Y image specifier enters two bytes of input data into a numeric or numeric array variable. The first byte received becomes the most significant eight bits, while the second byte received becomes the least significant eight bits. The 16 bit word is converted to the data type of the enter variable. The form [n] Y is not allowed.

W: The W image specifier inputs one 16-bit word from an interface and stores it into the appropriate numeric or numeric array variable. The form [n] W is not allowed. If WHS is not specified, only 8 meaningful bits are entered. The upper 8 bits are reset to zero.

T: The T image specifier enters string data in freefield format. If a carriage-return is followed by a line-feed, the carriage-return is not entered into the string. The string is filled until either a line-feed or EOI terminator is received or until the number of characters entered equals the dimensioned length of the string. The line-feed and EOI terminators for the T image specifier are not disabled by the #, +, or % specifiers.

[n]A: The A image specifier enters [n] number of characters into a string variable.

[n]X: The X image specifier skips [n] number of input characters.

[n]/: The slash image specifier skips all characters up to the next [n] line-feeds.

#: The # image specifier cancels the line-feed terminator, and must appear before any other specifiers in the image list. Data entry terminates with the last item in the list or EOI.

+: The + image specifier cancels the HP-IB EOI terminator, and must appear before any other specifiers in the image list.

%: The % image specifier cancels both line-feed and EOI as terminators, and must appear before any other specifiers in the image list.

[n](...): Image specifiers can be replicated [n] times by enclosing them in parentheses. One or more image specifiers can be replicated within the parentheses.

The ENTER statement defaults to freefield input when no image reference is specified. The default numeric input is explained under the F specification, and the default string input is explained under the T specification. No image specifications need be designated in the statement for freefield data input. The ENTER operation is terminated after the entire list has been satisfied by receipt of a line-feed character (unless a # or % image specifier is included in the image list), or by an EOI (unless a + or % is included in the image list).

The binary image specifiers B Y, and W are normally used for binary input which doesn't contain line-feeds between variables. Use the # image specifier to disable the line-feed terminator when using B, Y, or W for normal binary input.

Some examples of formatted input are shown here to illustrate the syntax usage in a program:

Example 1: 150 numeric variables punched on a paper tape are to be read into the array Q. There are no delimiters between variables, and each variable is eight characters including a decimal radix point:

```
530  OPTION BASE 1
540  DIM Q(150)
541   !  THE IMAGE STRING Img$ SPECIFIES A NUMERIC FORMAT
542   !  OF 8 DIGITS WITH A DECIMAL POINT RADIX SYMBOL.
550  Img$="8N"
560  ENTER 6 USING Img$;Q(*)
570  MAT PRINT Q;
580  END
590   !
```

Example 2: This example is similar to (1) above, except a comma is on the tape as a delimiter between variables:

```
620  OPTION BASE 1
630  DIM Q(150)
631   ! SINCE A COMMA DELIMITS NUMERICS, NO IMAGE SPEC IS NECESSARY.
670  ENTER 6;Q(*)
680  MAT PRINT Q;
690  END
700   !
```

Example 3: A 10-bit analog-to-digital converter presents binary data to the 98032A interface at select code 8. The information is available as a 16-bit word and is entered into the integer array N.

```
730  OPTION BASE 1
740  INTEGER N(15)
741   !  NOTE THAT THE C/R-L/F TERMINATOR IS DISABLED
742   !  BY THE # IMAGE SPECIFIER.
750  IMAGE #,W
760  ENTER 8 USING 750;N(*)
770  MAT PRINT N;
780  END
790   !
```

Example 4: Data is presented as a 3000 byte series of 8-bit values over an HP-IB interface. The data is saved as characters in the string A$.

```
820  DIM A$[3000]
821   !  THE IMAGE REFERENCE Ref SPECIFIES FREEFIELD STRING INPUT
822   !  IGNORING THE L/F TERMINATOR REQUIREMENT.
823 Ref:  IMAGE #,T
830  ENTER 707 USING Ref;A$
840  PRINT LEN(A$);" CHARACTERS HAVE BEEN ENTERED."
850  END
```

The ENTER statement can be used in conjunction with a conversion table to change incoming data from a non-ASCII chararacter code to the ASCII code used in the System 45. This is covered in the section on Conversions found later in this chapter.

The Read Binary Function

The READBIN function returns one word of data from a specified interface or HP-IB device. In the case of an 8-bit interface, the upper 8-bits of the word are set to zero; for a 16-bit interface, 16 meaningful bits are returned.

Syntax:

```
READBIN (sc)
```

sc: The select code parameter may specify either the interface select code (0-12) or the HP-IB address of a single device.

For example, to read a keystroke from the System 45 keyboard, the following program can be run:

```
880   !  READBIN EXAMPLE
890   !
900   !  NOTE THIS PROGRAM REQUIRES A CONTROL-STOP TO TERMINATE.
901 Loop:  REM
910  Keycode=READBIN(0)
920   !  THE VARIABLE Keycode IS THE KEYCODE LAST PRESSED.
930  DISP Keycode;" IS THE CODE FOR A KEYPRESS OF ";CHR$(Keycode)
940  GOTO Loop
950  END
960   !
970   !
```

A keyboard-to-keycode diagram is listed in the appendix for reference when using the READBIN function to capture keystroke codes.

The following statement takes one byte of data from device 9 on the HP-IB interface, select code 7:

```
30   Variable=READBIN(709)          ! TAKE THE BYTE
```

Data Conversions

The I/O ROM provides the capability of automatically converting the ASCII character code of the System 45 to a non-ASCII code for output. Non-ASCII codes can be automatically converted to the System 45's ASCII character code for data input as well.

The Conversion Table Statement

Data being input or output (ENTER and OUTPUT statements only) by the System 45 can be converted to and from non-ASCII code by using the conversion table statement. The CONVERT statement assigns the contents of a pre-dimensioned string variable to the conversion table for input and/or output through the corresponding interface select code. For advanced I/O operations, the CONVERT statement can also be used to convert data to and from program variables. More details concerning this type of operation are given in the Advanced I/O Operations chapter (refer to the section on Variable-to-Variable Transfers).

Syntax:

```
CONVERT sc; "mode" , conversion string [, parity]
CONVERT sc ; "mode" [, conversion string] , parity
CONVERT sc ; "mode" , expression 1 TO expression 2 [, parity]
```

sc: The select code parameter can be any valid interface select code (0-12, 16) or HP-IB device address.

mode: The mode parameter is a string expression that determines the conversion table mode. There are several conversion modes that are specified as follows:

I – The conversion table is generated for input data only.

O – The conversion table generated is for output data only.

IO – Conversions are performed on input and output. The specified conversion table is used for input data conversions, and an inverse conversion table is generated for output data conversions.

OI – Conversions are performed on output and input. The specified conversion table is used for output data conversions, and an inverse conversion table is generated for input data conversions.

conversion string: The conversion string is a string variable used to set up a conversion table to perform indexed data conversion. The numeric value of the character to be converted is used as an index for locating the actual character to be substituted from the conversion table.

parity: When specified, the parity parameter causes parity to be generated for output data and checked on input data. The parity type is specified as follows:

Numeric expression = 0. The parity bit is always cleared (0).

Numeric expression = 1. The parity bit is always set (1).

Even value (not 0) numeric expression. Even parity is used.

Odd value (not 1) numeric expression. Odd parity is used.

expression 1,expression 2: Expressions 1 and 2 may be string expressions that are evaluated to a single character (only the first character is used) or the numeric equivalent of the conversion character.

Expression 1 specifies the character to be converted to the character specified by expression 2. This form of the CONVERT syntax may be used to specify single character conversions. More than one CONVERT...TO...statement may be used to specify conversions for more than one character. Note that a full 256 byte table is generated automatically: unspecified characters are not converted.

Example statements:

```
    CONVERT Sel; "CI", Conv $, 1
```

Conv $ is used to set up the output conversion table, an inverse table is set up for input, the parity bit is always set.

```
    CONVERT 706; "O", Ctbl$,3
```

Ctbl$ is used to set up an output only conversion table for device 706, and odd parity is generated.

```
    CONVERT S; "I", "A" TO "C"
```

Convert the character "A" to the character "C" on input from select code S: ignore parity.

When data is being input, the character's parity bit is checked and stripped from the character if parity is specified; the character is then converted using the conversion table.

For output, the character is first converted using the specified conversion table, then the correct parity bit is generated and added to the converted character for output.

The first character in the conversion table corresponds to a value (the numeric value of the character to be converted) of 1, the second character in the table corresponds to a value of 2, and so on, up to the 255th character. The 256th character corresponds to a numeric value of 0.

If a string with a dimensioned length less than 256 is specified in the CONVERT statement, the remaining characters in the conversion table (those positions not defined by the string) are set to the numeric equivalent of the table index. The effect of this is that non-specified characters are not converted from their internal form.

A simple example of the operation of the conversion table follows:

```
10    IMAGE #,B
20    ! THIS EXAMPLE CAN BE EXECUTED TO THE CRT
30    ! TO SHOW THE MANNER IN WHICH CONVERT WORKS.
40    !
50    A$="ABC"  ! THIS IS THE CONVERSION STRING.
60    ! AN OUTPUT VALUE OF 1 INDEXES TO THE LETTER "A"
70    ! AN OUTPUT VALUE OF 2 INDEXES TO THE LETTER "B".
80    ! AN OUTPUT VALUE OF 3 INDEXES TO THE LETTER "C".
90    ! OTHER OUTPUT VALUES ARE NOT CONVERTED.
100   CONVERT 16;"O",A$
110   OUTPUT 16 USING 10;1 ! AN "A" WILL BE OUTPUT.
120   OUTPUT 16 USING 10;2 ! A "B" WILL BE OUTPUT.
130   OUTPUT 16 USING 10;3 ! A "C" WILL BE OUTPUT.
140   OUTPUT 16 USING 10;75! SINCE THERE IS NO CONVERSION FOR 75
150   !                      A "K" WILL BE OUTPUT.(ASCII VALUE FOR "K")
160   END
```

To illustrate the use of the CONVERT statement, an EBCDIC to ASCII conversion table can be generated for I/O to a terminal on select code 6 by the following sequence of statements:

```
20     !
30     ! THE DATA TABLE IS SET UP TO RETURN:  FIRST, THE POSITION (EQUAL TO THE
         EBCDIC CODE); SECOND, THE  CORRESPONDING ASCII CHARACTER.
40    DIM A$[256]
50    A$=RPT$(CHR$(0),256)
60    DATA 64," ",75,.,76,<,77,(,78,+,79,|,80,&,90,"!",91,$,92,*,93,),94,;
70    DATA 96,_,97,/,106,^,108,%,109,-,110,>,111,?,122,:,123,#,124,@,125,',126,=
80   DATA 129,a,130,b,131,c,132,d,133,e,134,f,135,g,136,h,137,i,145,j,146,k,147,
l,148,m,149,n,150,o,151,p
90   DATA 152,q,153,r,161,~,162,s,163,t,164,u,165,v,166,w,167,x,168,y,169,z,177,
\,178,{,179,},180,[,181,]
100  DATA 193,A,194,B,195,C,196,D,197,E,198,F,199,G,200,H,201,I,209,J,210,K,211,
L,212,M,213,N,214,O,215,P
110  DATA 216,Q,217,R,224,\,226,S,227,T,228,U,229,V,230,W,231,X,232,Y,233,Z,240,
0,241,1,242,2,243,3,244,4,245,5,246,6,247,7,248,8,249,9
120   !
130   !
140  FOR I=1 TO 93
150  READ Ebcdic  ! THIS GETS THE NEXT INDEX POSITION FOR THE ASCII CHARACTER.
160   !            (NOTE THIS IS THE EBCDIC CHARACTER VALUE FOR THE ASCII CHAR.)
170  READ A$[Ebcdic,Ebcdic]  !  NOW GET THE CORRESPONDING ASCII CHARACTER
180  NEXT I
190   !
200  A$[107,107]=","     ! SET UP THE COMMA AS A SPECIAL CASE
210  A$[127,127]=CHR$(34)   ! AND THE QUOTE SIGN
211  A$[37,37]=CHR$(10)   ! SET UP LINE-FEED CONVERSION
212  A$[10,10]=CHR$(13)   ! SET UP CARRIAGE-RETURN CONVERSION
220   !
230   ! FOR INPUT, THE CONVERSION TABLE INDEXES EBCDIC CHARACTER VALUES TO OBTAIN
        THE EQUIVALENT ASCII CHARACTERS.
240    ! FOR OUTPUT, AN INVERSE TABLE IS GENERATED TO INDEX ON ASCII CHARACTER
         VALUES TO OBTAIN THE EBCDIC CHARACTER (WHICH IS OUTPUT INSTEAD OF ASCII)
250    ! NOTE THE PARITY USED FOLLOWS THE VALUE OF THE VARIABLE "P"
260    ! SINCE "P" IS ODD (#1), ODD PARITY IS GENERATED AND CHECKED.
270  P=3   ! SPECIFY ODD PARITY.
280  CONVERT 705;"IO",A$,P
290    !
300    ! NOTE THAT THE STRING A$ IS AVAILABLE FOR USE ELSEWHERE IN THE PROGRAM
         ONCE THE CONVERT STATEMENT IS EXECUTED.
310    !
```

Any output to select code 6 is now converted from ASCII to EBCDIC character code. For instance, the ASCII character "D" has a decimal value of 68 – the 68th character in A$ is the numeric value of an EBCDIC "D", which is 196. When the statement OUTPUT 6; "D" is executed, the actual code output is 196, the binary code for an EBCDIC"D".

A partial conversion table can be specified or a current conversion table can be added to by using the CONVERT...TO...syntax. Any number of CONVERT...TO...statements can be executed, and each conversion character is placed in the conversion table for that select code in the proper position. For example, suppose only the ASCII numbers, comma, decimal, and carriage return are to be converted according to the following table:

Character	ASCII Value(decimal)	EIA[1] Value(decimal)
0	48	32
1	49	1
2	50	2
3	51	19
4	52	4
5	53	21
6	54	22
7	55	7
8	56	8
9	57	25
,	44	59
.	46	107
C/R	10(line feed)	128

[1] EIA = Electronics Industries Association standard

The following program sequence can be used to set up the conversion table:

```
20    !  PROGRAM SEQUENCE TO SET UP AN ASCII-TO-EIA CONVERSION TABLE
30   DATA 48,33,49,2,50,3,51,20,52,5,53,22,55,8,56,9,57,26,44,60,46,108,10,129
40   FOR I=1 TO 12
45    !  READ THE NEXT ASCII CHARACTER VALUE AND THE NEXT EIA CHARACTER VALUE
50   READ K,L
60    ! THE ASCII CHAR. VALUE IS CONVERTED TO THE EIA CHAR. VALUE ON OUTPUT
61    ! THE EIA CHAR. VALUE IS CONVERTED TO THE ASCII CHAR. VALUE ON INPUT
65   CONVERT Sel_code;"OI",K TO L
70  NEXT I
80    ! THE CONVERSION TABLE FOR "Sel_Code" IS NOW SET UP FOR ASCII-EIA NUMERIC
        CHARACTER CONVERSIONS
90    !
100   END
```

Each iteration through the loop (lines 40-70) takes two corresponding conversion values and places them in variables K (ASCII) and L (EIA). The CONVERT statement sets up an output and input conversion table and places the conversion code into the table. On output, the ASCII characters are converted to EIA character codes; on input, incoming EIA character codes are converted to ASCII characters.

The Conversion Disable Statement

Syntax:

```
CONVERT sc ; "mode"
```

Executing the CONVERT statement specifying only the interface select code and the mode parameters cancels data conversions for the specified interface and conversion mode.

Basic Interface Control

The first statement of this section allows you to check the status register(s) of an interface for control of program flow based on the current conditions of the interface. The second statement covered in this section provides you with the capability of resetting an interface card without having to reset the System 45. The final two statements allow you to activate and deactivate I/O through an interface for program debugging purposes.

The Interface Status Statement

The Interface Status statement reads the current status information from the specified interface card and returns a decimal-equivalent number. In addition, up to four interface status bytes can be obtained from a 98034A HP-IB Interface card; the values are placed into the variables specified in the statement syntax. The specified variables can be of type REAL, SHORT, or INTEGER.

Syntax:

```
STATUS sc ; var 1 [, var 2[, var 3[, var 4]]]
```

sc: The select code parameter specifies the select code (0-12) of an interface.

var 1: The value returned in variable 1 corresponds to the status of the interface card. This value is interpreted according to the interface type, and can have a range of 0 to 511.

var 2,3,4: The values returned in variables 2 through 4 correspond to status bytes accessible in the 98034A HP-IB interface card. These variables are not modified when reading the status of any interface other than a 98034A.

```
10    ! EXAMPLES FOR STATUS STATEMENT
11    !
12    !
20    STATUS Hpib;V4,V1,V2,V3


40    !  CHECK THE INTERFACE STATUS; IF THE DEVICE IS DOWN, SET Flag=0
50    STATUS Sel_code;V
60    IF NOT BIT(V,8) THEN Flag=0
70    !
```

The following diagrams show the status words returned from each of the different interface cards.

98032A Interface Status

			Interface Identifier					
BIT 8	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
STS	INT	DMA	1	0	IID	IOD	STI1	STI0

Bit 8: Peripheral Status
Bit 7: Interrupt Enable
Bit 6: Direct Memory Access Enable
Bits 4&5: Interface Identifier. Bit 4 = 0, Bit 5 = 1
Bit 3: Invert Input Data
Bit 2: Invert Output Data
Bit 1: Status bit 1
Bit 0: Status bit 0

98033A Interface Status

			Interface Identifier					
BIT 8	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
STS	INT	0	1	0	0	0	0	0

Bit 8: Peripheral Status
Bit 7: Interrupt Enable
Bit 6: Always 0
Bits 4&5: Interface Identifier. Bit 4 = 0, Bit 5 = 1
Bits 0-3: Always 0

98034A Interface Status

Variable 1:

BIT 8	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
STS	SERVICE REQUEST	CONTROLLER ACTIVE	TALKER ACTIVE	LISTENER ACTIVE	SYSTEM CONTROLLER SET	1	SERIAL POLL SET	END of RECORD

Bit 8: Peripheral Status
Bit 7: Is 1 when the SRQ signal line is true
Bit 6: Is 1 for active controller
Bit 5: Is 1 for active talker
Bit 4: Is 1 for active listener
Bit 3: Is 1 for active system controller
Bit 2: Always 1
Bit 1: Is 1 when a serial poll is in progress
Bit 0: Is 1 when the EOI (end or identify) line is true

Variable 2:

			Interface Identifier				
BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
0	0	0	0	0	DEVICE CLEAR	0	ERROR

Bits 3-7: Always 0
Bit 2: Is 1 when Device Clear received
Bit 1: Always 0
Bit 0: Is 1 when error detected

Variable 3:

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
1	1	0	HP-IB ADDRESS (MSB)				(LSB)

Variable 4:

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
EOI	REN	SRQ	ATN	IFC	NDAC	NRFD	DAV

Logical 1 indicates corresponding HP-IB signal line is true.

98035A Interface Status

Interface Identifier

BIT 8	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
STS	INT	0	1	0	0	0	INTERRUPT FLAG	ERROR FLAG

Bit 8: Peripheral Status
Bit 7: Is 1 when interrupts are enabled by CARD ENABLE
Bits 4&5: Interface Identifier. Bit 4 = 0, Bit 5 = 1
Bit 1: Is 1 when interrupts are enabled; not reset by servicing an interrupt request
Bit 0: Is 1 when an error condition exists

98036A Interface Status

Interface Identifier

BIT 8	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
STS	INT	0	0	1	0	0	RECEIVER INTERRUPT	TRANSMITTER INTERRUPT

Bit 8: Peripheral Status
Bit 7: Interrupt Enable
Bit 6: Not used, always 0
Bits 4&5: Interface Identifier. Bit 4 = 1, Bit 5 = 0
Bits 2&3: Not used, always 0
Bit 1: Is a 1 when the receiver interrupt control is enabled (bit 2 of R5 OUT)
Bit 0: Is a 1 when the transmitter interrupt control is enabled (bit 1 of R5 OUT)

The Interface Reset Statement

This statement resets the specified interface card (not a 98034A type interface) by setting bit 5 of the interface's R5 register.

Syntax:

```
    RESET sc
```

sc: The select code parameter (1-12) specifies the select code of the interface card to be reset.

The RESET statement can also be used with a 98034A interface, however, the results differ from those described here. Refer to the HP-IB Operations chapter for details specific to using the RESET statement with the 98034A interface.

The Select Code Activate/Deactivate Statement

This statement allows you to syntax and "dry run" a program containing I/O statements without actually performing any I/O. All input and output operations are checked for syntax and are executed, although no data transfers actually take place. Input operations (ENTER statement only) on an inactive select code do not affect variables in the enter list. The READBIN and PPOLL functions covered in the Advanced I/O and HP-IB Operations chapters return a zero value (0) from an inactive select code. The IOFLAG and IOSTATUS functions of the same chapter return a one value (1) from an inactive select code.[3]

Syntax:

`SELECT CODE` sc `INACTIVE`

sc: The select code parameter can be any valid interface select code expression (0-13, 16). The specified select code becomes inactive until the System 45 is reset or until activated by the SELECT CODE...ACTIVE statement.

Syntax:

`SELECT CODE` sc `ACTIVE`

select code: The select code parameter can be any valid interface select code expression (0-13, 16). The specified select code is activated; all subsequent I/O transfers are processed in the normal manner.

[3] The reason these particular values are returned is to enable your program to execute as though the inactive select codes are present and operational.

Chapter 3
Advanced I/O Operations

This chapter covers the topics necessary to perform complex or special interfacing tasks. Topics covered include:

- A basic description of programmable interrupts.
- The statements necessary to utilize and control programmable interrupts.
- High speed and specialized I/O operations.
- Inter- and intra- computer communication.
- Overlapped I/O operations
- Advanced interface control operations.

Interrupt I/O Operations

Introduction

The interrupt I/O statements give you control over program flow when I/O transfers are completed or when peripherals request service on an interrupt basis. The On Interrupt statement establishes the priority level for a program branch when an interrupt is received. End-of-line branching for an interface can be disabled by the Off Interrupt statement. The Card Enable statement enables a particular interface card to request end-of-line branches. A fourth statement, the Control Mask statement, is used to set the hardware interrupt mask for the interface.

There are two important distinctions to remember when using interrupts:

1) There is a hardware level interrupt that is transparent to you, the user.[1] For OUTPUT and ENTER, it can be used for data transfers (BINT and WINT transfer types). For the ON INT statement, it can be used for end-of-line branch requests.

[1] Refer to Appendix B for a flowchart illustrating hardware-level interrupt service.

Hardware interrupts demand system attention on two levels, where select codes 8-15 have a higher level of service priority than do select codes 0-7. This essentially means that a device requiring immediate attention should have a select code in the higher range. Devices requiring infrequent attention or that operate at slow or intermittent rates can be set to the lower range (0-7) of select codes.

2) There is a program level interrupt, called an end-of-line branch that "interrupts" normal program execution at the end of a program line. A branch is taken to the interface service routine when the system priority drops below the priority level established for end-of-line branches from the interface. This operation is discussed in the following section, and a flowchart illustrating end-of-line branching is included in the Appendix for reference.

The On Interrupt Statement

An interface card (and its associated peripheral) can cause end-of-line program branching. This is accomplished by the ON INT statement which specifies the branch and the priority level for the branch. With the exception of ON INT...CALL...,any ON INT statement is effective only within the current program segment.

Syntax:

ON INT # sc [, priority] GO TO line ID
ON INT # sc [, priority] GOSUB line ID
ON INT # sc [, priority] CALL name

sc: The select code parameter can be any valid interface select code, 1-12.

priority: The optional priority parameter specifies the priority level (1-15) assigned to end-of-line branches from the specified select code. If a GO SUB or CALL is specified, the system priority level is set to the priority level specified by the ON INT statement, then restored to the previous system priority level upon return from the subroutine or subprogram. End-of-line branches of a higher priority are granted end-of-line service within the interface service subroutine or subprogram, but lower priority branches are not serviced until the system priority level drops lower than the priority level of the requesting interface. The default system priority is 0.

line ID: The line ID parameter specifies the line identifier of the interface service routine or subroutine.

name: The name parameter specifies the name of the interface service subprogram.

GO TO: An ON INT statement specifying a GO TO is active only within the program segment[2] in which it is executed. Transfer to a different program segment temporarily disables the ON INT...GOTO...conditions. When the program segment containing the ON INT...GO TO...statement is re-entered, the ON INT conditions are automatically reinstated. An end-of-line branch associated with an ON INT...GO TO...statement does not alter the system priority level or exit the current program segment.

GO SUB: An ON INT statement specifying a GO SUB is active only within the program segment[2] in which it is executed. Transfer to a different program segment temporarily disables the ON INT...GO SUB...conditions. When the program segment containing the ON INT...GO SUB...statement is re-entered, the ON INT conditions are automatically reinstated. An end-of-line branch associated with an ON INT...GO SUB...statement sets the system priority level to the statement's priority level (the priority parameter) but does not exit the current program segment.

CALL: An ON INT statement specifying a CALL is active regardless of the current program segment. An end-of-line branch associated with an ON INT...CALL...statement remains active until it is redefined by another ON INT statement for the same select code or until it is cancelled by an OFF INT statement for the same select code.

Two system statements, ENABLE and DISABLE affect ON INT end-of-line branches as well as ON KEY end-of-line branches. The DISABLE statement inhibits end-of-line branching until an ENABLE statement is executed. If an interrupt is logged in while end-of-line branches are inhibited, the end-of-line branch for that interrupt is taken when the ENABLE statement is executed.

[2] Refer to the chapter of Subprograms in the System 45 Operating and Programming manual for an explanation of program segments.

Example:

Establish a priority level of 8 for interface select code 4: perform a GO SUB branch to line 8000 when servicing the end-of-line branch and set the system priority level to 8 within the subroutine.

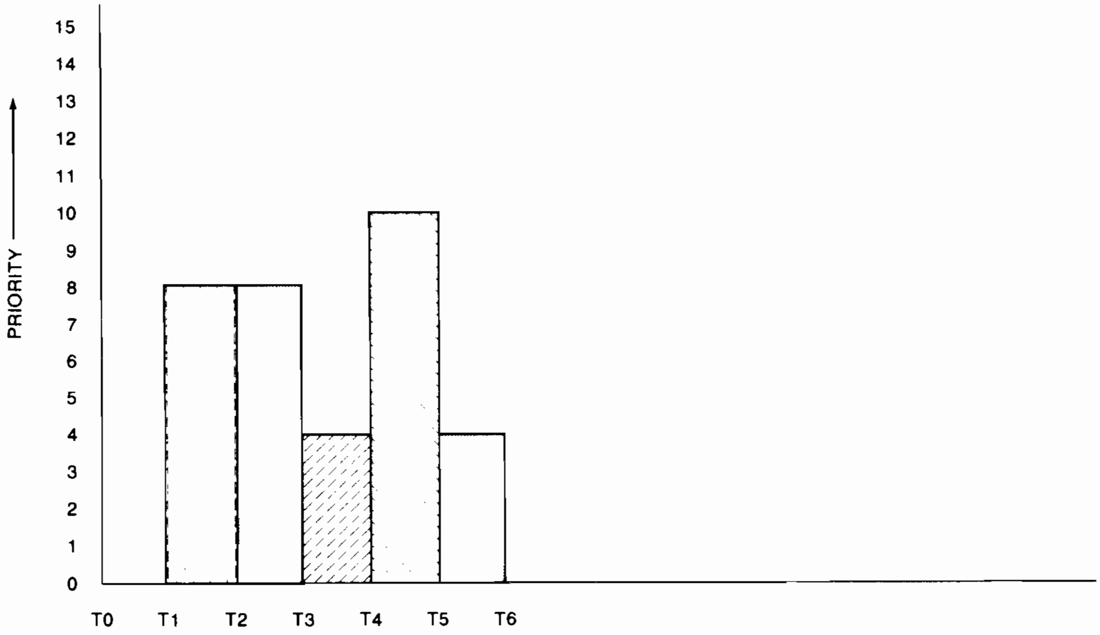
```
10   ON INT #4,8 GOSUB 8000          ! SET PRIORITY TO 8.
20   !    GOSUB 8000 WHEN END-OF-LINE BRANCH IS TAKEN.
```

Example:

Branch to Interrupt when an end-of-line branch request from select code 6 is received: do not reset system priority level.

```
260  !
261  ON INT #6 GOTO Interrupt   ! SET UP END-OF-LINE BRANCH TO Interrupt.
270  CONTROL MASK 6;128         ! ESTABLISH THE ENABLE MASK FOR CARD ENABLE
280  CARD ENABLE 6              ! ENABLE THE INTERFACE CARD FOR END-OF-LINE
290                             ! BRANCH REQUESTS
310  !   PROGRAM EXECUTION BEGINS HERE
320  !
```

The following diagram illustrates the relationship between system priority and end-of-line branch requests.



T_0 — Initially, the system priority is set to 0.

T_1 — At time T_1, an interface request with a higher priority level (8) than the system priority level (0) is received and granted end-of-line service. The ON INT branch is a GO SUB, so system priority is redefined to 8.

T_2 — An interface request with a lower priority (4) than the current system priority (8) is received and logged in.

T_3 — The priority 8 interface service subroutine is completed and system priority is restored to 0; the pending end-of-line branch (priority 4) is granted service, and the system priority defined to level 4.

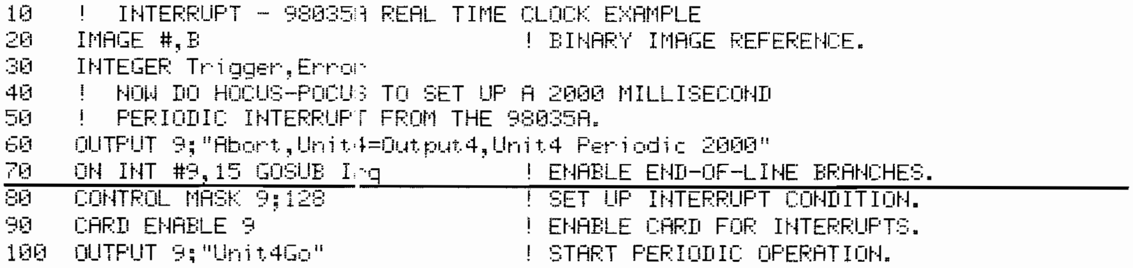
T_4 — A priority 10 interface request is received and granted end-of-line service because it has a higher priority (10) than the current system priority level of 4.

T_5 — The priority 10 interface service routine is completed, and the program continues where left off in the priority 4 interface service routine.

T_6 — The priority 4 service routine is completed, and the system priority drops back to its default value of 0.

Once the interface priority and service routine branch have been defined by the ON INT statement, the interface card must then be enabled to request end-of-line branching. This is accomplished by the CARD ENABLE statement described in a later section.

An example of using the ON INT statement to define end-of-line branch service for the 98035A Real Time Clock, select code 9, follows:



```
10    !  INTERRUPT - 98035A REAL TIME CLOCK EXAMPLE
20    IMAGE #,B                        ! BINARY IMAGE REFERENCE.
30    INTEGER Trigger,Error
40    !  NOW DO HOCUS-POCUS TO SET UP A 2000 MILLISECOND
50    !  PERIODIC INTERRUPT FROM THE 98035A.
60    OUTPUT 9;"Abort,Unit4=Output4,Unit4 Periodic 2000"
70    ON INT #9,15 GOSUB Irq           ! ENABLE END-OF-LINE BRANCHES.
80    CONTROL MASK 9;128               ! SET UP INTERRUPT CONDITION.
90    CARD ENABLE 9                    ! ENABLE CARD FOR INTERRUPTS.
100   OUTPUT 9;"Unit4Go"               ! START PERIODIC OPERATION.
```

```
110 Loop: DISABLE                     ! SECURE TIME READ FROM
120                                   ! END-OF-LINE BRANCHING.
130  OUTPUT 9;"Read time"        ! REQUEST THE TIME STRING.
140  ENTER 9;Time$                    ! INPUT THE TIME STRING.
150  ENABLE                           ! NOW ALLOW ANY PENDING END-
160                                   ! OF-LINE BRANCH.
170  DISABLE                          ! DISABLE END-OF-LINE BRANCHES.
180  OUTPUT 9;"Error"                 ! REQUEST THE ERROR CODE.
190  ENTER 9 USING 20;Error           ! AND INPUT IT (ONE BYTE CODE).
200  ENABLE                           ! RE-ENABLE END-OF-LINE BRANCHES.
210                                   !
220  IF Error=0 THEN No_error         ! IF NO ERROR CONDITION, CONTINUE.
230   BEEP                            ! ELSE SHOW ERROR
240   PRINT "ERROR IN THYME",Error
250 No_error: DISP Time$              ! HERE DISPLAY THE REAL TIME.
260  GOTO Loop                        ! AND LOOP
270                                   !
280                                   !
290 Irq:  REM INTERFACE SERVICE ROUTINE
300  OUTPUT 9;"Trigger"               ! REQUEST THE NUMBER OF THE
310                                   ! INTERRUPTING UNIT.
320  ENTER 9 USING 20;Trigger         ! INPUT ONE BYTE OF INFORMATION.
330  BEEP                             ! INDICATE INTERRUPT RECEIVED.
340  IF BIT(Trigger,3) THEN PRINT "Periodic...",Time$
350                                   !
360  ! NOTE THE INTERFACE CARD IS RE-ENABLED FROM THE SERVICE ROUTINE.
370  CARD ENABLE 9                    ! RE-ENABLE CARD INTERRUPTS.
380                                   !
390  RETURN                           ! AND RETURN TO MAIN PROGRAM.
400  END
```

The Interrupt Disable Statement

The interrupt disable statement, OFF INT, disables the current ON INT condition for a specified select code. The OFF INT interrupt disable is in effect only within its program segment.

Syntax:

OFF INT # select code

select code: Any valid interface select code expression (1-12) may be used. The select code specified is not granted end-of-line service while the OFF INT statement is in effect.

The Set Timeout Statement

The timeout statement specifies a maximum time for the System 45 to wait for an interface to respond to an input or output operation. This prevents a program hang-up on an ENTER or OUTPUT operation if an interface is initially down. If the interface goes down while a default handshake or word handshake input or output operation is in progress, a timeout limit is tested. After the specified time limit is passed, an end-of-line branch is requested and program control is passed to the interface service routine. This is an end-of-line branch, and an ON INT statement must be executed for the interface before a timeout end-of-line branch can occur. If no ON INT statement has been executed and a timeout occurs, the result is an ERROR 163.

Syntax:

```
    SET TIMEOUT sc ; time limit
```

sc: Any valid select code (1-12) may be specified.

time limit: The time limit is a numeric expression specifying the minimum length of time in milliseconds (0-32767) that the System 45 should wait for a response from a device on the associated interface. A time limit of zero (0) disables the timeout end-of-line branch for the given select code.

NOTE

A timeout for INT, FHS, and DMA transfers can occur only if the interface is not ready when the transfer is initiated. If the interface goes "not ready" after the transfer is initiated, the program will remain in the ENTER or OUTPUT statement until the interface again goes ready, allowing the transfer to complete.

The SET TIMEOUT statement is select code dependent; it is associated with the interface specified by the select code parameter of the statement. In addition, a timeout end-of-line branch applies to the ENTER and OUTPUT statements only.

```
10   ! TIMEOUT EXAMPLE
20   ! THIS SUBPROGRAM CHECKS THE END-OF-TAPE
30   ! STATUS OF THE TAPE READER AND SETS A
40   ! FLAG TO INDICATE EOT STATUS.
50   !
60   !
70 Sub_enter:  COM A(150)           ! COMMON ARRAY A.
80   OPTION BASE 1
90   SET TIMEOUT 6;2000             ! SET UP 2 SECOND TIMEOUT.
100  ON INT #6,3 GOSUB Timout       ! SET UP END-OF-LINE BRANCH.
110  Flag=0                         ! CLEAR INPUT ENABLE FLAG.
120  STATUS 6;S
130  IF NOT BIT(S,8) THEN 130       ! WAIT UNTIL READER IS ON.
140  FOR I=1 TO 150
150  IF Flag=1 THEN Exit            ! ENTER ONLY IF Flag IS 0.
160  ENTER 6;A(I)
170  NEXT I
180 Exit:   SUBEXIT
190  !
200 Timout:   IF NOT TIME OUT(6) THEN RETURN
210  STATUS 6;S                     ! CHECK END-OF-TAPE STATUS.
220  Flag=BIT(S,1)                  ! Flag REFLECTS EOT STATUS.
230  IF Flag THEN OUTPUT 6 USING "#,B";32  ! TURN OFF READER IF EOT.
240  RETURN
250  SUBEND
```

The Timeout Interrupt Function

To determine if a timeout is the cause of an end-of-line branch on a particular select code (this would normally be necessary information for the interface service subroutine), the timeout interrupt function can be used.

Syntax:

TIMEOUT sc

The timeout function returns a value of 1 if a timeout has caused an end-of-line branch on the specified "select code", or a 0 for no timeout. If the end-of-line branch to the interface service routine was taken as a result of a timeout and another end-of-line branch condition (SRQ for instance), then the value returned by the timeout function is zero (0).

The timeout function clears the timeout value when it is executed, so the timeout value should be saved in an intermediate variable if it is necessary to access the timeout value more than once per timeout end-of-line branch.

Device Timeout Control

The System 45 device timeout message `DEVICE TIMEOUT ON SELECT CODE` n can be enabled and disabled by using the two device timeout control statements in this section. The SET TIMEOUT statement has no effect on the system device timeout message.

The Timeout Message Disable Statement

This statement disables the system 45 device timeout message. All select codes are affected, so executing the timeout message disable statement effectively turns off the system timeout message for all interfaces.

Syntax:

```
    SYSTEM TIMEOUT OFF
```

The Timeout Message Enable Statement

This statement enables the System 45 device timeout message. All select codes are affected, so executing the timeout message enable statement effectively turns on the system timeout message for all interfaces.

Syntax:

```
    SYSTEM TIMEOUT ON
```

Hardware Interrupt Control

The following statements, CONTROL MASK and CARD ENABLE, provide hardware-level interrupt control for use with ON INT end-of-line branching.

The Control Mask Statement

Certain interface cards can selectively interrupt the System 45 on a hardware level according to the interface conditions enabled by the Control Mask statement. The CONTROL MASK statement is used to specify the interface R5 register user-defined bit configuration for either ENTER and OUTPUT transfers to a 98032A Interface, or end-of-line branch conditions using the CARD ENABLE and ON INT statements. For example, the interrupt conditions for the 98034A HP-IB interface are: device service request, active controller, active talker, active listener, or device clear as shown below.

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
SERVICE REQUEST	CONTROLLER ACTIVE	TALKER ACTIVE	LISTENER ACTIVE	INPUT REGISTER FULL	OUTPUT REGISTER EMPTY	OTHER INTERRUPT CONDITIONS	

Bit 7: Logical 1 enables interrupt on SRQ
Bit 6: Logical 1 enables interrupt on active controller
Bit 5: Logical 1 enables interrupt on active talker
Bit 4: Logical 1 enables interrupt on active listener
Bit 3: Logical 1 enables interrupt on input register full
Bit 2: Logical 1 enables interrupt on output register empty
Bit 1: Logical 1 enables interrupt when error detected, device clear or selective device clear/received (when not active controller)
Bit 0: Enable EOI to clear status line (STS)

Normally the 9845A is the active controller on the bus, and the service request is the only end-of-line branch condition needed from a device. When the 9845A is not the active controller, the remaining branch conditions (such as active talker) are useful to allow the computer to continue processing until the interface requires the computer to take alternate action.

Syntax:

CONTROL MASK sc ; mask

sc: The select code parameter is a numeric expression (1-12) specifying the interface select code setting.

mask: The "mask" parameter is either a numerical expression or a string of 1 and 0 characters that is to be the bit pattern sent to the interface R5 register. The bits set by the Control Mask statement determine the interface hardware interrupt conditions. Up to 16 bits can be specified for the mask parameter.

When using end-of-line branches to service a 98034A Interface, it is important to keep in mind that the interface card is generating the hardware interrupt, not the devices on the bus. The service routine must determine the cause of the end-of-line branch and take appropriate action. The cause of the branch can be analyzed by using the STATUS and BIT functions of the I/O ROM. It is important to remember that before end-of-line branches can be used, branch requests must be enabled (by the ON INT statement) and the interface card must be enabled (CARD ENABLE) as described in the Interrupt section of this manual.

You should be aware that the default control mask value is 0. When a CARD ENABLE statement is executed, a 0 value interrupt mask is sent to the interface unless a CONTROL MASK statement has been previously executed specifying the desired interrupt mask.

An example of the CONTROL MASK statement and an interface service routine follows:

```
340  !
350  ON INT #Sel_code,8 GOSUB Service    ! SET UP PRIORITY AND ROUTINE
360  CONTROL MASK Sel_code;128           ! ENABLE REQUESTS FROM INTERFACE
370  CARD ENABLE Sel_code                ! SEND THE INTERFACE CONT. MASK(128)
380  !
390  !  END-OF-LINE BRANCHES ARE ENABLED AT THIS POINT FOR Sel_code
400  !  PROGRAM WOULD TYPICALLY BE HERE
410  !
420 Service:  STATUS Sel_code;S          ! GET INTERFACE STATUS
430  ! END-OF-LINE INTERFACE SERVICE WOULD TYPICALLY BE HERE
440  RETURN                              ! RETURN FROM SERVICE ROUTINE
450  !
```

The Card Enable Statement

In order for an interface card to initiate a request for an end-of-line branch, the card must first be enabled. The CARD ENABLE statement, in conjunction with the CONTROL MASK statement, provides you with the capability to enable an interface for end-of-line branch operation.

Syntax:

`CARD ENABLE` select code

There are several considerations to keep in mind when you are operating in an end-of-line branch mode:

1. When the card is enabled for hardware interrupts, an interface service routine must already be defined by an ON INT statement.
2. Each time the computer services an end-of-line branch request, all branch requests with a priority less than or equal to the current system priority are locked out until the system priority is reset to a lower level than the pending end-of-line branch.
3. The select code must be reenabled for requests each time a request is serviced, or else end-of-line branches must be enabled within the main program at a point that is executed after the service routine. If this is not done, only one end-of-line branch request will be received and serviced.
4. Branch conditions must be specified by using the CONTROL MASK statement.

   To illustrate two methods of enabling end-of-line branch requests from an interface card, the following two examples enable branches from: 1) The main program, 2) The interface service routine.

Example 1

```
 20   !  EXAMPLE END-OF-LINE SERVICE ROUTINE CARD ENABLE
 30   !  THIS EXAMPLE EXECUTES THE CARD ENABLE FROM THE
 40   !  INTERFACE SERVICE ROUTINE.
 50   !
 60   IMAGE #,B                       ! BINARY IMAGE REFERENCE.
 70   INTEGER Trigger,Error
 80   !  NOW DO HOCUS-POCUS TO SET UP A 2000 MILLISECOND
 90   !  PERIODIC INTERRUPT FROM THE 98035A.
100   OUTPUT 9;"Abort,Unit4=Output4,Unit4 Periodic 2000"
110   ON INT #9,15 GOSUB Irq          ! ENABLE END-OF-LINE BRANCHES.
120   CONTROL MASK 9;128              ! SET UP INTERRUPT CONDITION.
130   CARD ENABLE 9                   ! ENABLE CARD FOR INTERRUPTS.
```

```
140  OUTPUT 9;"Unit4Go"              ! START PERIODIC OPERATION.
150 Loop: DISABLE                     ! SECURE TIME READ FROM
160                                   ! END-OF-LINE BRANCHING.
170  OUTPUT 9;"Read time"        ! REQUEST THE TIME STRING.
180  ENTER 9;Time$                    ! INPUT THE TIME STRING.
190  ENABLE                           ! NOW ALLOW ANY PENDING END-
200                                   ! OF-LINE BRANCH.
210  DISABLE                          ! DISABLE END-OF-LINE BRANCHES.
220  OUTPUT 9;"Error"                 ! REQUEST THE ERROR CODE.
230  ENTER 9 USING 60;Error           ! AND INPUT IT (ONE BYTE CODE).
240  ENABLE                           ! RE-ENABLE END-OF-LINE BRANCHES.
250                                   !
260  IF Error=0 THEN No_error         ! IF NO ERROR CONDITION, CONTINUE.
270   BEEP                            ! ELSE SHOW ERROR
280   PRINT "ERROR IN THYME",Error
290 No_error: DISP Time$              ! HERE DISPLAY THE REAL TIME.
300  GOTO Loop                        ! AND LOOP
310                                   !
320                                   !
330 Irq:  REM INTERFACE SERVICE ROUTINE
340  OUTPUT 9;"Trigger"               ! REQUEST THE NUMBER OF THE
350                                   ! INTERRUPTING UNIT.
360  ENTER 9 USING 60;Trigger         ! INPUT ONE BYTE OF INFORMATION.
370  BEEP                             ! INDICATE INTERRUPT RECEIVED.
380  IF BIT(Trigger,3) THEN PRINT "Periodic...",Time$
390                                   !
400  ! NOTE THE INTERFACE CARD IS RE-ENABLED FROM THE SERVICE ROUTINE.
410  CARD ENABLE 9                    ! RE-ENABLE CARD INTERRUPTS.
420                                   !
430  RETURN                           ! AND RETURN TO MAIN PROGRAM.
440  END
```

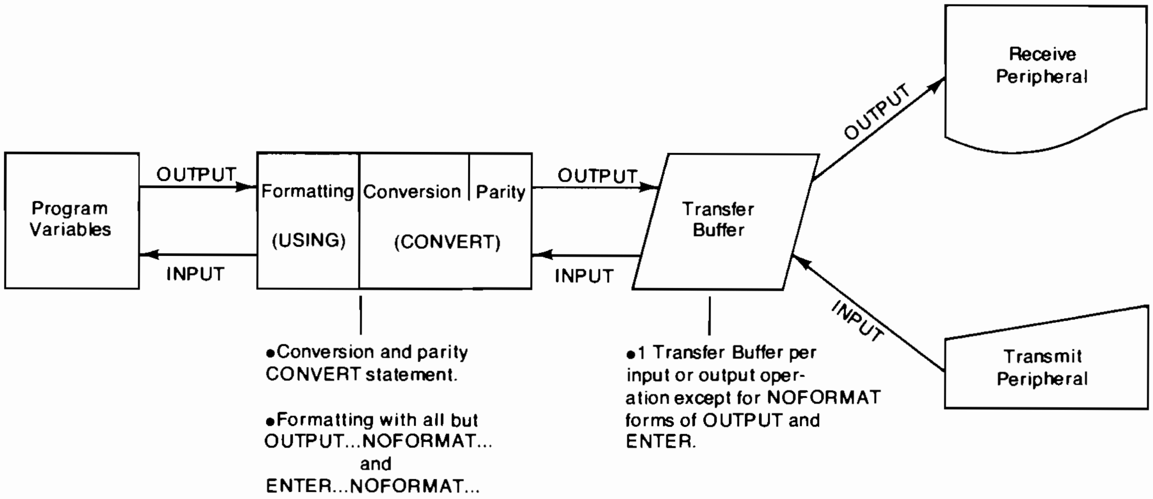
Example 2

```
10   !  THIS EXAMPLE DEMONSTRATES CARD ENABLE FROM THE
20   !  MAIN PROGRAM LOOP.  HP-IB CONTROLLER FUNCTIONS
30   !  ARE PASSED TO DEVICE 22.  THE HP-IB INTERFACE
40   !  IS PROGRAMMED TO INTERRUPT THE SYSTEM 45 WHEN
50   !  CONTROL IS PASSED BACK TO THE SYSTEM 45.  THE
60   !  CARD ENABLE STATEMENT IS PERIODICALLY EXECUTED
70   !  FROM THE MAIN PROGRAM HERE, BUT COULD BE EXECUTED
80   !  FROM THE SERVICE ROUTINE Control AS WELL.
90   !
100  ON INT #7,15 GOSUB Control   ! SET UP END-OF-LINE BRANCHES.
110  CONTROL MASK 7;2^6           ! ENABLE INTERRUPT ON ACTIVE CONTROL.
120  PASS CONTROL 722             ! DEVICE 22 RECEIVES CONTROL.
130  !
140 Loop:  CARD ENABLE 7          ! ENABLE INTERFACE INTERRUPTS FROM
150                               ! MAIN PROGRAM LOOP.
160  STATUS 7;A,B,C,D             ! OBTAIN AND SHOW INTERFACE STATUS.
170  DISP A,B,C,D
180  GOTO Loop                    ! LOOP BACK TO CARD ENABLE.
190  !
200 Control:                      ! INTERFACE SERVICE ROUTINE.
210  BEEP
220  PRINT "RECEIVED CONTROL FROM DEVICE 22."
230  PASS CONTROL 722             ! PASS CONTROL BACK FOR DEMONSTRATION.
240  RETURN                       ! AND RETURN TO MAIN PROGRAM.
250  END
```

Advanced I/O Transfers Operations

Advanced I/O transfers enable the System 45 to transfer data to or from external devices automatically using a variety of transfer modes and data formats. This capability allows the System 45 to communicate with peripherals ranging from very fast to very slow devices with little impact on processing speed. The SERIAL/OVERLAP mode affects the manner in which I/O is handled by the System 45, and these effects are covered in detail in the section on Overlapped I/O.

A diagram of typical I/O operation shows the relationship between program variables, the buffer, and the peripheral:


Program Variables
OUTPUT
INPUT
Formatting
(USING)
Conversion
Parity
(CONVERT)
OUTPUT
INPUT
Transfer Buffer
OUTPUT
INPUT
Receive Peripheral
Transmit Peripheral
•Conversion and parity CONVERT statement.
•Formatting with all but OUTPUT...NOFORMAT... and ENTER...NOFORMAT...
•1 Transfer Buffer per input or output oper- ation except for NOFORMAT forms of OUTPUT and ENTER.

The transfer buffer is a temporary buffer that is established when an ENTER or OUTPUT statement is executed. A separate buffer is established for each I/O operation, and exists until the transfer is completed. The creation of the buffer, and the subsequent I/O operation, are both contingent upon the availability of sufficient R/W memory to establish the buffer. If there is not enough memory to create the transfer buffer, the System 45 must wait for the necessary memory to become available. **This may require completion of a current I/O operation**, and is worth considering for programs having a large data base and performing multiple I/O operations.

Transfer Types

Extremely slow devices or asynchronous input devices (such as an operator controlled digitizer) can present a problem when interfacing to a computer. If the computer spends time waiting for the device to accept or present data, much processing time is wasted. On the other hand, a high speed device (such as a digital voltmeter in burst read) can present data too rapidly for the computer to capture if any processing or formatting is being done on the incoming data. A fast handshake or direct memory access transfer can be used to gather the data as quickly as possible without using transfer time to do processing on the data; this can be done after the data transfer is complete.

By selecting the proper transfer type for the peripheral-computer data transfer, I/O operations can be made more efficient with less impact on the program execution rate of the System 45.

The following table exemplifies some typical applications of transfer types to peripheral devices:

I/O Transfer Applications

Peripheral	Transfer Type
Slow devices:	
9863A Tape Reader	Interrupt
9869A Card Reader	Interrupt
9884A Tape Punch	Interrupt
110 Baud Teleprinter	Interrupt
Asynchronous devices:	
9864A Digitizer	Interrupt
Operator's Console	Interrupt
Medium speed devices:	
9883A Tape Reader	Default Handshake
2640A Terminal (Block)	Default Handshake
High Speed Devices:	
DVM in Burst Read	Fast Handshake or DMA
High Speed HP-IB Devices	Fast Handshake

Upon completion of an interrupt, fast handshake, or DMA transfer, the program branches to the select code service routine if an ON INT statement for that select code has been executed and if the system priority is lower than the select code priority.

The Interrupt Transfer

When an interrupt transfer is specified, the interface is enabled to transfer one byte or word at a time when it is either ready for data or ready with data. Each time the peripheral is ready, a hardware interrupt is sent to the System 45, which then transfers the next word or byte of data. The entire operation continues until the transfer is completed. The computer then signals an end-of-transfer program branch and (if end-of-line branches are enabled by the ON INT statement) the program vectors to the select code service routine according to the priority assigned to the interface.

The Fast Handshake Transfer

When a fast handshake transfer is specified, all input/output capabilities of the System 45 are dedicated to the specified interface for a high-speed transfer. Hardware interrupts are disabled until the transfer is completed. When the fast handshake transfer completes, other I/O activities are again resumed.

NOTE

The computer halts all other I/O operations except any DMA transfers already in progress for the duration of a fast-handshake transfer.

The DMA Transfer

When a DMA (Direct Memory Access) transfer is specified, the hardware DMA channel of the System 45 is assigned to the designated interface. (98034A HP-IB interfaces cannot be used with DMA transfers.) Data is transferred directly to or from memory with no processor intervention.

DMA transfers are the fastest method of input or output available dependent on the type of interface, however, not all interfaces are capable of supporting DMA transfers (refer to the interface's Installation and Service Manual). An additional advantage to DMA transfers is that the computer is free to continue other concurrent I/O operations. Note that when a DMA operation is in progress, no new DMA transfers can be initiated until the current DMA operation is completed.

NOTE

DMA and FHS transfers work only with the negative-true logic setting of the 98032A general purpose interface. Do not use the positive true logic setting, as the data will be complemented for input and output.

The following table shows which transfer types can be used with the different interface cards:

	HS	INT	FHS	DMA
98032	Yes	Yes	Yes	Yes
98033	Yes	Yes	Yes	No
98034	Yes	Yes	Yes	No
98035	Yes	No	Yes	No
98036	Yes	Yes	Yes	No

Output Transfers

Output data transfers can be done by any of 6 methods in addition to the two methods discussed in Chapter 2: byte handshake-on-interrupt (BINT), word handshake-on-interrupt (WINT), byte fast-handshake (BFHS), word fast-handshake (WFHS), byte DMA (BDMA), and word DMA (WDMA). Conversion and parity (the CONVERT statement) generation can be used with all transfer types.

Syntax:

OUTPUT {sc} {type} ; data list
OUTPUT{sc} {type} USING image ; data list

sc: Any valid select code (0-12,16) or HP-IB device addresses can be specified.

image: The image specification is identical to the OUTPUT and PRINT statements' image specification: refer to Chapter 2 of this manual for more detailed information.

type: WHS – specifies a 16 bit output using handshake.

BINT – specifies a byte-by-byte handshake transfer per interrupt to the peripheral interface.

WINT – specifies a word-by-word handshake transfer per interrupt to the peripheral interface.

BFHS – specifies a byte-by-byte fast handshake transfer.

WFHS – specifies a word-by-word fast handshake transfer.

BDMA – specifies a byte-by-byte DMA transfer.

WDMA – specifies a word-by-word DMA transfer

data list: The data list is identical to the OUTPUT or PRINT USING statements' data list.

The following example uses BINT output to print the matrix calculations in lines 100-200. The printer and display list the results of the previous calculations while the System 45 computes the next results:

```
30   OPTION BASE 1
40   DIM Inv(9,9),A(9,9)
50   !
60   ! *** NOTE THAT THE OVERLAP MODE ALLOWS COMPUTATION AND I/O
70   !     TO OCCUR SIMULTANEOUSLY.  TO SEE THE RESULTS OF THIS
80   !     OPERATION, CHANGE THE OVERLAP STATEMENT TO SERIAL.
90   !
100  OVERLAP                            ! SET UP OVERLAP MODE
110  MAT A=CON                          ! INITIALIZE MATRIX  A VALUES
120  MAT Inv=CON                        ! DITTO FOR Inv
130  MAT Inv=TAN(A)                     ! COMPUTE FIRST VALUES
140  MAT Inv=INV(Inv)                   ! INVERT THE MATRIX (ADDED COMP TIME)
150  OUTPUT 16 BINT USING Ref;Inv(*)    ! OUTPUT THE MATRIX TO DISPLAY
160  OUTPUT 16                          ! OUTPUT AN EXTRA BLANK LINE
170  GOTO 130                           ! AND LOOP FOREVER
180 Ref:  IMAGE 9(9(3D.2D,X)/)          ! OUTPUT 9 LINES OF 9 VALUES
190  END
```

Input Transfers

The ENTER statement can be used for input transfers of the following types in addition to the two transfer types discussed in Chapter 2: byte handshake-on-interrupt, word handshake-on-interrupt, byte fast-handshake, word fast-handshake, byte DMA (direct memory access), and word DMA. Conversions and parity (the CONVERT statement) checking can be done on all transfers. Two general syntaxes are used to specify the transfer types as follows:

Syntax 1:

```
ENTER {sc} {type 1} ; list
ENTER {sc} {type 1} USING image ; list
```

Select code (sc), image, and list specifications are identical to the specifications given for the basic ENTER statement syntax of Chapter 2.

type 1:

BINT – specifies a one-byte transfer per interrupt from the peripheral interface.

WINT – specifies a one-word transfer per interrupt from the peripheral interface.

Syntax 2:

```
ENTER {sc} {type 2} {count} ; list
ENTER {sc} {type 2} {count} USING image ; list
```

Select code, image, and list specifications are identical to the specifications given for the basic ENTER statement syntax of Chapter 2.

type 2:

BFHS – specifies a byte-by-byte fast handshake transfer for {count} number of **bytes**.

WFHS – specifies a word-by-word fast handshake transfer for {count} number of **words**.

BDMA – specifies a byte-by-byte DMA transfer for {count} number of **bytes**.

WDMA – specifies a word-by-word DMA transfer for {count} number of **words**.

count: The count parameter specifies the number of bytes or words to be transferred by the ENTER statement. The "type" parameter described above determines whether count specifies a byte count or word count.

Overlapped I/O Operations

The OVERLAP mode of I/O gives the System 45 the capability of executing a program and simultaneously communicating with several peripherals. Because separate processors deal with I/O and program execution, the System 45 uses little program execution time to communicate with peripheral devices. A program written with the overlap capabilities of the System 45 in mind can run much more efficiently than one in which program execution and individual I/O operations are performed sequentially.

On output, once the program variables have been copied to the transfer buffer, the program is free to access the variables. The actual data transfer to the peripheral is handled automatically by the computer whenever the peripheral is ready for new data.

On input, the program can access the input variables only after the data transfer is complete and the data is copied into the program variables. The following list covers concepts that should be considered for maximum OVERLAP mode I/O efficiency:

- ENTER NOFORMAT adds overlap capabilities to input operations. Use ENTER...NOFORMAT to input into a string, then a variable-to-variable ENTER to format the data and copy it into the proper variables.
- Any attempt to access or modify variables currently receiving data causes the program to pause until the transfer is complete. To avoid this pause, use ENTER...NOFORMAT...in conjunction with the ON INT...statement to determine when the transfer has completed. (An end-of-line branch is taken at the end of an INT, FHS, or DMA transfer if branches are enabled.)
- Fast handshake transfers tie up the I/O section of the System 45 for the duration of the transfer. All other I/O operations stop until the fast handshake transfer is complete.

- When overlapped I/O is performed in a program, input and output transfers are done asynchronously; that is, just because one I/O operation is started before another does not ensure that it will be finished before the other. To be sure that key operations are performed in sequence (as is done in SERIAL mode) while taking advantage of performing other tasks in the OVERLAP mode, use the sequential forms of OUTPUT and ENTER explained in the following section.

Sequential I/O with OVERLAP

The OUTPUT and ENTER statements can be specified to force sequential output and input operations in the OVERLAP mode where each specified sequential input and output operation is performed in the order it is executed in the program. Other I/O operations can be operating in the OVERLAP mode. Sequential ENTER or OUTPUT is indicated by adding an "S" to the front of the statement. ENTER statements become SENTER, while OUTPUT statements become SOUTPUT. These statements cannot be executed from the keyboard. Examples of these syntaxes are shown below:

```
30 SENTER 705; A$
40 SOUTPUT 710 BINT USING 99; S, Q, C$
```

Debugging OVERLAP Programs

A program designed to run in OVERLAP mode can be analyzed and debugged by specifying the SERIAL mode. Computer operation is the same for both modes[3], however, if the program runs correctly in the SERIAL mode but not in the OVERLAP mode, then certain input and output operations may need to be performed sequentially. This is easily accomplished by using the SENTER and SOUTPUT syntaxes for the suspect input and output operations.

For instance, say a stimulus level value must be sent before a response input can be taken. In this case, an OUTPUT to a programmable power source must be sent before taking a reading from a digital voltmeter. In the OVERLAP mode, this sequence cannot be guaranteed, so you need to use the sequential OUTPUT and ENTER statements to ensure that the correct sequence is followed.

[3] The exception to this is error trapping (ON ERROR...). Errors resulting from an interface failure (such as ERROR 163) cannot be trapped in the OVERLAP mode of execution. Other errors (such as ERROR 150, select code out of range) can be trapped in the OVERLAP mode.

Unformatted I/O Operations

Introduction

When it is necessary to transfer data at the maximum possible rate, to achieve overlapped data input, or to have random access of transfer buffers, the NOFORMAT mode of the OUTPUT and ENTER statements can be used.

For communications between identical computers, more time-efficient data transfers are possible if each computer spends little or no time formatting data being sent or deformatting data being received. When input and output data transfers are executed in the NOFORMAT mode, data is sent in its internal binary form and received as an internal binary form. Because data is assumed to be in internal form when entered in the NOFORMAT mode, and because no parity or error checking is performed, it is possible to generate a system error by entering incorrect data. Numeric data may be incorrect when it has not been previously output in the NOFORMAT mode.

CAUTION

DATA CAN BE LOST IF IT IS IMPROPERLY ENTERED INTO NUMERIC VARIABLES USING THE NOFORMAT SYNTAX OF THE ENTER STATEMENT. ONLY USE NOFORMAT INPUT WITH STRING VARIABLES OR WHEN THE NUMERIC DATA IS IN THE PROPER INTERNAL FORMAT.

Unformatted I/O is an advanced form of data transfer and requires a good understanding of the procedures and precautions necessary to use with this type of operation. Be sure you read this section thoroughly before attempting unformatted I/O operations.

System 45 Internal Formats

Numeric data is represented internally in a binary-coded-decimal (BCD) format consisting of the exponent[4], the mantissa sign bit, and 6 bytes for the mantissa. Short precision variables differ only in the number of bytes used to represent the decreased accuracy: the differences are illustrated on the next page.

[4] The form of the signed exponent is two's complement binary.

FULL PRECISION

Word 1	Signed Exponent 10 Bits		Not Used 5 Bits	Sign Bit 1 Bit
Word 2	Digit 4 Bits	Digit 4 Bits	Digit 4 Bits	Digit 4 Bits
Word 3	Digit 4 Bits	Digit 4 Bits	Digit 4 Bits	Digit 4 Bits
Word 4	Digit 4 Bits	Digit 4 Bits	Digit 4 Bits	Digit 4 Bits

SHORT PRECISION

Word 1	Signed Exponent 7 Bits	Sign Bit 1 Bit	Digit 4 Bits	Digit 4 Bits
Word 2	Digit 4 Bits	Digit 4 Bits	Digit 4 Bits	Digit 4 Bits

INTEGER

Most Significant Byte 8 Bits	Least Significant Byte 8 Bits

String data is packed two characters per word, each character being represented by one byte of binary data. For example, the string LOVELAND is represented as follows:

STRING

Word 1	L 8 Bits	O 8 Bits
Word 2	V 8 Bits	E 8 Bits
Word 3	L 8 Bits	A 8 Bits
Word 4	N 8 Bits	D 8 Bits

Unformatted output sends out every byte of a variable or array in its internal format. For full precision numeric variables, this means the 10 bit signed exponent, 5 unused bits, the mantissa sign bit, and 6 bytes of BCD digits. Short precision variables are similarly output, but fewer bytes of information are needed to represent the variable, as shown above. String variables are simply output and input in their internal format (two bytes of character data per word).

Unformatted input does not make use of an intermediate buffer for data transfer: the data is placed directly into the variable for an input transfer making the transfer rate considerably faster than formatted input. A variable cannot be modified while an unformatted input on that variable is in progress, and attempting to modify the variable causes the program to pause until the unformatted transfer is complete. The program can then alter the variable and continue normal execution. Conversions and parity checking are not performed for NOFORMAT transfers.

Unformatted I/O Syntaxes

Syntax 1:

OUTPUT sc [type] NOFORMAT ; list

sc: The select code parameter specifies a valid interface select code.

type: The valid output transfer types are the same as for buffered output –

BINT, WINT, BFHS, WFHS, BDMA, WDMA, WHS, and default handshake.

list: Allowable items in the output list are numeric and string variables, string array elements, and numeric arrays. Entire string arrays such as A$(*) are not allowed. Only one item may be in the list for DMA and FHS transfers.

Syntax 2:

ENTER sc {type 1} NOFORMAT; enter list
ENTER sc {type 2} {count} NOFORMAT; enter list

sc: The select code parameter specifies a valid interface select code.

type 1: Valid type 1 transfers are:
WHS, BINT, WINT, and default handshake.

type 2: Valid type 2 transfers are:
BFHS, WFHS, BDMA and WDMA.

count: The count parameter specifies the number of bytes (BFHS and BDMA) or words (WFHS and WDMA) to be transferred.

enter list: The enter list can include any valid program variables, but is limited to one item for DMA and FHS transfers. Entire string arrays (A$(*)) are not allowed.

Note that BDMA and WDMA transfers are equivalent transfers for NOFORMAT input. One word (16 bits) at a time is transferred from the interface during DMA transfers using NOFORMAT.

For example, NOFORMAT input is used to enter data from a digitizer on select code 706:

```
460  !
470  OVERLAP                           ! 9874A DIGITIZER EXAMPLE
480  DIM A$(2000)                      ! SET UP INPUT DATA STRING
490  B$="CN"                           ! ASSIGN CONTINUOUS MODE COMMAND TO B$
500  Digitizer=706                     ! ASSIGN SELECT CODE
510  OUTPUT Digitizer;B$               ! PUT DIGITIZER IN CONTINUOUS MODE
520  ! *** SET UP A SERVICE ROUTINE FOR INPUT TRANSFER COMPLETION
530  ON INT #Digitizer,2 GOSUB Service  ! SET UP PRIORITY AND ROUTINE
540  ENTER Digitizer BINT NOFORMAT;A$   ! INITIATE INPUT TRANSFER
550  ! THE PROGRAM COULD BE INTERACTING WITH THE OPERATOR
560  ! FROM THIS POINT.  WHEN THE TRANSFER IS COMPLETED,
570  ! ANOTHER TRANSFER COULD BE INITIATED, THE OLD DATA
580  ! PROCESSED, AND OPERATOR INTERACTION RESUMED.
590  ! ....
600 Service:  ! THE ROUTINE TYPICALL WOULD RE-INITIATE THE
610  ! DATA TRANSFER , PROCESS THE OLD DATA AND SAVE THE RESULTS.
620  RETURN
```

Variable-to-Variable Transfers

Variable-to-variable transfers can be used to add data conversion, parity generation, and parity checking capabilities to unformatted input and output. For instance, using unformatted input (ENTER...NOFORMAT...) into a string enables simultaneous computation and I/O to other peripherals. When the unformatted input operation is complete, the data can be formatted from string data to numeric data. Character conversions and parity checking can then be performed by using the variable-to-variable ENTER statement. The example at the end of this section illustrates this.

Syntax:

```
ENTER var [BYTE] ; enter list
ENTER var [BYTE] USING image ; enter list
```

var: A valid string variable or string array element can be specified as the data source for the ENTER syntaxes listed. A numeric array must have its starting point specified, i.e., A (1). A(*) or A$(*) is not allowed.

BYTE: The optional keyword BYTE specifies a data transfer from a string or array containing a single byte (lower byte) of information per word. If BYTE is not specified, the transfer takes two bytes of data per word from the source string or array.

enter list: The enter list may consist of any combination of numeric or string variables, string arrays, or array identifiers. Items in the list are separated by commas or semicolons.

In this type of operation, the source of input data is a program variable rather than an external peripheral. Operation is identical to input from an external peripheral, and can be performed in conjunction with data conversions and parity checking.

Variable-to-variable output transfers transfer data to a program variable rather than to an external peripheral. As with the output transfers to a peripheral, data conversion, and parity checking can be used.

Syntax:

OUTPUT var [BYTE] USING image; data list

var: A valid string variable or string array element can be specified as the data destination for the OUTPUT syntaxes listed. A numeric array must have its starting point specified, i.e., A (1). A(*) or A$(*) is not allowed.

BYTE: The optional keyword BYTE specifies a data transfer of one byte (lower byte) per word to the string variable or array specified. If BYTE is not specified, the transfer places two bytes of data in each word of the destination string or array.

data list: The data list may consist of any of the following, separated by commas or semicolons:

variable names

array identifiers

numeric expressions

string expressions

For example:

```
660  !
670  ENTER Sel_code NOFORMAT;A$          ! INITIATE OVERLAPPED INPUT
680  ENTER A$;A(*)                       ! PLACE FORMATTED DATA INTO ARRAY
690  !


700  !
710  ENTER Sc1 BINT NOFORMAT;A$          ! INPUT DATA (OVERLAPPED) FROM
720  OUTPUT Sc2 BINT NOFORMAT;A$         ! OUTPUT DATA TO DESTINATION Sc2
730  !
740  !
```

Advanced Interface Control

The advanced interface control statements of this section alow direct transfer of data or status information between the System 45 and the interface registers R4-R7. Note that because these operations are fundamental I/O operations, you must completely understand the function and protocol of each interface card and peripheral device; if not, you may experience unwanted or unexpected results!

CAUTION

UNEXPECTED OR UNDESIRABLE RESULTS CAN OCCUR WITH THE WRITEIO STATEMENT IF YOU DO NOT FULLY UNDERSTAND THE PROTOCOL AND OPERATION OF THE DEVICE BEING ADDRESSED.

The Write Interface Statement

This statement outputs a 16 bit binary value to the specified register of the addressed interface. This statement is provided to help you tailor your own interface driver or to program a special interface card, but exercise appropriate caution when using this statement.

Syntax:

WRITE IO sc, register; output expression

sc: The select code parameter specifies the select code (0-12) of an interface.

register: The register parameter specifies the desired interface register 4 through 7.

output expression: The output expression is a numeric expression that is converted as necessary to a 16 bit integer and sent to the specified interface register. Specific register information is available in the appropriate interface installation and service manual.

Interface Register Assignments:

- R4: Primary data register
- R5: Primary status/control register
- R6: Secondary data register
- R7:Secondary status/control register

Be sure to test the condition of the flag line (FLG) of the interface to be equal to 1 before attempting a WRITEIO operation. An example follows the explanation of the IOSTATUS function.

The Read Interface Statement

This statement inputs the 16 bit binary value of the specified interface register, leaving the value in the specified variable. The specified variable can be of type REAL, SHORT, or INTEGER.

Syntax:

READ IO sc, register; input variable

sc: The select code parameter specifies the select code (0-12) of an interface.

register: The register parameter specifies the desired interface register. (4-7)

input variable: The 16 bit value of the specified interface register is stored into the input variable. The input variable can be an integer or full or short precision real variable.

Be sure to test the condition of the flag line (FLG) of the interface to be equal to 1 before attempting a READIO operation. (Use the IOFLAG function described in the next section.)

The I/O Flag Function

The I/O flag function returns the current state of the interface flag (FLG) line. The possible values are 0 and 1 with meanings of peripheral busy and peripheral ready, respectively. If the select code has been made inactive, the value returned by the I/O flag function is 1.

Syntax:

```
IOFLAG (sc)
```

sc: The select parameter specifies the select code (0-12) of an interface.

Example:

```
41   ! SHOW THE STATE OF THE FLG LINE OF THE INTERFACES.
50   FOR Sc=1 TO 12
60   Flg=IOFLAG(Sc)
70   PRINT "FLAG LINE, SELECT CODE";Sc;" = ";Flg
80   NEXT Sc
90   END
```

The I/O Status Function

The I/O status function returns a 1 or 0 to indicate the status of the specified interface status (STS) line. A 1 indicates the peripheral is operational; a 0 indicates an error or exceptional condition. If the select code has been made inactive, the value returned by the I/O status function is 1.

Syntax:

```
IOSTATUS (sc)
```

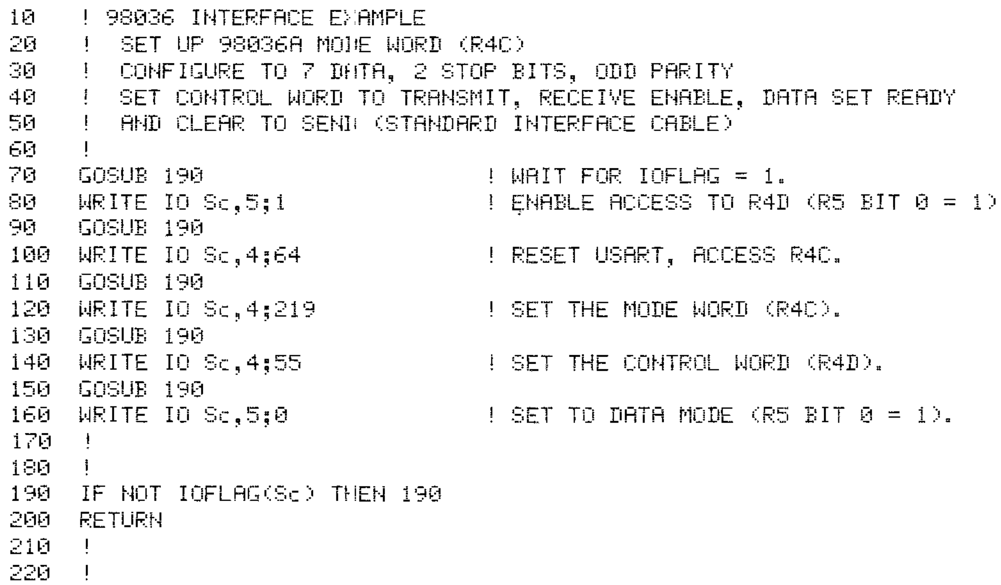
sc: The select code parameter specifies the select code (0-12) of an interface.

Refer to the appropriate interface installation and service manual for details specific to the particular interface.

NOTE

Executing the IOFLAG or IOSTATUS functions from the keyboard with a select code parameter equal to the PRINTER or the PRINTALL device select code results in an error condition (ERROR 38) and should not be attempted.

An example of a serial interface (98036A) I/O driver follows:



```
10    ! 98036 INTERFACE EXAMPLE
20    !  SET UP 98036A MODE WORD (R4C)
30    !  CONFIGURE TO 7 DATA, 2 STOP BITS, ODD PARITY
40    !  SET CONTROL WORD TO TRANSMIT, RECEIVE ENABLE, DATA SET READY
50    !  AND CLEAR TO SEND (STANDARD INTERFACE CABLE)
60    !
70    GOSUB 190                    ! WAIT FOR IOFLAG = 1.
80    WRITE IO Sc,5;1              ! ENABLE ACCESS TO R4D (R5 BIT 0 = 1)
90    GOSUB 190
100   WRITE IO Sc,4;64             ! RESET USART, ACCESS R4C.
110   GOSUB 190
120   WRITE IO Sc,4;219            ! SET THE MODE WORD (R4C).
130   GOSUB 190
140   WRITE IO Sc,4;55             ! SET THE CONTROL WORD (R4D).
150   GOSUB 190
160   WRITE IO Sc,5;0              ! SET TO DATA MODE (R5 BIT 0 = 1).
170   !
180   !
190   IF NOT IOFLAG(Sc) THEN 190
200   RETURN
210   !
220   !
```

Chapter 4
Binary Operations

This chapter covers the binary operations of the System 45 I/O ROM. Topics covered include:

- A brief explanation of binary information as it is represented in the System 45.
- The logical operations AND, OR, exclusive OR, COMPLEMENT, ROTATE, SHIFT, and bit test.

Introduction

The I/O ROM provides the System 45 computer with the capability of manipulating integer data on a bit-by-bit basis. Binary functions available include rotate, shift, binary 'and', binary 'or', exclusive or, complement, and bit test. The binary functions operate on data as a 16 bit word with an integer value[1] between − 32768 and 32767.

Any type of number or numeric expression can be given as an argument to a binary function, however, only the integer value of the argument is used. The fractional part of any argument is rounded and the value returned by a binary function is an integer value between − 32768 and 32767.

[1] All values referred to in this chapter are decimal values unless otherwise noted.

System 45 Binary Representation

The smallest unit of information that can be operated on by the System 45 (or any digital computer) is the **bit**. A bit can assume a value of "1" or "0". The next larger unit of information is the **byte**, which is a package of eight bits. A byte can assume integer values of 0 through + 255 unsigned[1] or − 128 to + 127 signed[1] binary.

All of the binary functions in this chapter operate on a **word**. A binary word is sixteen bits, or two bytes. A word can represent integer values from 0 to 65535 unsigned[2] or − 32768 to + 32767 signed[2] binary.

Binary numbers on the System 45 are internally represented in 16 bits of two's complement binary format. This allows the System 45 to have positive **and** negative binary numbers. There are still 65,536 possible numbers, but half of them are negative (− 32,768 to − 1) and half are positive (0 to 32,767).

The Binary AND Function

The Binary AND function ANDs the two arguments[3] bit-by-bit and returns an integer result.

Syntax:

```
BINAND (expression A, expression B)
```

The truth table for the logical AND function is given on the right.

		Exp. A	
		0	1
Exp. B	0	0	0
	1	0	1

The following example demonstrates the binary AND of two numbers, 45 and 15.

```
30   A=BINAND(45,5)
40   PRINT "A= ";A
```

```
A=  5
```

[2] Unsigned means that there are no negative values possible; signed binary values reserve the top bit to represent the sign (+ or −) of the binary value.

[3] See the introduction to this chapter for the description and limitations of binary arguments.

The Inclusive OR Function

The Inclusive OR function ORs the two arguments[4] bit-by-bit and returns an integer result.

Syntax:

```
BINIOR (expression A, expression B)
```

The truth table for the Inclusive OR function is given on the right.

		EXP. A	
		0	1
Exp. B	0	0	1
	1	1	1

The following example demonstrates the Inclusive OR function of the numbers 45 and 112:

```
80   A=BINIOR(45,112)
90   PRINT "A= ";A
```

```
A=  125
```

The Exclusive OR Function

The Exclusive OR function performs an Exclusive OR on the two arguments[4] bit-by-bit and returns an integer result.

Syntax:

```
BINEOR (expression A, expression B)
```

The truth table for the Exclusive OR function is given on the right.

		Exp. A	
		0	1
Exp. B	0	0	1
	1	1	0

The following example demonstrates the Exclusive OR function of the numbers 45 and 12.

```
130  A=BINEOR(45,12)
140  PRINT "A= ";A
```

```
A=  33
```

[4] See the introduction to this chapter for the description and limitations of binary arguments.

The Complement Function

The complement function returns the one's complement value of the 16 bit argument[5].

Syntax:

BINCMP (expression A)

The one's complement of a number is obtained by changing all zeroes to ones and all ones to zeroes.

The following example takes the complement of the two numbers 45 and 12:

```
 180  A=BINCMP(45)
 190  B=BINCMP(12)
 200  PRINT "A= ";A,"B= ";B
```

```
                A= -46              B= -13
```

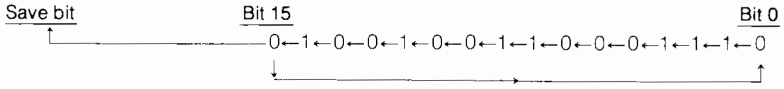
The Rotate Function

The ROTATE function returns the value obtained by rotating the first argument[5] by the number of bits specified by the second argument[5]. A positive value for the second argument rotates to the right, a negative value rotates to the left. The state of the last bit rotated out of the first expression can be determined by the LASTBIT function.

Syntax:

ROTATE (expression A, expression B)

Expression A is rotated to the right or left by (expression B) number of bits. The lastbit is set to the value of the last bit rotated out of the word. As expression A is rotated, bits being rotated out of the word are rotated around to the opposite end of the word, i.e., when a right rotate is executed bit 0 is rotated into bit 15, and vice-versa for a left rotate. For example, a left rotate looks like this:



[5] See the introduction to this chapter for the description and limitations of binary arguments.

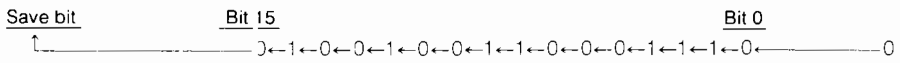
The following example rotates ·45 first by + 12 bits (to the right) then by − 12 bits (to the left).

```
240  Right=ROTATE(45,12)
250  Left=ROTATE(45,-12)
260  PRINT "LEFT= ";Left,"?IGHT= ";Right
```

```
                    LEFT= -12286        RIGHT=  720
```

The Shift Function

The SHIFT function returns the value obtained by shifting the first argument[6] right or left for the number of bits specified by the second argument[6]. A positive value for the second argument specifies a right shift while a negative value specifies a left shift. The lastbit reflects the state of the last bit shifted out of the word, and as bits are shifted out of the word, zeroes are shifted into the opposite side of the word. For example, a left shift looks like this:



Syntax:

SHIFT (expression A, expression B)

Expression A is shifted to the right or left by (expression B) number of bit positions. The last bit shifted out of expression A can be accessed by the LASTBIT function.

The following example shifts 45 by + 12 bits (to the right) and by − 12 bits (to the left).

```
300  Right=SHIFT(45,12)
310  Left=SHIFT(45,-12)
320  PRINT "LEFT= ";Left,'RIGHT= ";Right
```

```
                    LEFT= -12288        RIGHT=  0
```

[6] See the introduction to this chapter for the description and limitations of binary arguments.

The LASTBIT Function

The LASTBIT function returns the last bit shifted or rotated out of the specified word. The value returned is a "1" or "0".

Syntax:

```
LASTBIT
```

For example, the following program prints out the bits of the variable A as they are shifted out.

```
360  A=45
370  FOR I=1 TO 16
380  A=SHIFT(A,-1)
390  Z=LASTBIT
400  PRINT Z;
410  NEXT I
```

```
0  0  0  0  0  0  0  0  0  0  1  0  1  1  0  1
```

The BIT Function

The BIT function tests the condition of **one or more** bits of the argument[7] against the bits of the mask and returns a "1" or "0" depending on the match of the mask with the argument.[7] The bit function can test for a one, zero, or "don't care" state on any particular bit or bits of a word.

Syntax:

```
BIT (expression A, expression B)
```

There are two special cases for the different values expression B may assume:

1. Where expression B is a numerical expression, the BIT function tests only one bit (0-15). See example 2.
2. Where expression B is a string variable or literal representing a binary number, the BIT function returns a "1" only when **all** bits of the mask and the argument match; however, any characters of the mask not equal to a "1" or "0" represent a "don't care" state. A typical mask might look like: `001XXX100X01100X`, where the `X` is a "don't care" bit. (See example 1.) A mask specifying less than 16 bits is right justified, with the undefined bits represented as "don't care" bits. "Don't care" bits have no effect on the result of the BIT function.

[7] See the introduction to this chapter for the description and limitations of binary arguments.

Example 1

The following example tests the integer 45 for the bit pattern 0000xxxx00101xx1

```
460  A=BIT(45,"0000xxxx00101xx1")
470  PRINT "RESULT OF BIT MASK TEST = ";A
```

```
                         RESULT OF BIT MASK TEST = 1
```

Example 2

This example tests the integer 45 for a bit set in the 2^5 bit position:

```
520  A=BIT(45,5)
530  PRINT "RESULT OF BIT 2^5 TEST ON 45 = ";A
```

```
                    RESULT OF BIT 2^5 TEST ON 45 =  1
```

Chapter 5
HP-IB Operations

The System 45 I/O ROM provides the necessary statements and functions for complete control of HP Interface Bus (HP-1B) systems. Bus messages, data, device control, serial and parallel poll, and bus managerıent functions are available and described in full detail in this chapter.

Introduction

The HP-IB is Hewlett-Packard's implementation of IEEE standard 488-1975. A copy of the IEE-488-1975 standard (HP-IB) can be ordered from the IEEE Standards Office; 345 East 47th Street; New York, N.Y. 10017.

The System 45 has the capability of being the system controller, active controller, or a device. As a device, the System 45 can transmit the Require Service message and automatically respond to serial and parallel polls.

Bus sequences and operations for each command are summarized and listed in Appendix C to this manual. These are descriptions intended primarily for reference or detailed bus analysis. It is not necessary to understand the bus in such detail to use it, as there should be enough information in this chapter to allow you to successfully accomplish your interfacing tasks with the System 45 and the HP-IB.

An HP-IB Overview

The HP Interface Bus (HP-IB) provides an interconnecting channel for data transfer between devices on the HP-IB.

The following list defines the terms and concepts used to describe HP-IB (bus) system operations.

HP-IB System Terms

1. Addressing – The characters sent by a controlling device specifying which device sends information on the bus and which device(s) receives that information.
2. Byte – A unit of information consisting of 8 binary digits (bits).
3. Device – Any unit that is compatible with the IEEE Standard 488-1975.
4. Device Dependent – A response to information sent on the HP-IB that is characteristic of an individual device's design, and may vary from device to device.
5. Operator – The person that operates either the system or any device in the system.
6. Polling - The process typically used by a controller to locate a device that needs to interact with the controller. There are two types of polling:

- Serial Poll – This method obtains one byte of operational information about an individual device in the system. The process must be repeated for each device from which information is desired.
- Parallel Poll – This method obtains information about a group of devices simultaneously.

More detailed information on HP-IB operations is listed in Appendix C of this manual.

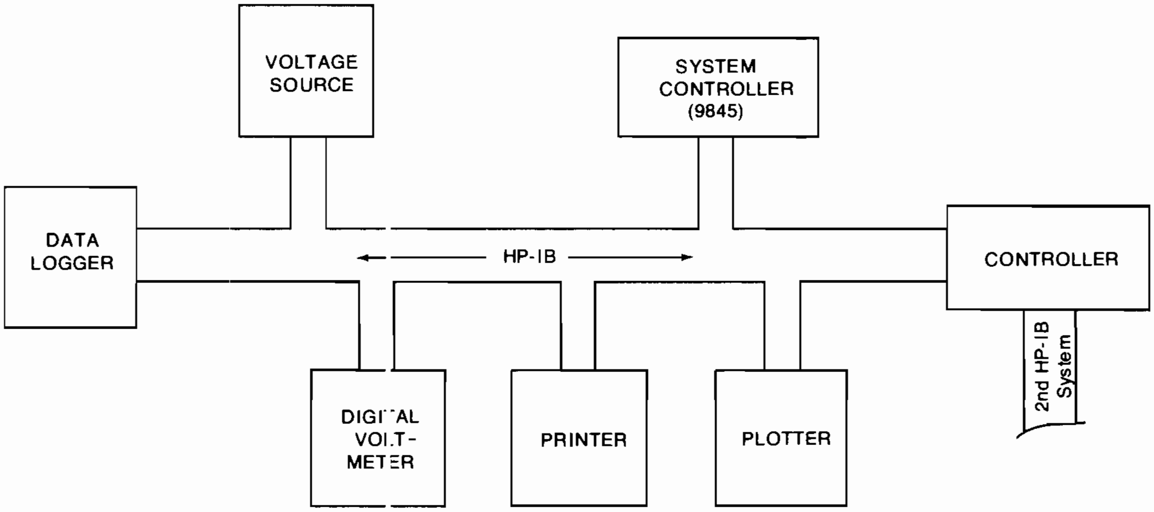
Interface Bus Concepts

Devices which communicate along the interface bus can be classified into three basic categories:

1. Talkers – Devices which send information on the bus when they have been addressed.
2. Listeners – Devices which receive information sent on the bus when they have been addressed.
3. Controllers – Devices that can specify the talker and listeners for an information transfer. Controllers can be categorized as one of two types:

- Active Controller – The current controlling device on the bus. Only one device can be the active controller at any time.
- System Controller – The only controller that can take priority control of the bus if it is not the current active controller. Although each bus system can have only one system controller, the system can have any number of devices capable of being the active controller.

A typical HP-IB System is shown below.



Message Concepts

Devices which communicate along the interface bus are transferring quantities of information. The transfer of information can be from one device to another device, or from one device to more than one device. These quantities of information can easily be thought of as "messages".

In turn, the messages can be classified into twelve types. The HP System 45 Desktop Computer is capable of implementing all twelve of the interface messages. The list below gives the twelve message types for the HP-IB.

1. The Data Message. This is the actual information which is sent from one talker to one or more listeners along the interface bus.
2. The Trigger Message. This message causes the listening device(s) to perform a device-dependent action when addressed.

3. The Clear Message. This message causes either the listening device(s) or all of the devices on the bus to return to their predefined device-dependent states.
4. The Remote Message. This message causes listening devices to switch from local front-panel control to remote program control when addressed to listen.
5. The Local Message. This message clears the Remote Message from the listening device(s) and returns the device(s) to local front-panel control.
6. The Local Lockout Message. This message prevents a device operator from manually inhibiting remote program control.
7. The Clear Lockout/Local Message. This message causes all devices on the bus to be removed from Local Lockout and revert to Local. This message also clears the Remote Message for all devices on the bus.
8. The Require Service Message. A device can send this message at any time to signify that the device needs some type of interaction with the controller. This message is cleared by sending the device's Status Byte Message if the device no longer requires service.
9. The Status Byte Message. A byte that represents the status of a single device on the bus. Bit 6 indicates whether the device sent a Require Service Message, and the remaining bits indicate operational conditions defined by the device. This byte is sent from a talking device in response to a serial poll operation performed by a controller.
10. The Status Bit Message. A byte that represents the operational conditions of a group of devices on the bus. Each device responds on a particular bit of the byte thus identifying a device-dependent condition. This bit is typically sent by devices in response to a parallel poll operation.

    The Status Bit Message can also be used by a controller to specify the particular bit and logic level that a device will respond with when a parallel poll operation is performed. Thus more than one device can respond on the same bit.
11. The Pass Control Message. This transfers the bus management responsibilities from the active controller to another controller.
12. The Abort Message. The system controller sends this message to unconditionally assume control of the bus from the active controller. This message teminates all bus communications (but does not implement a Clear Message).

These messages represent the full implementation of all HP-IB system capabilities. Each device in a system may be designed to use only the messages that are applicable to its purpose in the system. It is important for you to be aware of the HP-IB functions implemented on each device in your HP-IB system to ensure the operational compatibility of the system. (The IEEE equivalent descriptions of these messages may be found in Appendix C.)

Addressing the HP-IB

Every message sent on the HP-IB must specify the originator (talker) and the receiver (listener) of the message. The talker and listener specifications may be implicit or explicit. If the bus is left addressed from a previous transfer, the talker/listener specifications are implicit. Explicit talker/listener specifications require the talker and listener addresses to be sent as part of the transfer. For most messages, the controller (generally the System 45) specifies the device sending the message (talker) and the device(s) receiving the message (listener).

There are four methods of addressing the HP-IB interface and bus devices, and each has a different effect on the manner in which bus addressing is handled. These addressing modes are described here and apply to the statements and functions in this chapter as well as to applicable statements and functions elsewhere in this manual (such as READBIN, CONVERT and EOL).

isc

Interface select code. No bus addressing is performed, therefore the bus remains as previously configured. Data transfer statements such as ENTER, OUTPUT, and READBIN do not transfer data until the HP-IB interface card indicates addressed to talk or listen, as appropriate. (`READBIN (7)` for example, must be addressed to listen before a byte can be input from the interface.) Specify only the interface select code for non-active controller data transfers.

− isc

Negative interface select code. The bus is not reconfigured from a previous configuration. This method of specifying the select code parameter assumes the System 45 to be the active controller on the bus.

scs

Select code sequence. This sequence specifies one or more device bus addresses with the form {ssaa.scg}, where {ss} is a one or two digit interface select code (1-12) and {aa} is a two digit device address (0-30). Multiple device bus addresses are separated by commas, and may include a secondary command sequence {.scg} as described in the next section. Use of a select code sequence (scs) assumes the System 45 is the active controller on the bus. A typical statement looks like:

```
OUTPUT 701,706,712; A$
```

scs

The following statement represents a variation of the scs, and is recognized and accepted by the system:

$$\texttt{OUTPUT}\underbrace{\texttt{7,1,6,12}}_{\text{scs}}\texttt{; A\$}$$

The above two statements are equivalent OUTPUT statements.

– scs

Negative select code sequence. This sequence specifies one or more device bus addresses with either the form {– ssaa.scg} or {– ss,aa.scg,aa.scg...}. The parameter {–ss} is a negative one or two digit interface select code, {aa} is a two digit device address, and {.scg} is the secondary command group with a terminator added as necessary (secondary commands are discussed in the next section). The first form of the negative select code sequence is used to send secondary commands to a device previously addressed without reconfiguring the bus. The second form is used to add one or more devices to the list of listeners already addressed to listen from a previous bus configuration. Both forms assume the System 45 is the active controller on the bus. Typical statements look like this:

$$\texttt{OUTPUT }\underbrace{\texttt{-706.132027}}_{-\text{ scs}}\texttt{; A\$} \qquad \text{(form 1)}$$

$$\texttt{OUTPUT }\underbrace{\texttt{- 7,13,20.1327,05}}_{-\text{ scs}}\texttt{; A\$} \qquad \text{(form 2)}$$

The first statement is valid only if device 06 was the last device addressed to listen, and serves to send the secondary commands 13, 20, and 27 without reconfiguring the bus. The second statement is valid only if the bus has been previously configured for the System 45 to talk, and serves to add devices 13, 20, and 05 to the list of listeners for data output.

Note that not every HP-IB statement allows you to specify a select code using all four of the methods described here. A list of the allowable select code specifications for the HP-IB statements is in Appendix C under HP-IB Command Bus Sequences. The results of executing a statement using the different methods of specifying the select code is shown in the table, as well as which methods are not allowed.

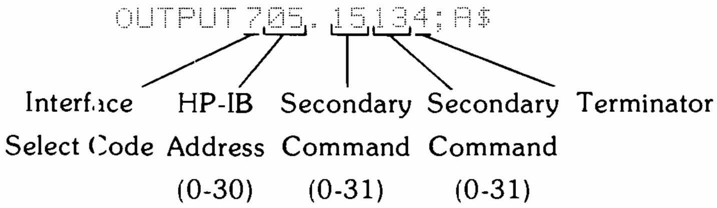
Except for the Configure Bus statement, all I/O operations which have a select code sequence automatically transmit the System 45's preset talker/listener address and the specified device address code(s) in the appropriate order on the bus. When the System 45 is the active controller, the address sequence includes the bus Unlisten command to clear all devices previously set to listen. The sequence of actions for a data transfer on the bus is as follows:

OUTPUT	INPUT
1. System 45 Talk address	1. Device Talk address
2. Unlisten	2. Unlisten
3. Device Listen address(es)	3. System 45 Listen address

After the transfer, the bus is left setup as addressed for the data transfer.

Secondary Commands

When communicating among devices which use extended addressing on the HP-IB, the device's extended address is specified by adding two more digits to the select code parameter. The complete syntax of an extended address is illustrated by the following output statement:



```
OUTPUT 7Ø5.15134;A$
```

The use of extended addressing or secondary commands is provided for in the bus definition (IEEE Std. 488-1975). The primary address of a bus device can be followed by another byte of addressing or command information with an allowed range of 00 through 31 having the Secondary Command Group (SCG) bits (5 & 6) set. These optional bytes are automatically sent when the additional Extended Address digits are specified in the select code parameter.

The secondary command field is terminated by a secondary command greater than 31 to allow you to send a final secondary command of 00. A maximum of twelve digits are allowed in the select code parameter. For instance, the statement

```
    OUTPUT 705.2000;A$
```

would produce the following results when stored:

```
    OUTPUT 705.20;A$
```

The actual secondary command sent is 20. Using the terminator 40 would change the statement to

```
    OUTPUT 705.200040;A$
```

and produce the result

```
    OUTPUT 705.20004;A$
```

The secondary commands actually sent are 20 and 00.

To illustrate the bus sequence for extended addressing, the above output statement initiates the following sequence:

1. System 45 Talk address
2. Unlisten
3. Device Primary address 05
4. Secondary Commands 20 and 00
5. Data `A$`

Non-Active Controller Addressing

When a device is not the active controller on the bus, it cannot address other devices to talk or listen. If the System 45 specifies a device address in an I/O operation while not the active controller, an error message occurs. The System 45 can still send Data and other messages when addressed to talk or listen, however, by specifying the select code of the interface with no device addresses. The System 45 waits until it is addressed to talk or listen before transferring any data if it is not the active controller and only the interface select code is specified. The OUTPUT statement waits until addressed to talk, and the ENTER statement waits until addressed to listen. For example, the following sequence sends a Data message (contents of `Q$`) when addressed to talk:

```
120  !  THE OUTPUT STATEMENT WAITS UNTIL ADDRESSED TO TALK.
130  OUTPUT 7;Q$        !  THIS IS BECAUSE ONLY A SELECT CODE
140                     !  IS SPECIFIED, WITH NO DEVICE ADDRESSES.
```

Device Communication

The Data message is the primary reason for the existence of the interface bus. The Data message exchanges device-dependent data among bus devices, and consists of one or more 8-bit bytes sent across the bus. The Data message is transmitted from one Talker to one or more Listeners on the bus according to the format specified in the IEEE 488-1975 Standard. The Data message can range from control information to device measurement values to program listings, and may be terminated implicitly or explicitly.

Data Output

The OUTPUT statement can be used to transfer device-dependent information from the System 45 (Talker) to one or more devices (Listener) on the bus. More than one device's HP-IB address can be included in the statement syntax, and each device address can include one or more secondary commands.[1] Specifying a negative interface select code causes data to be transferred without reconfiguring the bus. By not reconfiguring the bus for an HP-IB operation, the device previously designated as Talker and the devices previously designated as Listeners remain addressed as Talker and Listeners, respectively, for the next transfer. For example:

```
60   !
70   OUTPUT 705;A$      ! ADDRESS DEVICE 5, SEND A$
80   OUTPUT 705.15;A$   ! ADDRESS DEVICE 5, SEND SECONDARY COMMAND 15, A$
90   OUTPUT -7;A$       ! OUTPUT A$, DON'T RECONFIGURE THE BUS (CONTROLLER)
100  OUTPUT 7;A$        ! OUTPUT WHEN ADDRESSED TO TALK (NON-CONTROLLER)
110  !
```

Data Input

The ENTER statement can be used to transfer device-dependent information from a device on the bus to the System 45. Only one device may be addressed (as Talker) in the ENTER statement syntax, and if a negative device address is specified no change is made to the previous bus configuration, but any specified secondary commands are sent. One or more secondary commands[1] can be included in the device address. The following examples illustrate extended addressing and receiving data without reconfiguring the bus:

```
120  !
130  ENTER 705;A$       ! ADDRESS DEVICE 5
140  ENTER 705.15;A$    ! PRIMARY ADDRESS 5, SECONDARY ADDRESS 15, DATA A$
150  ENTER -7;A$        ! ENTER, DON'T RECONFIGURE THE BUS (CONTROLLER)
160  ENTER 7;A$         ! WAIT UNTIL ADDRESSED TO LISTEN (NON-CONTROLLER)
170  !
```

[1] Secondary command and Secondary address are terms used synonomously. Their interpretation depends on the device addressed.

Device Control

The Trigger Statement

The Trigger statement sends a Trigger message from the active controller to a selected device or group of devices. The purpose of the Trigger message is to initiate some device-dependent action: for example, to trigger a digital voltmeter to perform its measurement cycle, or a digital voltage source to switch to a new setting. Because the response of a device to a Trigger message is strictly device-dependent, neither the Trigger message nor the interface indicates what action is initiated by the device.

Syntax:

```
TRIGGER [-] scs
TRIGGER [-] isc
```

Specifying only the interface card's select code (`TRIGGER 7`) outputs a Trigger message to all devices currently addressed to listen on the bus. Including device addresses in the statement triggers only those devices addressed by the statement. The System 45 cannot respond to a device trigger from another controller on the bus.

The following example presets functions on a frequency counter and a voltmeter, then outputs a Trigger message (line 250) to simultaneously initiate action on both devices. Line 260 then inputs data from the DVM.

```
200  !
210  Counter=703
220  Voltmeter=707
230  OUTPUT Counter;"I2E8E?G?52"   ! PRESET COUNTER FUNCTION.
240  OUTPUT Voltmeter;"T0F1M3E"    ! PRESET METER FUNCTION.
250  TRIGGER Counter,Voltmeter     ! TRIGGER THE COUNTER AND METER.
260  ENTER Voltmeter;Val_1,Val_2   ! ENTER THE METER READINGS.
270  PRINT "VOLTMETER VALUES ARE";Val_1,Val_2
280  !
```

The Reset Statement

The Reset statement provides a means of initializing devices to a predefined device-dependent state. When the Reset statment is executed on an HP-IB interface, the Clear message is sent either to all devices or to a selected set of devices as specified in the statement syntax. Only the active controller can send a Clear message. The Clear message is single-valued in the sense that it is either true or not true on the bus.

Syntax:

RESET [-] scs
RESET [-] isc

Specifying only the interface select code outputs a Device Clear (DCL) command to all devices capable of responding to the DCL command. Specifying device addresses (e.g., RESET 711) outputs a Selected Device Clear (SDC) command which resets only the specified device(s).

For example

```
300  RESET 7                      ! SEND DCL COMMAND (ALL DEVICES)
310  RESET 711                    ! SEND SDC COMMAND (DEVICE 11)
```

The Remote Statement

The Remote statement outputs a Remote message which enables devices on the bus to switch over to remote program control from local front panel control. If the device is addressed by the Remote statement, it switches to the remote state. A device in the remote state may be designed so as to remain unresponsive to some or all of its front panel controls.

Syntax:

REMOTE [-] scs
REMOTE [-] isc

The Remote message is automatically sent whenever the System 45 **as system controller** is switched on, or reset. This is accomplished by setting the REN line true. Therefore, as soon as a device is addressed, it goes into the Remote state. A device can be prevented from being switched to local from a front panel switch by sending the Local Lockout Message. (Refer to the Local Local Statement.) The System 45 must be the system controller to execute the REMOTE statement.

The Local Statement

Executing the Local statement sends the Local message, which returns the specified devices to local, front panel control. During system operation, it is sometimes necessary for an operator to interact with one or more devices. For instance, an operator might need to work from the front panel to make special tests or to troubleshoot. Also, it is generally good systems practice to return all devices to local control upon conclusion of automatic operations.

Syntax:

```
LOCAL [-] sc
LOCAL [-] isc
```

When the LOCAL statement is specified with device address(es), a Local message (GTL) is output to the specified devices. The System 45 must be the active controller to send the LOCAL message.

When no device addresses are specified, executing the LOCAL statement outputs the Clear Lockout/Local message. This message returns all devices to local, front panel control, automatically clearing a previous Local Lockout message if it is still in effect. The System 45 must be the System Controller to send the Clear Lockout/Local message.

Examples:

```
340  LOCAL 7                         ! RETURN ALL DEVICES TO LOCAL


350  LOCAL 712,15                    ! DEVICES 12 AND 15 TO LOCAL
```

The Local Lockout Statement

The Local Lockout message (LLO), sent by executing the LOCAL LOCKOUT statement, prevents an operator from returning a device to local control from its front panel. This message can be sent only when the System 45 is the active controller, and is sent to all devices on the bus. As long as Local Lockout is in effect, no bus device can be returned to local control except by the system controller itself: this maintains automatic system integrity. The Local Lockout message effectively locks out the "local" switch present on most HP-IB device front panels, and prevents a casual passer-by from interfering with system operations by pressing buttons.

Syntax:

```
LOCAL LOCKOUT isc
```

isc: The select code of the HP-IB interface is the only parameter for the LOCAL LOCKOUT statement.

The Local Lockout message is cleared when the Clear Lockout/Local message is sent by executing the LOCAL statement. Note that the Abort message (ABORT IO statement) does **not** cancel the Local Lockout message.

It is generally good practice to send the Local Lockout message when there are devices on the bus that are used for other purposes. The following statement sets local lockout for HP-IB interface 7:

```
10 LOCAL LOCKOUT 7
```

HP-IB Serial Poll

The Serial Poll is a sequential poll of individual devices on the bus. One byte of device status is returned by each device in response to a Serial Poll. This byte is called the Status Byte message, and depending on the device, may indicate an overload, a request for manual service, printer out of paper, etc.

The STATUS statement of this section performs a Serial Poll when the System 45 is the active controller on the bus. The REQUEST statement enables the System 45 to respond to a Serial Poll when some other controller on the bus is the active controller.

The Read Status Statement

A device capable of responding to a Serial Poll returns its status byte when requested by a Serial Poll. The STATUS statement obtains the device status byte and places the decimal equivalent value of the status byte into the variable specified by the STATUS statement.

Syntax:

```
STATUS[-] scs;var
STATUS  -isc; var
```

scs: A standard select code sequence is allowed **with the exception of** secondary addressing, which is **not** allowed.

var: The variable parameter specifies a numeric variable for the returned decimal equivalent of the device status byte.

The following example obtains device status from device 05, select code 7:

```
380  STATUS 705;Stat              ! DO SERIAL POLL ON DEVICE 5
```

The Request Service Statement

The System 45 can request service from an active bus controller by executing the Request Service statement. This statement can be executed any time the System 45 is not the active controller on the bus. Bus configuration is not altered (and cannot be altered except by the active controller) by the Request Service Statement.

Syntax:

```
REQUEST isc; response byte
```

response byte: The response byte parameter is either a numeric expression with a value between 0 and 127 inclusive, or a string of seven characters (Ø or 1) representing the response byte in binary form. The response byte is sent by the System 45 in response to a serial poll from the active controller.

Setting bit 6 of the response byte parameter will cause the SRQ line of the bus to be set true by the System 45. The SRQ line is not reset until the active controller conducts a serial poll to the System 45 or another REQUEST statement is executed with bit 6=0. Setting bit 6 of the response byte also enables the Parallel Poll response bit in the case of a Parallel Poll conducted by the active controller.

HP-IB Parallel Poll

The Parallel Poll is the fastest means of gathering device status on the HP-IB when several devices are connected to the bus. Each device responds with one bit of status in response to a Parallel Poll, making it possible to obtain device status on more than one device at once. When a device responds to a Parallel Poll, more information as to the specific status of the device can be obtained by conducting a Serial Poll to the responding device.

The Parallel Poll Configure Statement

Certain devices can be remotely programmed by the active bus controller to respond on a specific bit for a Parallel Poll. The logical state of the response (1 or 0) as well as the bit number (0−7) for the device can be programmed by the Parallel Poll Configure statement.

Syntax:

```
PPOLL CONFIGURE [−] scs; mask
PPOLL CONFIGURE −isc; mask
```

mask: The mask is a numeric expression to be converted to a binary integer or a string mask of characters 1 and 0. The least significant 3 bits of the numeric expression or the string mask are used to determine which line number (Status bit) the device(s) is to respond on. The fourth bit specifies the "true" state of the Parallel Poll response bit of the device(s).

This statement should only be used with HP-IB devices having remote parallel poll configuration capability. The 98034A HP-IB interface card has only a switch-selectable response to a Parallel Poll from another controller and cannot be remotely configured for a Parallel Poll.

For example, the following two statements configure a Parallel Poll response of "0" true on bit 4 from device 06:

```
410  PPOLL CONFIGURE 706;"00000100"      ! RESPOND ON BIT 4 WITH A 0
420  PPOLL CONFIGURE 704;4               ! EQUIVALENT TO ABOVE STATEMENT
```

The Parallel Poll Unconfigure Statement

This statement gives the System 45 (as active controller) the capability of disabling Parallel Poll responses from one or more devices on the bus.

Syntax:

PPOLL UNCONFIGURE [–] scs

PPOLL UNCONFIGURE [–] isc

If no device addresses are specified, all bus devices are disabled from responding to a Parallel Poll. If device addresses are specified, the specified devices having a Parallel Poll Configure capability are disabled.

The following statements perform the following: 1) disables all devices from Parallel Poll response 2) disables device 06 only:

```
450  PPOLL UNCONFIGURE Hpib         ! DISABLE ALL FROM PARALLEL POLL
460  !
470  PPOLL UNCONFIGURE 706          ! DISABLE PARALLEL POLL FOR DEV 6
```

The Parallel Poll Function

The Parallel Poll function can be executed when the System 45 is the active controller, and returns a single byte representing the 8 Status Bit messages of the devices on the bus capable of responding to a Parallel Poll. Each bit corresponds to the status bit of the device(s) responding to the Parallel Poll. (Recall that one or more devices can respond on a single line.)

Syntax:

PPOLL(isc)

If a parallel poll is taken from an inactive select code, the value returned is zero (0).

For example...

```
500  Poll=PPOLL(7)                  ! CONDUCT PARALLEL POLL
510  PRINT "PARALLEL POLL=";Poll    ! Poll REFLECTS A PARALLEL POLL
520  SELECT CODE 7 INACTIVE
530  Poll=PPOLL(7)                  ! PARALLEL POLL INACTIVE INTERFACE
540  PRINT "INACTIVE POLL=";Poll    ! Poll IS NOW 0
```

Bus Management

The bus management statements allow you to configure the bus for your own specialized application, to send multi-line commands and data fields for full device-dependent control, and to relinquish control of the bus to another, non-active controller. Another statement, ABORTIO, allows the System 45 to terminate bus activities and resume system controller responsibilities when circumstances require immediate action.

The Configure Bus Statement

The Configure Bus statement is provided for applications where the System 45 is not directly interacting with data transfers on the bus, or where one talker on the bus sends data to one or more listeners. The Configure Bus statement can be executed only when the System 45 is the active controller on the bus.

Syntax:

CONFIGURE isc TALK=dev$_1$ LISTEN=[, dev$_2$[, dev$_3$]]...

CONFIGURE isc TALK=dev$_1$

CONFIGURE isc LISTEN =dev$_2$ [, dev$_3$[, dev$_4$]]...

dev$_1$: The talker device address can be any valid HP-IB device address specification.

dev$_2$-dev$_n$: Any number of valid listener device addresses can be specified.

The bus remains in the configuration set by the CONFIGURE statement until it is changed by another HP-IB operation.

The special case where the System 45 is not set up as either the talker or as the listener by the CONFIGURE statement needs further explanation. Your program must read the status of the interface to check for EOI (End or Identify). This information is available in the STATUS section of the Elementary I/O Operations chapter. The EOI status bit, when set, indicates the data transfer is complete and that the System 45 can initiate other HP-IB operations.

If the System 45 is set up by the CONFIGURE statement as one of the bus listeners, it is necessary to follow the CONFIGURE statement with an ENTER[1] or READBIN[1] statement. Likewise, where the System 45 is set up as the bus talker by the CONFIGURE statement, an OUTPUT[1] statement is necessary.

For example...

```
580  CONFIGURE Hpib TALK = 7 LISTEN = 1,9,15     ! DEVICE 7 IS TALKER.
590  !   DEVICES 1,9, AND 15 ARE LISTENERS.(SYSTEM 45 IS NOT ADDRESSED)
600  !
610  CONFIGURE Hpib TALK = 7             ! SYSTEM 45 IS LISTENER,
620  !   DEVICE 7 IS TALKER.
630  !
640  CONFIGURE Hpib LISTEN = 1,9,15      ! SYSTEM 45 IS TALKER,
650  !   DEVICES 1,9, AND 15 ARE LISTENERS.
660  !
```

The Send Bus Statement

The SENDBUS statement provides you with the capability of directly specifying command and data sequences as either ASCII characters or numeric expressions. All HP-IB commands and addresses can be sent, allowing you to custom-tailor your own HP-IB address and command sequences.

Syntax:

SENDBUS scs ; commands [; data][; commands][; data]...

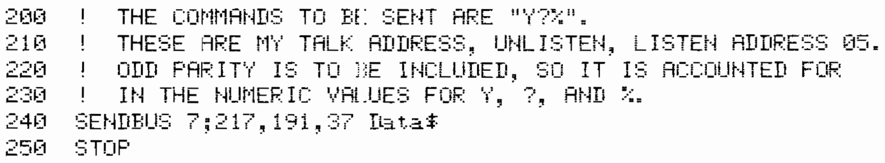
commands: The commands field is a sequence of address(es) and commands, and can be specified as either numeric expressions separated by commas, or a single string expression. Commands are sent with ATN true.

[1] These statements are executed with a negative interface select code specified or the interface select code with no device addresses.

data: The data field is a sequence of data specified as either numeric expressions separated by commas, or a single string expression. Data is sent with ATN false, and is not automatically terminated with any default control characters such as carriage-return or line-feed.

The System 45 must be the active controller to execute the SEND BUS statement. The SENDBUS statement gives you extensive control over the HP-IB command and data transfers, so be certain you completely understand the bus configuration and command set of your particular HP-IB system before using the SENDBUS statement. Unexpected operation and/or loss of data may result from improper utilization of the SENDBUS statement capabilities. For instance, the result of specifying another device's talk address or the System 45's listen address in the command field is that the following data field is not sent.

Some devices require parity to accept commands, and the SENDBUS statement does not generate parity. You must specify commands with the proper parity to satisfy those devices requiring parity. For example,. .



```
200  !  THE COMMANDS TO BE SENT ARE "Y?%".
210  !  THESE ARE MY TALK ADDRESS, UNLISTEN, LISTEN ADDRESS 05.
220  !  ODD PARITY IS TO BE INCLUDED, SO IT IS ACCOUNTED FOR
230  !  IN THE NUMERIC VALUES FOR Y, ?, AND %.
240  SENDBUS 7;217,191,37 Data$
250  STOP
```

The Pass Control Statement

The Pass Control statement implements the Pass Control message. When the System 45 is the active bus controller, it can specify another, nonactive controller to be the active bus controller. As the other device becomes the active controller, the System 45 assumes a nonactive controller role. The System 45 becomes, in effect, a bus device that can be addressed to talk or listen.

Syntax:

```
PASS CONTROL [-]scs
PASS CONTROL -isc
```

When the System 45 is the active controller and executes the PASS CONTROL statement, the System 45 becomes a bus device. As a device, it cannot address other devices to talk or listen. To enter and output data while the System 45 is a nonactive controller, you must specify the interface select code without device addresses or secondary commands in the ENTER and OUTPUT statements.

When the System 45 is not the active controller on the bus, control can be passed back to the System 45 by the active controller. An end-of-line branch condition can be enabled for this event by using the CONTROL MASK, CARD ENABLE and ON INT statements. The program will branch to the service routine for that select code when control is passed to the System 45.

For example...

```
680  ON INT #7,7 GOTO Control     ! ESTABLISH END-OF-LINE BRANCHES.
690  CONTROL MASK 7;2^6         ! ENABLE BRANCH ON ACTIVE CONTROLLER.
700  PASS CONTROL 705           ! PASS CONTROL TO DEVICE 05.
710  CARD ENABLE 7              ! ENABLE INTERRUPTS, SELECT CODE 7.
720  GOTO 720
730 Control:   REM
740  PRINT "SYSTEM 45 IS NOW ACTIVE CONTROLLER"
750  GOTO 700
```

The Abort I/O Statement

The purpose of the Abort I/O statement is to reset the interface functions of all devices on the bus. This statement can be executed when the System 45 is the system controller on the bus. The message sent on the bus is the Abort message, which returns the System 45 (must be the system controller) to active bus control.

Syntax:

ABORTIO **isc**

Appendix A

ENTER Capabilities

	FORMATTED (Freefield or Image)					NOFORMAT					
	Termination	ON INT Branch?	Convert & Parity?	Overlap[3] Compute?	Overlap[4] I/O?	Termination	ON INT Branch	Convert & Parity	Overlap[3] Compute?	Overlap[4] I/O?	Variable List Limitations
HS fault andshake	Variable size, error condition, LF or EOI[2]	No	Yes	Yes, if list is a string var.	With INT, DMA transfers[1]	Variable size or EOI	No	No	Yes	With INT, DMA transfers	Not A$(*)
HS andshake	Variable size, error condition, LF or EOI[2]	No	Yes	Yes, if list is a string var.	With INT, DMA transfers[1]	Variable size or EOI	No	No	Yes	With INT, DMA transfers	Not A$(*)
INT Interrupt	Variable size, error condition, LF or EOI[2]	Yes	Yes	Yes, if list is a string var.	With HS, INT, DMA transfers	Variable size or EOI	Yes	No	Yes	With HS, INT, DMA	Not A$(*)
INT Interrupt	Variable size, error condition, LF or EOI[2]	Yes	Yes	Yes, if list is a string var.	With HS, INT, DMA transfers	Variable size or EOI	Yes	No	Yes	With HS, INT, DMA	Not A$(*)
FHS e Fast dshake	COUNT parameter or EOI	Yes	Yes	No	No	COUNT parameter	Yes	No	Yes	No	Only 1 Variable not A$(*)
FHS d Fast dshake	COUNT parameter	Yes	Yes	No	No	COUNT parameter	Yes	No	Yes	No	Only 1 Variable not A$(*)
DMA Direct y Access	COUNT parameter	Yes	Yes	No	With HS, INT,FHS transfers	COUNT parameter	Yes	No	Yes	With INT, HS,FHS transfers	Only 1 Variable not A$(*)
DMA Direct y Access	COUNT parameter	Yes	Yes	No	With HS, INT,FHS transfers	COUNT parameter	Yes	No	Yes	With INT, HS,FHS transfers	Only 1 Variable not A$(*)
R-VAR able to ariable	Source or Destination Variable size	No	Yes	No	Yes						

ansfers must be in effect before initiating the handshake transfer.
R waits for a Line-Feed or EOI to terminate operation, unless the image reference specifies a "%" or "+" or "#".
p Compute: Does the program continue executing when the I/O transfer occurs?
p I/O: Does the I/O transfer allow concurrent I/O operations to another select code?

TRANSFER TYPES

OUTPUT Capabilities

	FORMATTED (Freefield or Image)					NOFORMAT					
	Termination	ON INT Branch?	Convert & Parity	Overlap[3] Compute?	Overlap[4] I/O?	Termination	ON INT Branch?	Convert &Parity?	Overlap[3] Compute?	Overlap[4] I/O?	Variable List Limitations
BHS Default Byte Handshake	End of Variable List or Error	No	Yes	Yes	With INT, DMA transfers[1]	End of Variable List or Error	No	No	Yes	With INT, DMA transfers[1]	Not A$(*)
WHS Word Handshake	End of Variable List or Error	No	Yes	Yes	With INT, DMA transfers[1]	End of Variable List or Error	No	No	Yes	With INT, DMA transfers[1]	Not A$(*)
BINT Byte/Interrupt	End of Variable List or Error	Yes	Yes	Yes	With INT HS, DMA transfers	End of Variable List or Error	Yes	No	Yes	With INT HS, DMA transfers	Not A$(*)
WINT Word/Interrupt	End of Variable List or Error	Yes	Yes	Yes	With INT, HS, DMA transfers	End of Variable List or Error	Yes	No	Yes	With INT, HS, DMA transfers	Not A$(*)
BFHS Byte Fast Handshake	End of Variable List or Error	Yes	Yes	Yes	No	End of Variable List or Error	Yes	No	Yes	No	Only 1 Variable not A$(*)
WFHS Word Fast Handshake	End of Variable List or Error	Yes	Yes	Yes	No	End of Variable List or Error	Yes	No	Yes	No	Only 1 Variable not A$(*)
BDMA Byte Direct Memory Access	End of Variable List or Error	Yes	Yes	Yes	With INT,HS transfers	End of Variable List or Error	Yes	No	Yes	With HS,INT transfers	Only 1 Variable not A$(*)
WDMA Word Direct Memory Access	End of Variable List or Error	Yes	Yes	Yes	With INT,HS transfers	End of Variable List or Error	Yes	No	Yes	With HS,INT transfers	Only 1 Variable not A$(*)
VAR-VAR Variable to Variable	End of Variable List or Error[2]	No	Yes	Yes	Yes						

[1] The INT and DMA transfers must be in effect before initiating the handshake transfer.
[2] An Error 17 occurs when the amount of data exceeds the size of the destination Variable.
[3] Overlap Compute: Does the program continue executing when the I/O transfer occurs?
[4] Overlap I/O: Does the I/O transfer allow concurrent I/O operations to another select code?

Appendix B

Halting Program Execution

PAUSE: Pressing the PAUSE key while a program is executing and any I/O is in progress stops the program but allows any I/O in progress to run to completion.

STOP: Pressing the STOP key while a program is executing with I/O in progress stops the program and allows any input in progress to run to completion. Any output in progress terminates with the end of the current line of output.

RESET Pressing CONT • STOP while a program is executing and I/O is in progress stops the program and immediately terminates all I/O. It is possible that under certain circumstances a SCRATCH ALL will be executed if the System 45 is reset while I/O is in progress.

NOTE

Resetting the System 45 while I/O is in progress can result in a SCRATCH ALL. Programs and data are lost when this occurs, so reset with caution!

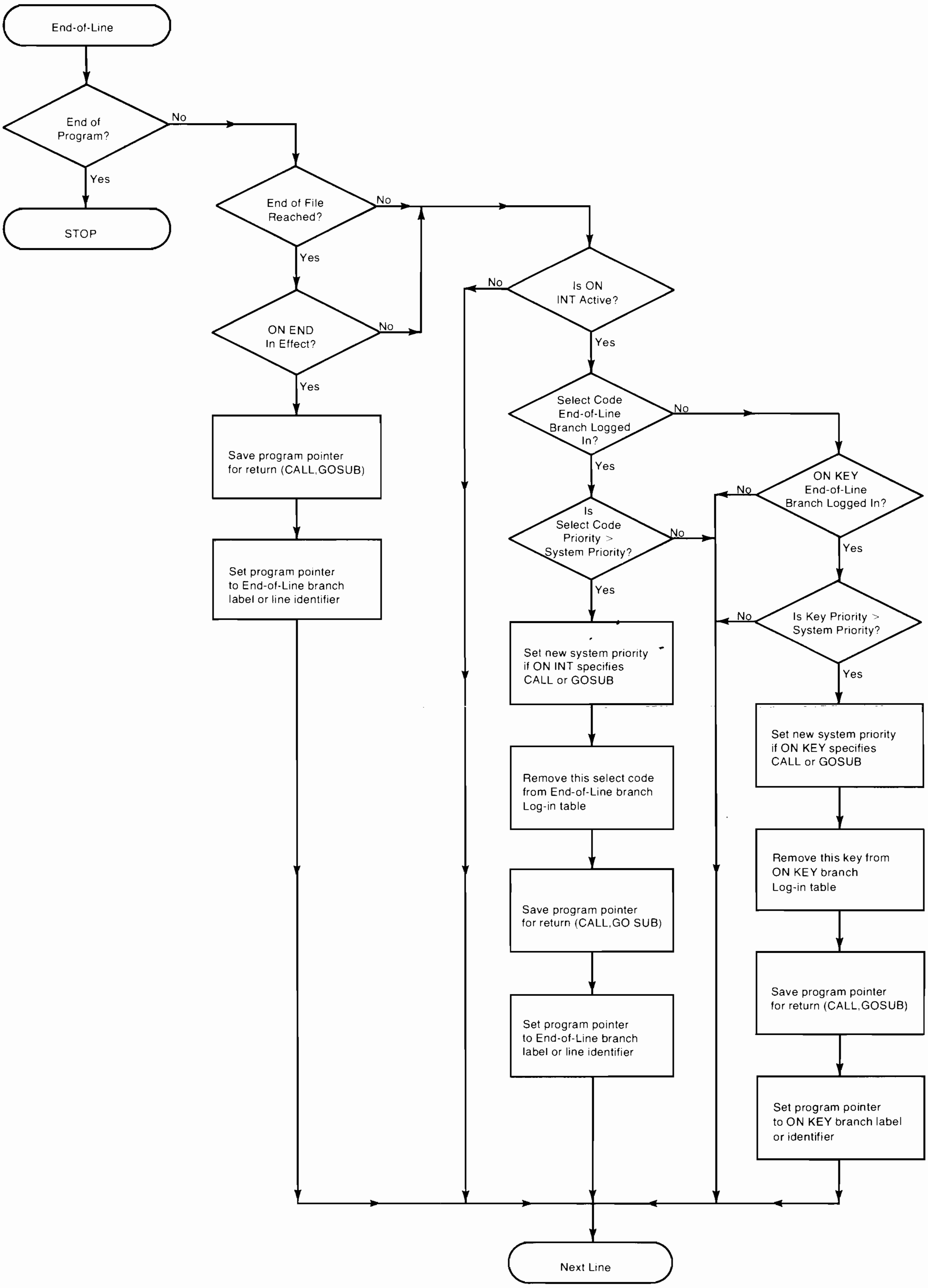
End-of-Line Execution

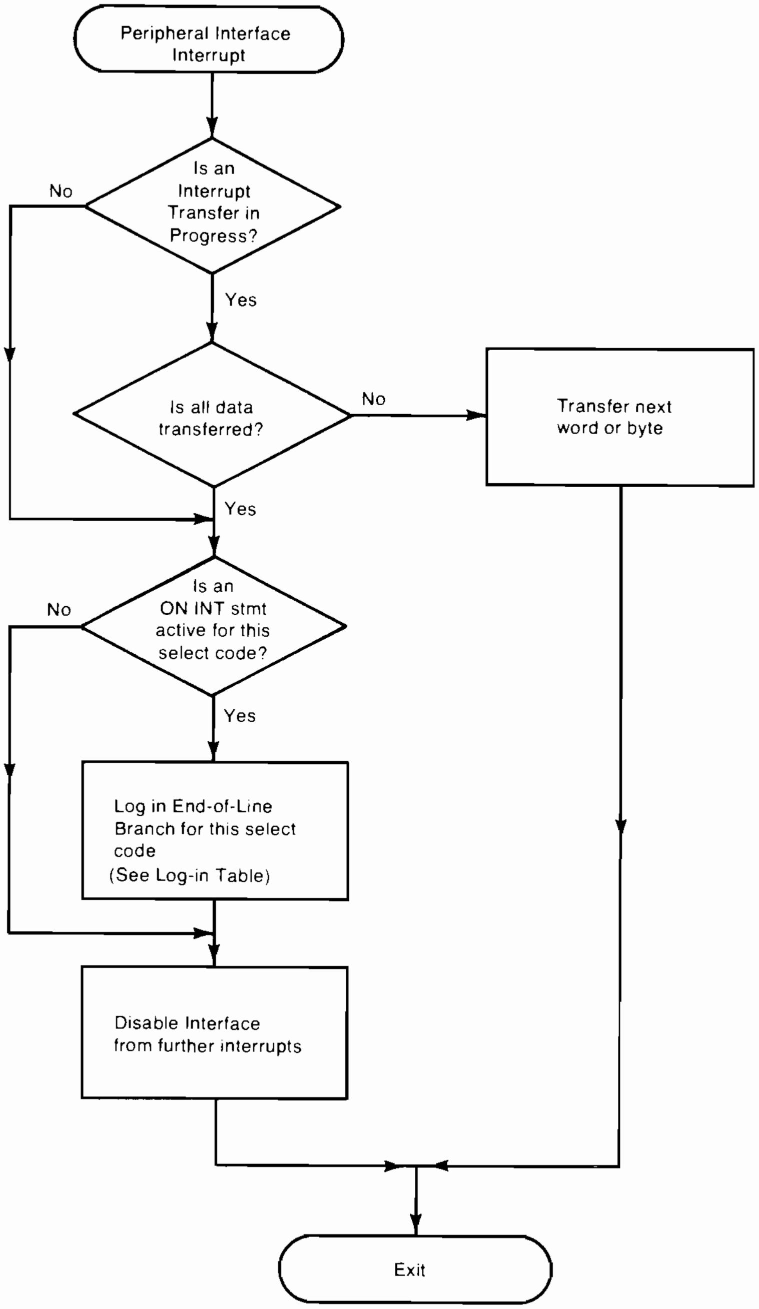
The attached flowchart illustrates the decision process associated with each program line. At the end of a program line, interrupt and ON KEY status is checked to determine if an end-of-line program branch is to be taken.

The flowchart on the following page depicts the ROM-level processing done upon receipt of an interface interrupt, and the log-in-table used to maintain a record of interrupts received.

rogram line. At
ne if an end-of-

on receipt of an
received.


End-of-Line
End of Program?
No
Yes
STOP
End of File Reached?
No
Yes
ON END In Effect?
No
Yes
Save program pointer for return (CALL,GOSUB)
Set program pointer to End-of-Line branch label or line identifier
Is ON INT Active?
No
Yes
Select Code End-of-Line Branch Logged In?
No
Yes
Is Select Code Priority > System Priority?
No
Yes
Set new system priority if ON INT specifies CALL or GOSUB
Remove this select code from End-of-Line branch Log-in table
Save program pointer for return (CALL,GO SUB)
Set program pointer to End-of-Line branch label or line identifier
ON KEY End-of-Line Branch Logged In?
No
Yes
Is Key Priority > System Priority?
No
Yes
Set new system priority if ON KEY specifies CALL or GOSUB
Remove this key from ON KEY branch Log-in table
Save program pointer for return (CALL,GOSUB)
Set program pointer to ON KEY branch label or identifier
Next Line


Peripheral Interface Interrupt
Is an Interrupt Transfer in Progress?
No
Yes
Is all data transferred?
No
Transfer next word or byte
Yes
Is an ON INT stmt active for this select code?
No
Yes
Log in End-of-Line Branch for this select code (See Log-in Table)
Disable Interface from further interrupts
Exit

End-of-Line Branch Log-In Table

Select Code 15	Select Code 14	← Select Codes 2-13 →	Select Code 1	Select Code 0

9830A-9825A-9845A
I/O Comparisons

This section presents you with a brief look at some of the ways in which I/O operations are accomplished by the 9830A, the 9825A, and the 9845A desktop computers. It is not within the scope of this manual to provide an extensive statement-by-statement breakdown of the different I/O syntaxes, but the more fundamental ones are described below.

Output Format Equivalents

Task: Send three bytes, format text, and three fix-field numerics.

9830A

```
10 FORMAT 3E,"JELLY", 3F1010.1
20 OUTPUT (6,10) 34,91,93,10.2,97.4,45.3
```

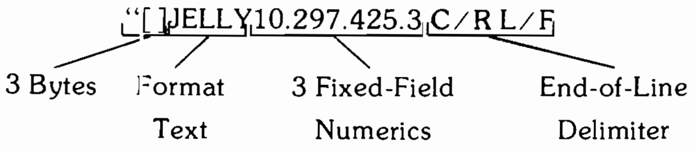
9825A

```
10fmt1,3b,"JELLY",3f4.1
11wrt6.1,34,91,93,10.2,97.4,45.3
```

9845A

```
10 IMAGE 3(E),"JELLY", 3(2D.D)
20 OUTPUT 6 USING 10;34,91,93,10.2,97.4,45.3
```

Results:



Input Format Equivalents

Task: Read the above output data.

9830A

```
10 FORMAT 3B,5B,3F4.1
20 ENTER (9,10)J,K,L,(FOR I=1 TO 5,A[I]),A,B,C
30 TRANSFER A[1,5] TO A$
```

9825A

```
10fmt1,3b,c5,3f4.1
11red9.1,J,K,L,A$,A,B,C
```

9845A

```
10 IMAGE 3(B),5A,3(4N)
20 ENTER 9 USING 10;J,K,L,A$,A,B,C
```

Results:

J=34,K=91,L=93
A$="JELLY"
A=10.2,B=97.4,C=25.3

Functional Equivalents:

9830A	9825A	9845A
RBYTE	rdb	READBIN
PRINT WBYTE	wtb	OUTPUT Sc USING "#,B"
CMD	cmd	CONFIGURE, SENDBUS
OUTPUT	wrt	OUTPUT
FORMAT	fmt	IMAGE
	ctbl	CONVERT
	oni	ON INT
	ios	STATUS
	iof	IOFLAG
	rdi	READIO
	wti	WRITEIO
	wtc	CONTROL MASK, CARD ENABLE
	eir	CONTROL MASK, CARD ENABLE
	rds	STATUS
	dto	FNOct[1]
	otd	FNDec[2]
	iret	RETURN

[1]

```
140   ! OCTAL-TO-DECIMAL CONVERSION ROUTINE.
150   !
160   DEF FNDec(Oct)
170   Dec=0
180   FOR I=0 TO 5
190   Dec=Dec+10*FRACT(Oct/10)*8^I
200   Oct=INT(Oct/10)
210   NEXT I
220   IF Dec>32767 THEN Dec=Dec-65536
230   RETURN Dec
240   FNEND
```

[2]

```
10    ! DECIMAL-TO-OCTAL CONVERSION ROUTINE.
20    !
30    DEF FNOct(Dec)
40    IF Dec<0 THEN Dec=65536+Dec
50    Oct=0
60    FOR I=0 TO 5
70    Oct=Oct+Dec MOD 8*10^I
80    Dec=INT(Dec/8)
90    NEXT I
100   RETURN Oct
110   FNEND
```

Appendix C

HP-IB

Available Bus Addresses and Codes

Address Characters		Address Switch Settings					Address Codes
Listen	Talk	(5)	(4)	(3)	(2)	(1)	decimal
SP	@	0	0	0	0	0	0
!	A	0	0	0	0	1	1
"	B	0	0	0	1	0	2
#	C	0	0	0	1	1	3
$	D	0	0	1	0	0	4
%	E	0	0	1	0	1	5
&	F	0	0	1	1	0	6
'	G	0	0	1	1	1	7
(	H	0	1	0	0	0	8
)	I	0	1	0	0	1	9
*	J	0	1	0	1	0	10
+	K	0	1	0	1	1	11
,	L	0	1	1	0	0	12
–	M	0	1	1	0	1	13
.	N	0	1	1	1	0	14
/	O	0	1	1	1	1	15
0	P	1	0	0	0	0	16
1	Q	1	0	0	0	1	17
2	R	1	0	0	1	0	18
3	S	1	0	0	1	1	19
4	T	1	0	1	0	0	20
5	U	1	0	1	0	1	21
6	V	1	0	1	1	0	22
7	W	1	0	1	1	1	23
8	X	1	1	0	0	0	24
9	Y	1	1	0	0	1	25 ← preset
:	Z	1	1	0	1	0	26
;	[	1	1	0	1	1	27
<	\	1	1	1	0	0	28
=	]	1	1	1	0	1	29
>	^	1	1	1	1	0	30

Bus Mnemonic Descriptions

HP-IB Mnemonic	Description	Octal Value
IFC	Interface Clear	IFC line true
GET	Group Execute Trigger	010
LAG	Listen Address Group	Device's listen address
MTA[1]	My Talk Address	Controller's talk address
MLA[2]	My Listen Address	Controller's Listen Address
UNL	Unlisten	077
DCL	Device Clear	024
SDC	Selected Device Clear	004
REN	Remote Enable	Remote line true
GTL	Go To Local	001
PPC	Parallel Poll Configure	005
PPE	Parallel Poll Enable	140 + mask
PPD	Parallel Poll Disable	160
PPU	Parallel Poll Unconfigure	025
EOI	End or Identify:	EOI line true
	End if –	$\overline{\text{ATN}}$ ● EOI
	Identify (Parallel Poll) if –	ATN ● EOI

[1] 98034A Interface's preset talk address = 125 Octal

[2] 98034A Interface's preset listen address = 065 Octal.

HP-IB Universal Commands (ATN true)

Primary Command Group (PCG)

Decimal Value	ASCII Character	Interface Message	Description
0	NUL		
1	SOH	GTL	Go To Local
2	STX		
3	ETX		
4	EOT	SDC	Selected Device Clear
5	ENQ	PPC	Parallel Poll Configure
6	ACK		
7	BEL		
8	BS	GET	Group Execute Trigger
9	HT	TCT	Take Control
10	LF		
11	VT		
12	FF		
13	CR		
14	SO		
15	SI		
16	DLE		
17	DC1	LLO	Local Lockout
18	DC2		
19	DC3		
20	DC4	DCL	Device Clear
21	NAK	PPU	Parallel Poll Unconfigure
22	SYN		
23	ETB		
24	CAN	SPE	Serial Poll Enable
25	EM	SPD	Serial Poll Disable
26	SUB		
27	ESC		
28	FS		
29	GS		
30	RS		
31	US		
32-62	SP to > (Numbers, special char)	LAG	Listen Address Group
63	?	UNL	Unlisten
64-94	@ to ↑ (Upper case ASCII)	TAG	Talk Address Group
95	—	UNT	Untalk
96-126	(lowercase ASCII)	SCG	Secondary Command Group
127	DEL		

HP-IB Command Bus Sequences

Select Code Specifications

isc:	interface select code only
scs:	select code sequence
− isc:	negative interface select code only
− scs:	negative select code sequence

Command	Sequence
ABORTIO	
isc:	IFC
scs:	Not Allowed
− isc:	
− scs:	
LOCAL	
isc:	$\overline{\text{REN}}$ • $\overline{\text{ATN}}$
scs:	ATN, MTA, UNL, LAG, GTL
− isc:	ATN, GTL
− scs:	ATN, LAG, GTL
LOCAL LOCKOUT	
isc:	ATN, LLO
scs:	Not Allowed
− isc:	
− scs:	
PASS CONTROL	
isc:	Not Allowed
scs:	ATN, UNL, TA[1], TCT, $\overline{\text{ATN}}$
− isc:	ATN, UNL, TA[1], TCT, $\overline{\text{ATN}}$
− scs:	ATN, UNL, TA[1], TCT, $\overline{\text{ATN}}$
PPOLL	
isc:	ATN • EOI,[≅100us],$\overline{\text{ATN}}$ • $\overline{\text{EOI}}$
scs:	Not Allowed
− isc:	
− scs:	
PPOLL CONFIGURE	
isc:	Not Allowed
scs:	ATN, MTA, UNL, LAG, PPC, PPE
− isc:	ATN, PPC, PPE
− scs:	ATN, LAG, PPC, PPE

[1] The Talk address sent is the talk address of the device receiving control.

HP-IB Bus Sequences (Cont'd)

Command	Sequence
PPOLL UNCONFIGURE	
isc:	PPU
scs:	MTA, UNL, LAG, PPC, PPD
− isc:	PPC, PPD
− scs:	LAG, PPC, PPD
REMOTE	
isc:	REN • $\overline{ATN}$
scs:	REN • $\overline{ATN}$,ATN,MTA,UNL,LAG
− isc:	REN • $\overline{ATN}$
− scs:	REN • $\overline{ATN}$,LAG
REQUEST	
isc:	SRQ
scs:	Not Allowed
− isc:	
− scs:	
RESET	
isc:	ATN, DCL
scs:	ATN, MTA, UNL, LAG, SDC
− isc:	ATN, SDC
− scs:	ATN, LAG, SDC
SERIAL POLL	
isc:	Not Allowed
scs:	ATN,UNL,MLA,TA[1],SPE,$\overline{ATN}$,[DATA],ATN,SPD
− isc:	ATN,TA[1],SPE,$\overline{ATN}$,[DATA],ATN,SPD
− scs:	ATN,UNL,MLA,TA[1],SPE,$\overline{ATN}$,[DATA],ATN,SPD
TRIGGER	
isc:	ATN, GET
scs:	ATN, MTA, UNL, LAG, GET
− isc:	ATN, GET
− scs:	ATN, LAG, GET

[1] The Talk address sent is the talk address of the device being polled.

Key Code	ASCII Character	Comments	Key(s) to Press☆	Octal Code
0	NUL	Null	CONTROL space bar	00
1	SOH	Start of Header	CONTROL A	01
2	STX	Start of Text	CONTROL B	02
3	ETX	End of Text	CONTROL C	03
4	EOT	End of Transmission	CONTROL D	04
5	ENQ	Enquiry	CONTROL E	05
6	ACK	Acknowledgement	CONTROL F	06
7	BEL	Bell	CONTROL G	07
8	BS	Backspace	CONTROL H	10
9	HT	Horizontal Tab	CONTROL I	11
10	LF	Line Feed	CONTROL J	12
11	VT	Vertical Tab	CONTROL K	13
12	FF	Form Feed	CONTROL L	14
13	CR	Carriage Return	CONTROL M	15
14	SO	Shift Out	CONTROL N	16
15	SI	Shift In	CONTROL O	17
16	DLE	Data Link Escape	CONTROL P	20
17	DC1	Device Control	CONTROL Q	21
18	DC2	Device Control	CONTROL R	22
19	DC3	Device Control	CONTROL S	23
20	DC4	Device Control	CONTROL T	24
21	NAK	Negative Acknowledgement	CONTROL U	25
22	SYN	Synchronous Idle	CONTROL V	26
23	ETB	End of Text Block	CONTROL W	27
24	CAN	Cancel	CONTROL X	30
25	EM	End of Media	CONTROL Y	31
26	SUB	Substitute	CONTROL Z	32
27	ESC	Escape	CONTROL { [	33
28	FS	File Separator	CONTROL | \	34
29	GS	Group Separator	CONTROL }]	35
30	RS	Record Separator	CONTROL SHIFT ^ 6 *	36
31	US	Unit Separator	CONTROL SHIFT ? /	37

Key Code	ASCII Character	Comments	Key(s) to Press☆	Octal Code
32	SP	Blank	space bar	40
33	!	Exclamation Point	SHIFT ! 1	41
34	"	Double Quote	SHIFT " '	42
35	#	Pound Sign	SHIFT # 3	43
36	$	Dollar Sign	SHIFT $ 4	44
37	%	Percent Sign	SHIFT % 5	45
38	&	Ampersand	SHIFT & 7	46
39	'	Apostrophe	" '	47
40	(	Left Parenthesis	SHIFT (9 *	50
41	)	Right Parenthesis	SHIFT) Ø *	51
42	*	Asterisk	SHIFT * 8 *	52
43	+	Plus Sign	SHIFT = + *	53
44	,	Comma	< , *	54
45	—	Minus Sign (Dash)	— – *	55
46	.	Period	> . *	56
47	/	Forward Slash	? / *	57
48	0	Numerics	) Ø *	60
49	1		! 1 *	61
50	2		@ 2 *	62
51	3		# 3 *	63
52	4		$ 4 *	64
53	5		% 5 *	65
54	6		^ 6 *	66
55	7		& 7 *	67
56	8		* 8 *	70
57	9		(9 *	71
58	:	Colon	SHIFT : ;	72
59	;	Semicolon	: ;	73
60	<	Less Than	SHIFT < ,	74
61	=	Equal	SHIFT = + *	75
62	>	Greater Than	SHIFT > .	76
63	?	Question Mark	SHIFT ? /	77

☆Multiple keys must be pressed simultaneously.

*Also can be found among calculator keys.

Key Code	ASCII Character	Comments	Key(s) to Press☆	Octal Code
64	@	Commercial At	@ 2	100
65	A	Capital Letters	A	101
66	B		B	102
67	C		C	103
68	D		D	104
69	E		E	105
70	F		F	106
71	G		G	107
72	H		H	110
73	I		I	111
74	J		J	112
75	K		K	113
76	L		L	114
77	M		M	115
78	N		N	116
79	O		O	117
80	P		P	120
81	Q		Q	121
82	R		R	122
83	S		S	123
84	T		T	124
85	U		U	125
86	V		V	126
87	W		W	127
88	X		X	130
89	Y		Y	131
90	Z		Z	132
91	[	Left Bracket	[[**	133
92	\	Reverse Slash	| \	134
93	]	Right Bracket	]] **	135
94	↑	Up Arrow	SHIFT ^ 6 *	136
95	—	Underscore	SHIFT — _	137

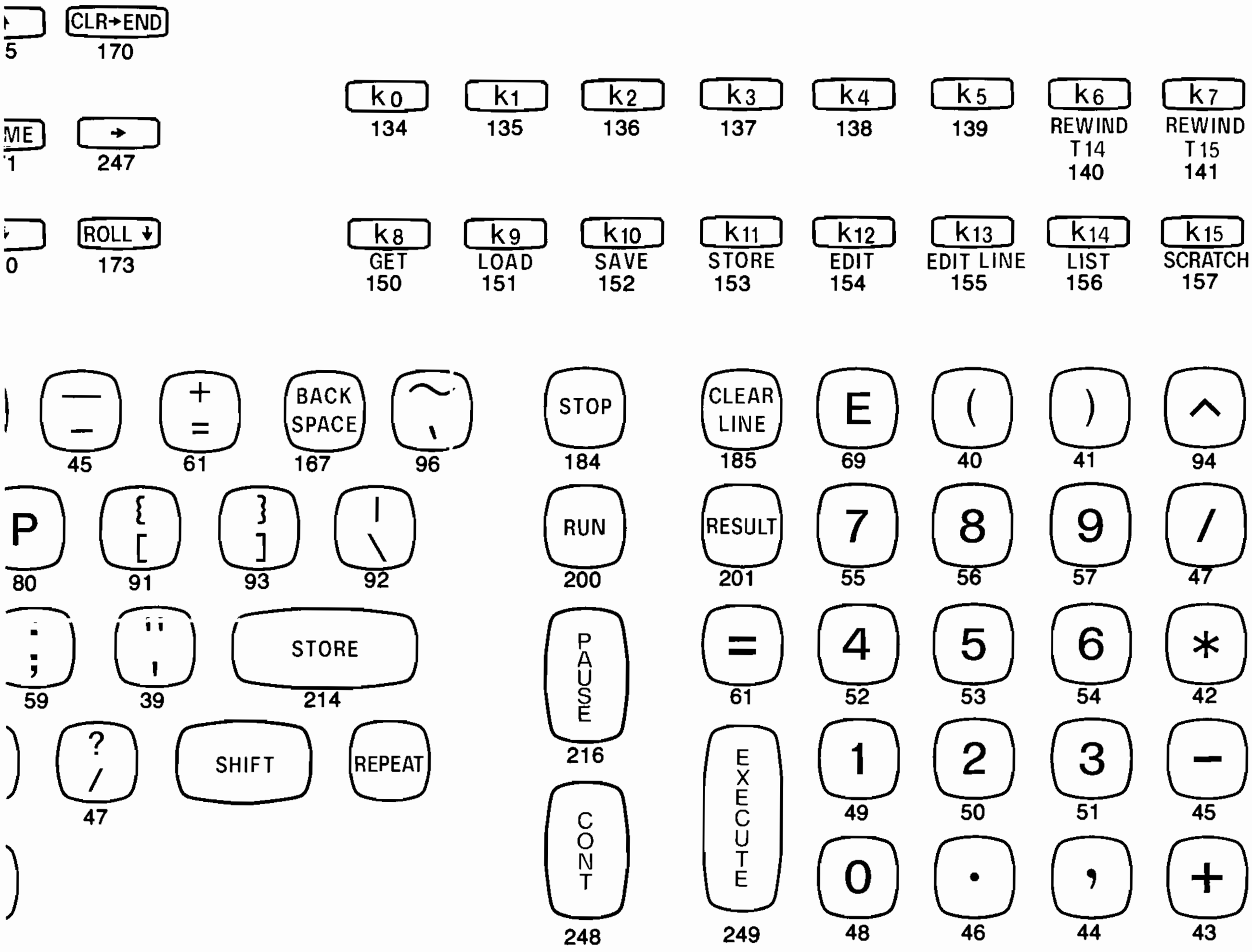
☆Multiple keys must be pressed simultaneously.

*Also can be found among calculator keys.

**Shift (and) on calculator keys.

Key Code	ASCII Character	Comments	Key(s) to Press☆	Octal Code
96	'	Grave Mark	SHIFT ~ `	140
97	a	Noncapital Letters	SHIFT A	141
98	b		SHIFT B	142
99	c		SHIFT C	143
100	d		SHIFT D	144
101	e		SHIFT E	145
102	f		SHIFT F	146
103	g		SHIFT G	147
104	h		SHIFT H	150
105	i		SHIFT I	151
106	j		SHIFT J	152
107	k		SHIFT K	153
108	l		SHIFT L	154
109	m		SHIFT M	155
110	n		SHIFT N	156
111	o		SHIFT O	157
112	p		SHIFT P	160
113	q		SHIFT Q	161
114	r		SHIFT R	162
115	s		SHIFT S	163
116	t		SHIFT T	164
117	u		SHIFT U	165
118	v		SHIFT V	166
119	w		SHIFT W	167
120	x		SHIFT X	170
121	y		SHIFT Y	171
122	z		SHIFT Z	172
123	{	Left Brace	SHIFT [[	173
124	|	Vertical Line	SHIFT | \	174
125	}	Right Brace	SHIFT]]	175
126	~	Tilde	SHIFT ~ `	176
127	DEL	Delete	Inaccessible from Keyboard	177

☆Multiple keys must be pressed simultaneously.


CLR→END
170
5
k0
134
k1
135
k2
136
k3
137
k4
138
k5
139
k6
REWIND
T14
140
k7
REWIND
T15
141
ME
1
→
247
ROLL ↓
173
0
k8
GET
150
k9
LOAD
151
k10
SAVE
152
k11
STORE
153
k12
EDIT
154
k13
EDIT LINE
155
k14
LIST
156
k15
SCRATCH
157
45
+
=
61
BACK SPACE
167
~
\
96
STOP
184
CLEAR LINE
185
E
69
(
40
)
41
^
94
P
80
{
[
91
}
]
93
|
\
92
RUN
200
RESULT
201
7
55
8
56
9
57
/
47
:
;
59
"
'
39
STORE
214
PAUSE
216
=
61
4
52
5
53
6
54
*
42
?
/
47
SHIFT
REPEAT
EXECUTE
249
1
49
2
50
3
51
−
45
CONT
248
0
48
.
46
,
44
+
43

Appendix E

System 45 I/O ROM Syntax

The following list presents a concise, formal syntax definition for all of the I/O ROM statements and functions.

Conventions and Terms

DOT MATRIX : All items in dot matrix must appear exactly as shown.

[ ] : Items within square brackets are optional unless the brackets are also in dot matrix.

| : A vertical line between two items reads as "or"; only one of the two items may be included.

... : Three dots indicate that successive parameters are allowed, when each is separated by a comma.

{ } : When more than one item appears in an item list with no separators, individual items are within braces.

source | dest: sc|string variable|numeric array

sc: isc|isc,da|hpa|-isc|−isc,da|−hpa

isc: interface select code

hpa: Three or four digit HP-IB device bus address sequence (for example 705,712.1014, etc.) See the description of da below for rules regarding the use of secondary commands.

da: One or two digit device address sequence (i.e.,5,12,03) separated by commas. Note that a secondary command sequence can follow any device address. The secondary command sequence is set off from the device address by a decimal point and terminated by a secondary command >31. The entire device address and secondary command expression is limited to twelve digits.

transfer: [type] [USING {image}] | [type] NO FORMAT

Etype: itfr|{htfr} N|tfr

N: Byte or word transfer count

type: itfr|htfr|tfr

itfr: {B|W} INT (Byte or word interrupt handshake)

htfr: {B|W} {FHS|DMA} (Byte or word fast handshake or direct-memory access)

tfr: WHS (word handshake)

list:

variable names
array identifiers
numeric expressions
string expressions

Syntaxes

ABORTIO isc

This statement is used to reset the interface functions of all HP-IB devices on the bus and return control to the System 45 if it is system controller.

BINAND (exp 1, exp 2)

This function returns the binary AND of the values specified by exp 1 and exp 2.

BINCMP (exp)

This function returns the binary complement of the value specified by exp.

BINEOR (exp 1, exp 2)

This function returns the binary EXCLUSIVE OR of the values specified by exp 1 and exp 2.

BINIOR (exp 1, exp 2)

This function returns the binary INCLUSIVE OR of the values specified by exp 1 and exp 2.

BIT (exp 1, {exp 2}|{string exp})

This function returns a value of 1 or 0 as follows:

- When exp 2 is a numeric expression, a 1 is returned when the bit position of exp 1 that is specified by exp 2 is a 1.

- When a string expression is specified, it is used as a mask for testing the bit pattern of exp 1. A 1 is returned when the string mask matches the bit pattern of exp 1. Any character of the mask that is not a 1 or 0 character represents a don't-care state for that bit.

CARD ENABLE isc

This statement enables the specified interface card to generate end-of-line program interrupts. The CONTROL MASK word is stored into the interface's R5 register by the CARD ENABLE statement.

CONFIGURE isc [TALK = da1] [LISTEN = da2[,da3...]]

This statement allows HP-IB data transfers not involving the System 45 to take place on the bus. If the TALK address parameter is excluded, the System 45 is assumed to be the talker; if the LISTEN address parameter(s) is excluded, the System 45 is assumed to be the listener.

CONTROL MASK isc; bit mask

This statement establishes an end-of-line branch mask byte. The numeric or string expression specified for the bit mask is converted to a 16 bit integer: bits set in the integer enable the corresponding end-of-line branch conditions from the interface. The string expression form of the bit mask consists of 1 and 0 characters.

CONVERT dest; "mode"{, string [, parity] } | { [, string], parity}
CONVERT dest; "mode", exp 1 TO exp 2 [, parity]

This statement establishes a conversion table and optionally generates and checks parity for I/O operations. The string specified indicates the conversion table to be used.

The mode parameter is specified as follows:

- I – input only conversions
- O – output only conversions
- IO – the conversion table specified is used for input, and an inverse table is generated for output.
- OI – the conversion table specified is used for output, and an inverse table is generated for input.

The parity parameter is a numeric value specified as follows:

- 0 – Parity bit is always reset (0).

- 1 – Parity bit is always set (1).
- Even value (≠0) – Even parity is generated and checked.
- Odd value (≠ 1)– Odd parity is generated and checked.

Single character conversions can be specified by the second syntax, CONVERT...TO... Successive statements can be executed, and their values are placed in the appropriate position of the conversion table generated by the first CONVERT...TO...statement for that select code.

CONVERT dest; mode

This statement turns off conversions previously specified for the dest parameter.

[S] ENTER {sc} {Etype} [NOFORMAT USING image]; enter list
[S] ENTER source [BYTE] [USING image]; enter list

This statement is used to initiate a transfer of data into the specified list. Two basic syntaxes are available as shown above, with parameter meanings as follows:

BYTE – This optional keyword is used for internal transfers only. Specifying BYTE causes one byte per word to be transferred from the source string or array, whereas omitting BYTE causes two bytes per word to be transferred from the source string or array.

image – valid input IMAGE specifiers are:

F Numeric freefield input using a decimal point radix symbol. Leading spaces are ignored, non-numeric characters are delimiters.

H Numeric freefield input using a comma for the radix symbol. Otherwise identical to F.

{n}N Numeric input of {n} characters per data item with a decimal point radix symbol. Non-numeric characters are counted but not entered.

{n}G Numeric input of {n} characters per data item with a comma radix symbol. Otherwise identical to N.

B Binary input of 8 bits per numeric or numeric array variable.

Y Binary input of two 8 bit bytes per numeric or numeric array variable. The first byte received becomes the 8 most significant bits of the variable.

W Binary input of 16 bits per numeric or numeric array variable.

For the B, Y, and W specifiers, if the input variable is a full or short precision real variable, the incoming binary value is converted to the appropriate data type.

T String variable freefield input with a line-feed delimiter. Carriage-returns not immediately followed by a line-feed are entered into the string. Input to a variable terminates with a line-feed or when the dimensioned string length is exceeded.

{n}A String variable input of {n} characters.

{n}X Causes {n} input characters to be skipped.

{n}/ Causes all characters up to the next {n} line-feeds to be skipped.

# Causes line-feed to be canceled as terminator, and must appear before any other image specification in the image list. Data entry terminates with the last item in the list or EOI.

+ Cancels the HP-IB EOI as a terminator and must appear before any other image specification in the image list. Data entry terminates when the last item in the enter list is received.

% Cancels both EOI and line-feed as terminators, and must appear before any other items in the image list.

{n}( ) Items or groups of items enclosed in parentheses are replicated {n} times.

enter list - Allowable items in the enter list include:

- Full precision variables
- Short precision variables
- String variables
- String array variables
- Full or short precision numeric arrays
- Integers

The optional form SENTER specifies sequential execution of the ENTER statement with respect to other SENTER and SOUTPUT statements when the program is the OVERLAP mode of execution.

EOL isc [; {sequence[, delay] } delay]

This statement replaces the default carriage-return line-feed that is sent for the L image specifier of the OUTPUT image reference with the specified sequence. The delay parameter specifies the milliseconds of delay before initiating another line of output. The EOL sequence can be specified by either a string variable or expression.

IOFLAG (isc)

This function returns a value of 1 when the specified interface is ready: a 0 indicates the interface is busy.

IOSTATUS (isc)

This function returns the state of the interface status line: a 1 indicates the peripheral is operational, a 0 indicates an error condition.

LASTBIT

This function returns the state of the last bit shifted or rotated out of the word specified in the ROTATE or SHIFT binary functions.

LOCAL sc

This statement puts the specified HP-IB devices back into their local state. Local Lockout is not cancelled.

LOCAL isc

This statement puts all devices on the bus back into their local state and cancels an existing Local Lockout state.

LOCAL LOCKOUT isc

This statement sends the Local Lockout message, which prevents an operator from returning a device to local control from its front panel. The System 45 must be the active controller to execute the LOCAL LOCKOUT statement.

OFF INT #isc

This statement cancels the ON INT condition for the specified interface.

ON INT #isc [, priority] CALL {label}

This statement enables end-of-line program branches for the specified interface. The priority parameter sets the priority level for the end-of-line branch, and the system priority level is set to the level of the end-of-line branch for the duration of the subprogram. Program transfer is to the subprogram specified by {label}.

ON INT #isc [, priority] GO SUB {line identifier}

This statement enables end-of-line program branches for the specified interface. The priority parameter sets the priority level for the end-of-line branch, and the system priority level is set to the level of the end-of-line branch for the duration of the subroutine. Program transfer is to the subroutine at the specified line identifier.

ON INT #isc [, priority] GO TO {line identifier}

This statement enables end-of-line program branches for the specified interface. The priority parameter sets the priority level for the end-of-line branch. System priority is not redefined, and program transfer is to the specified line identifier.

[S]OUTPUT {dest} {transfer} ; data list

This statement transfers data in the list to the specified destination. Items in the data list are separated by commas or semicolons and include:

- Full precision variables
- Short precision variables
- String variables
- String arrays
- Numeric arrays
- Integers

If a variable-to-variable OUTPUT is specified, the optional keyword BYTE can be included in the statement as the transfer type. This results in each byte of data in the data list being transferred to a word (16 bits) of the destination variable.

OUTPUT image specifiers must be separated by a comma, @; or slash, and include the following:

{n}D Specifies {n} digit positions with a blank fill character.

{n}Z Specifies {n} digit positions with a fill character of zero.

{n}* Specifies {n} digit positions with a fill character of an asterisk.

{n}X Causes {n} blanks to be printed.

{n}A Specifies {n} single string character positions.

. Indicates placement of a decimal point radix indicator. There may be only one radix indicator per numeric specifier.

R Indicates placement of a comma radix indicator. There may be only one radix indicator per numeric specifier.

C Indicates placement of a comma in a numeric specification. It is a conditional character and is output only if there is a digit to its left.

P Indicates placement of a period in a numeric specification. It is a conditional character and is output only if there is a digit to its left.

S Indicates a sign position for a + or −. The sign floats to the left of the leftmost significant digit if S appears before all digit symbols.

M Indicates a sign position; + is replaced by a blank. The sign floats to the left of the leftmost significant digit if M appears before all digit symbols.

E Causes output of an E, sign and two digit exponent. This is used for output of numbers in scientific notation.

K Specifies an entire string or numeric field. A numeric is output in STANDARD format, except that no leading or trailing blanks are output. The current value of a string is output.

Y Causes two bytes to be packed into the next two available bytes in the output buffer. If the source of data is a real variable, it is converted to an integer value and placed into the output buffer. Values must be greater than −32768 and less than 32767.

W Causes two bytes to be placed in the next available word boundary of the output buffer. If the source of data is a real variable, it is converted to an integer value and placed into the output buffer. Values must be greater than −32768 and less than 32767.

B Causes one byte of data to be placed into the output buffer. Values must be greater than −128 and less than 255.

{n}L Outputs an EOL sequence and delay.

+ Suppresses line-feed, and must precede any other image specifiers.

- Suppresses carriage-return, and must precede any other image specifiers.

Suppresses carriage-return/line-feed, and must precede any other image specifiers.

@ Outputs a form-feed.

{n}/ Causes a carriage-return/line-feed sequence to be output.

{n} ( ) Image specifiers can be replicated {n} times by enclosing the item or group of items in parentheses.

" " Text is enclosed in quotes.

PASS CONTROL sc

This statement can be used if the System 45 is the active controller on the HP-IB to specify another inactive controller on the bus to assume controller functions. The 9845A then takes the role of a device.

PPOLL (isc)

This function returns a byte representing the 8 Status Bit messages of those devices on the bus capable of responding to a parallel poll. The byte is returned in the form of a decimal value.

PPOLL CONFIGURE sc; mask

This statement programs the logical sense and data bus line on which the specified devices are to respond for a parallel poll. The mask can be specified as a string or numeric value: the three least significant bits determine the data bus line for the response, and the fourth bit determines the logical state of the response.

PPOLL UNCONFIGURE sc

This statement disables the parallel poll response of selected devices. By specifying only the interface select code, the parallel poll response of all bus devices is disabled.

READBIN (sc)

This function returns one 16 bit word from the specified interface. If the interface is an 8 bit interface, the most significant 8 bits are reset to 0.

READ IO isc, reg; var

This statement inputs a 16 bit value from the register specified by the {reg} parameter on the selected interface. The value is returned in the numeric variable specified by {var}.

REMOTE sc

This statement has two conditional effects:

- If individual devices are not specified, the remote state for all devices on the bus is enabled.
- If individual devices are specified, those devices being addressed are put into the remote state.

When the System 45 is switched on or reset, bus devices are automatically enabled for the remote state. When a device is addressed to listen, it automatically switches to the remote state.

REQUEST isc; exp

This statement is used when the System 45 is not the active controller on the bus and requires service from the active controller. The numeric or string expression {exp} is sent in response to a serial poll from the active controller. If {exp} is a numeric expression, it must have a value between 0 and 127: as a string expression, it must consist of the characters 1 and 0. Bit 6 of {exp} should be set to specify that the System 45 requested service.

RESET sc

This statement has four effects depending on the manner in which the select code is specified:

- If the select code of a non-HP-IB type interface is specified, the Reset bit (bit 5) of the interface control register (R5) is set true (logical 1).
- If the select code specifies an HP-IB interface with no device addresses, a Device Clear (DCL) message is sent.
- If the select code specifies an HP-IB interface and one or more HP-IB device addresses, the specified devices are sent a Selected Device Clear (SDC) message.
- If the negative select code of a 98034A interface is specified (i.e. RESET-7), those devices addressed to listen from a prior bus configuration are sent a Selected Device Clear (SDC) message.

ROTATE (exp 1, exp 2)

This function rotates the 16 bit binary value specified by {exp 1} as specified by {exp 2}:

- If {exp 2} is positive, the binary value of {exp 1} is rotated to the right by {exp 2} bit positions.
- If {exp 2} is negative, the binary value of {exp 1} is rotated to the left by {exp 2} bit positions.

The last bit shifted out of the word is saved in the save bit, which can be accessed by using the LASTBIT function.

SELECT CODE isc ACTIVE
SELECT CODE isc INACTIVE

These statements activate or deactivate I/O activities on the specified interface. I/O statements and functions are affected as follows:

Statement	Result
OUTPUT =	No Output
ENTER =	No Change
READBIN =	0
IOSTATUS =	1
IOFLAG =	1
STATUS =	0
PPOLL =	0

SENDBUS sc; commands [; data [; commands [; data]]]...

This statement provides complete HP-IB programming flexibility. The commands parameter can specify talker and listener addresses, multiline universal commands, and addressed and secondary commands. All commands are sent with the ATN line set true; no parity is generated. The data parameter can specify any desired device-dependent information; all items in the data parameter are sent with the ATN line set false.

Individual numeric items in the commands and data fields are separated by commas.

SENTER (See ENTER)

SET TIMEOUT isc; exp

This statement establishes a minimum time limit of {exp} milliseconds for the System 45 to wait for a peripheral to respond to an input or output operation. An end-of-line branch is requested when the timeout limit is exceeded.

SHIFT (exp 1, exp 2)

This function shifts the 16 bit binary value specified by {exp 1} by the number of bit positions specified by {exp 2}:

- If exp 2 is a positive value, {exp 1} is shifted to the right.

- If exp 2 is a negative value, {exp 1} is shifted to the left.

The last bit shifted out of the {exp 1} is put into the save bit. This bit can be accessed by using the LASTBIT function.

SOUTPUT (See OUTPUT)

STATUS sc; var 1 [, var 2 [, var 3 [, var 4]]]

This statement returns the interface or device status into the variables as follows:

- If the specified interface is not a 98034A, only 1 byte of status is returned. The status bit patterns are illustrated in this manual in the section covering the STATUS statement.
- If the specified interface is a 98034A, then up to 4 status bytes are returned:

  var 1 = fourth interface status byte

  (The following three variables are optional and need not be specified in the STATUS statement.)

  var 2 = first status byte of the 98034A interface.

  var 3 = second status byte of the 98034A interface.

  var 4 = third status byte of the 98034A interface.
- If an HP-IB device address is specified, the serial poll status byte of the device is returned in {var 1}.

SYSTEM TIMEOUT OFF

This statement disables the System 45 message:

DEVICE TIMEOUT ON SELECT CODE n

All select codes are affected.

SYSTEM TIMEOUT ON

This statement enables the System 45 message:

DEVICE TIMEOUT ON SELECT CODE n

All select codes are affected.

TIMEOUT (isc)

The timeout function returns a 1 or 0 value dependent on the cause of the end-of-line branch. A value of 1 indicates the cause of the end-of-line branch was a device timeout; a 0 value indicates the branch was not a result of a device timeout.

TRIGGER sc

This statement is used to initiate device-dependent action from either a selected device or all devices addressed to listen on the HP-IB.

- If only the select code of the interface is specified, all devices addressed to listen are triggered; the Group Execute Trigger (GET) message is sent on the bus.
- If one or more device addresses are specified, only those devices specified are triggered.

WRITE IO isc, reg #; var

This statement outputs the 16 bit binary value specified by {var} to the specified register(4-7) of the interface.

{reg #} = the specified interface register.

OUTPUT Image Specifiers

Image Specifier	Replication Allowed?	Purpose	Comments
X	Yes	Blank	Can go anywhere
" "	No	Text	Can go anywhere
D	Yes	Digit	Fill=Blanks
Z	Yes	Digit	Fill=Zeroes
*	Yes	Digit	Fill=Asterisks
S	No	Sign	“+” or “–”
M	No	Sign	“Δ” or “–”
E	No	Exponent	Format=ESDD
.	No	Radix	Output “.”
C	No	Comma	Conditional Number Separator
R	No	Radix	Output “,”
P	No	Decimal Point	Conditional Number Separator
A	Yes	Characters	Strings
(...)	Yes	Replicate	
L	Yes	Carriage Control	Output EOL Sequence
#	No	Carriage Control	Suppress CR-LF
+	No	Carriage Control	Suppress LF
–	No	Carriage Control	Suppress CR
K	No	Compact	Strings or Numerics
,	No	Delimiter	
/	Yes	Delimiter	Output CR-LF
@	No	Delimiter	Output FF
Y	No	Binary	Output Two Bytes
W	No	Binary	Output One 16 Bit Word
B	No	Binary	Output One 8 Bit Byte

ENTER Image Specifiers

Image Specifier	Replication Allowed?	Purpose	Comments
F	No	Freefield numeric	Decimal point radix
H	No	Freefield numeric	Comma radix
N	Yes	Digit	Decimal point radix
G	Yes	Digit	Comma radix
T	No	Freefield string	Line-feed delimiter
A	Yes	Character	String
B	No	Binary	Input One Byte
Y	No	Binary	Input Two Bytes
W	No	Binary	Input One 16 Bit Word
X	Yes	Skip character	String or Numeric
/	Yes	Skip record	Records delimited by "LF"
#	No	Cancel LF delimiter	Must be first specifier
+	No	Cancel EOI delimeter	Must be first specifier
%	No	Cancel LF and EOI	Must be first specifier
(...)	Yes	Replicate	

Special Considerations

Syntax

- An entire string array (`F$(*)`) is not allowed in a DMA or FHS type NOFORMAT transfer.

- There is no printer width specification (the WIDTH keyword) for the OUTPUT statement.

- The LIN, TAB, SPA, and PAGE print functions cannot be placed in the OUTPUT statement data list.

- The IOSTATUS and IOFLAG functions cannot be placed in the OUTPUT statement data list.

- The statement `ON INT 9`...will be stored as `ON INT (9)`...rather than `ON INT #9`...because the `#` character was left out of the syntax. This statement now is interpreted as a case statement (such as `ON X GOTO`...), where the `INT` is assumed to be the integer function.

Image Specifiers

- The EOL sequence only works with the `L` image specifier: it is not a pure CR-LF substitution.

- An invalid format specifier gives a `SYSTEM ERROR` at execution time. (An example would be using an ENTER image specifier to output a numeric variable.)

- The `B`, `Y`, and `W` specifiers cannot be of the form `2Y` for replication. Instead, use parentheses for replication, such as `2(Y)`.

- Numeric variables are rounded as necessary to fit the associated image specifier for the OUTPUT USING statement. (For example, if 100.325 is output using `DDD.DD`, the result is an output of 100.33.)

Terminators

- A numeric item in an enter list is not skipped if the first character received is a comma. This is different than numeric input for the 9825A, for example, which skipped a numeric item if a comma was received as the first character.

- Receipt of a horizontal tab does not cause data to be ignored until a LF character is received, as was done with the 9825A.

- A line-feed character does not terminate an ENTER statement. Data entry continues until either the enter list or the specified count is satisfied.

- Interrupt transfers (BINT,WINT) used with ENTER are terminated only by a line-feed or an EOI after the enter list has been satisfied, unless a #, +, or % image specifier is present in the image reference.

- Fast-handshake transfers (BFHS,WFHS) used with ENTER are not terminated by a special character, rather, the count parameter determines the number of words or bytes input.

- You must specify when a CR-LF sequence is to be output when sending long strings of data to a printer using the OUTPUT statement. This is necessary because there is no WIDTH keyword for OUTPUT.

- When inputting string data using ENTER...NOFORMAT..., the string input is only terminated when the string is filled. CR-LF does not terminate NOFORMAT string data input.

- FHS and DMA input transfers (ENTER) require a LF character to delimit string variables. If the input data (‘‘count’’ number of bytes or words) does not contain a LF for each string, an ERROR 153 results.

- The only way to send EOI with the last OUTPUT data item is to do Interface control to set EOI on the interface, then send the last data byte.

- The statement OUTPUT...USING"...L"; data list outputs a CR-LF after the EOL sequence. Change the statement to OUTPUT...USING"#...,L"; data list to inhibit the CR-LF.

- To use the ENTER statement to input a single byte or word, the # image specifier must be used with the B, Y, or W image specifier, or the statement will continue to wait for a LF to terminate the input operation.

- The #, +, and % image specifiers only eliminate the end of transfer terminator requirement of the ENTER statement. Each string variable in the enter list that is input using the T (or default) image specifier requires a LF to delimit the string.

A default OUTPUT data transfer outputs a CR-LF every 256 bytes.

Execution

- You cannot execute the IOSTATUS or IOFLAG functions to the system printer or printall device.

- A default transfer type (BYTE or WHS) can overlap with previous INT and DMA transfers, but prevents new INT and DMA transfers from being initiated until the default transfer completes.

- A card enable to a 98032A interface causes an immediate interrupt because the card is ready. To prevent this, make the card not ready by executing a WRITEIO to register 7 before the card enable.

- The default format for OUTPUT is identical to the default format for PRINT. A comma outputs a data item in a 20 character field with trailing blank fill, and a semicolon suppresses trailing blanks and/or the final CR-LF.

- Setting the System 45 up as both talker and listener (the CONFIGURE statement) causes the System 45 to hang up.

- All HP-IB commands with the exception of SENDBUS commands are sent with odd parity, regardless of the parity specified by a CONVERT statement. You should also note that the System 45 talk and listen addresses are sent with the parity of the primary address.

- An ENTER or OUTPUT operation to a 98032A interface enabled to interrupt on becoming ready, causes the interface to interrupt.

- A SYSTEM ERROR is caused if an ENTER image specifier is used for OUTPUT.

- ENTER interrupt transfers do not overlap if a numeric variable is in the enter list.

- If a variable does not fit the corresponding OUTPUT image format (for example, outputting 999.4 using DD.D), the format in error is printed to show the error.

- The 98034A card does not allow the EOI interrupt condition to initiate an end-of-line branch. You must use the STATUS statement to check for EOI.

- If an ENTER interrupt transfer is done concurrently with other I/O operations, the system can run out of buffer space. At this point, the System 45 pauses until the ENTER interrupt transfer completes.

- If an ENTER operation into a string is performed with no parity specified, all 8 bits are input. This can cause inverse video and other anomalies to be displayed or printed when the string is output to the CRT or printer. Specifying parity causes the eighth bit to be reset when the character is entered into the string, eliminating the display anomalies caused by the eighth bit being set.

- An inactive select code (SELECT CODE INACTIVE statement) does not change enter list variables. The values returned by other functions are:
  1. READBIN returns 0.
  2. PPOLL returns 0.
  3. IOSTATUS returns 1.
  4. IOFLAG returns 1.

- Attempting to output data from one interface and entering that same data simultaneously from another interface results in a machine deadlock, even in OVERLAP mode.

- An OUTPUT to multiple devices (on HP-IB) uses the CONVERT and EOL specified for the first parameter in the select code sequence.

  For example:

```
10   !  EXAMPLE STATEMENTS OF SELECT CODE EFFECT ON EOL AND CONVERT.
20   !
30   CONVERT 701;"O",A$         ! SET UP CONVERSION FOR DEVICE 01.
40   CONVERT 705;"O",B$         ! SET UP CONVERSION FOR DEVICE 05.
50   EOL 701;D$,100             ! SET UP EOL SEQUENCE FOR DEVICE 01.
60   EOL 705;E$,15              ! SET UP EOL SEQUENCE FOR DEVICE 05.
70   OUTPUT 701,705;Data$       ! OUTPUT Data$ TO DEVICES 01,05.
80   OUTPUT 705,701;Data$       ! OUTPUT Data$ TO DEVICES 05,01.
90   OUTPUT 7,1,5;Data$         ! OUTPUT Data$ TO DEVICES 01,05.
100  END
```

1. Line 70 outputs `Data$` using the CONVERT of line 30 and EOL of line 50. Identical data is sent to devices 701 and 705.
2. Line 80 outputs `Data$` using the CONVERT of line 40 and EOL of line 60. Identical data is sent to devices 705 and 701.
3. Line 90 outputs `Data$` using neither CONVERT nor EOL, because the first select code parameter is an interface select code, and is not either of the devices specified by the CONVERT or EOL statements.

- DMA and FHS type input transfers (ENTER) that are executed for an inactive select code do not allocate all the memory necessary for an actual data transfer from an active select code. Running the same program but changing the select code from inactive to active could result in `ERROR 2`, a memory overflow error.

- Establishing a timeout limit (SET TIMEOUT statement) for a select code has different effects based on the type of transfer being used:

  1. INT, FHS, and DMA transfers only check for a "not ready" timeout limit with the first datum transferred.
  2. Default (BYTE) and WHS transfers check for a "not ready" timeout limit with each datum transferred.

- Hitting the STOP key during interrupt transfers has either of two effects depending on the direction of the transfer:

  1. OUTPUT interrupt transfers (BINT,WINT) stop the output at the end of a line of output.
  2. ENTER interrupt transfers (BINT,WINT) stop the input at the completion of the entire transfer.

- Hitting the STOP key during a DMA or FHS ENTER operation does not halt the transfer until all data has been input.

Subject Index

HEWLETT hp PACKARD

SALES & SERVICE OFFICES

AFRICA, ASIA, AUSTRALIA

ANGOLA
Telectra
Empresa Técnica de
Equipamentos
Eléctricos, S A R L
R Barbosa Rodrigues, 42-I OT
Caixa Postal, 6487
Luanda
Tel 35515/6
Cable TELECTRA Luanda

AUSTRALIA
Hewlett-Packard Australia
Pty Ltd
31-41 Joseph Street
Blackburn, Victoria 3130
P O Box 36
Doncaster East, Victoria 3109
Tel 89-6351
Telex 31-024
Cable HEWPARO Melbourne

Hewlett-Packard Australia
Pty Ltd
31 Bridge Street
Pymble
New South Wales, 2073
Tel 449-6566
Telex 21561
Cable HEWPARD Sydney

Hewlett-Packard Australia
Pty Ltd
153 Greenhill Road
Parkside, S A , 5063
Tel 272-5911
Telex 82536 ADEL
Cable HEWPARD ADELAID

Hewlett-Packard Australia
Pty Ltd
141 Stirling Highway
Nedlands, W A 6009
Tel 86-5455 ·
Telex 93859 PERTH
Cable HEWPARD PERTH

Hewlett-Packard Australia
Pty Ltd
121 Wollongong Street
Fyshwick, A C T 2609
Tel 95-2733
Telex 62650 Canberra
Cable HEWPARD CANBERRA

Hewlett Packard Australia
Pty Ltd
5th Floor
Teachers Union Building
495-499 Boundary Street
Spring Hill, 4000 Queensland
Tel 229-1544
Cable HEWPARD Brisbane

GUAM
Medical/Pocket Calculators Only
Guam Medical Supply, Inc
Jay Ease Building, Room 210
P D Box 8947
Tamuning 96911
Tel 646-4513
Cable EARMED Guam

HONG KONG
Schmidt & Co (Hong Kong) Ltd
P O Box 297
Connaight Centre
39th Floor
Connaught Road, Central
Hong Kong
Tel H-255291-5
Telex 74766 SCHMC HX
Cable SCHMIDTCO Hong Kong

INDIA
Blue Star Ltd
Kasturi Buildings
Jamshedji Tata Rd
Bombay 400 020
Tel 29 50 21
Telex 001-2156
Cable BLUEFROST

Blue Star Ltd
Sahas
414/2 Vir Savarkar Marg
Prabhadevi
Bombay 400 025
Tel 45 78 87
Telex 011-4093
Cable FROSTBLUE

Blue Star Ltd
Band Box House
Prabhadevi
Bombay 400 025
Tel 45 73 01
Telex 011-3751
Cable BLUESTAR

Blue Star Ltd
7 Hare Street
P O Box 506
Calcutta 700 001
Tel 23-0131
Telex 021-7655
Cable BLUESTAR

Blue Star Ltd
7th & 8th Floor
Bhandari House
91 Nehru Place
New Delhi 110024
Tel 634770 & 635166
Telex 031-2463
Cable BLUESTAR

Blue Star Ltd
Blue Star House
11/11A Magarath Road
Bangalore 560 025
Tel 55668
Telex 043-430
Cable BLUESTAR

Blue Star Ltd
Meeakshi Mandiran
xxx/1678 Mahatma Gandhi Rd
Cochin 682 016
Tel 32069,32161,32282
Telex 0885-514
Cable BLUESTAR

Blue Star Ltd
1-1-117/1
Sarojini Devi Road
Secunderabad 500 003
Tel 70126, 70127
Cable: BLUEFROST
Telex: 015-459

Blue Star Ltd
2 34 Kodambakkam High Road
Madras 600034
Tel 82056
Telex 041-379
Cable BLUESTAR

INDONESIA
BERCA Indonesia P T
P O Box 496/Jkt
JLN.Abdul Muis 62
Jakarta
Tel 40369, 49886,49255,356038
JKT 42895
Cable BERCACON

BERCA Indonesia P t
63 JL Raya Gubeng
Surabaya
Tel 44309

ISRAEL
Electronics & Engineering Div
of Motorola Israel Ltd
17 Kremenetski Street
P O Box 25016
Tel-Aviv
Tel 38973
Telex 33569
Cable BASTEL Tel-Aviv

JAPAN
Yokogawa-Hewlett-Packard Ltd
Ohashi Building
59-1 Yoyogi 1-Chome
Shibuya-ku, **Tokyo** 151
Tel 03-370-2281/92
Telex 232-2024YHP MARKET
TOK 23-724
Cable YHPMARKET

Yokogawa-Hewlett-Packard Ltd
Chuo Bldg., 4th Floor
4-20, Nishinakajima 5-chome
Yodogawa-ku, Osaka-shi
Osaka,532
Tel 06-304-6021

Yokogawa-Hewlett-Packard Ltd
Nakamo Building
24 Kami Sasajima-cho
Nakamura-ku, **Nagoya** , 450
Tel (052) 571-5171

Yokogawa-Hewlett-Packard Ltd
Tanigawa Building
2-24-1 Tsuruya-cho
Kanagawa-ku
Yokohama, 221
Tel 045-312-1252
Telex 382-3204 YHP YOK

Yokogawa-Hewlett-Packard Ltd
Mito Mitsu Building
105, Chome-1,San-no-maru
Mito, Ibaragi 310
Tel 0292-25-7470

Yokogawa-Hewlett-Packard Ltd
Inoue Building
1348-3, Asahi-cho, 1-chome
Atsugi, Kanagawa 243
Tel 0462-24-0452

Yokogawa-Hewlett-Packard Ltd
Kumagaya Asahi
Hackijuni Building
4th Floor
3-4, Tsukuba
Kumagaya, Saitama 360
Tel 0485-24-6563

KENYA
Technical Engineering
Services(E A)Ltd .
P O Box 18311
Nairobi
Tel 557726/556762
Cable PROTON

Medical Only
International Aeradio(E A)Ltd .
P O Box 19012
Nairobi Airport
Nairobi
Tel 336055/56
Telex 22201 22301
Cable INTAERIO Nairobi

KOREA
Samsung Electronics Co , Ltd
20th Fl Dongbang Bldg 250, 2-KA
C P O Box 2775
Taepyung-Ro, Chung-Ku
Seoul
Tel (23) 6811
Telex 22575
Cable ELEKSTAR Seoul

MALAYSIA
Teknik Mutu Sdn Bhd
2 Lorong 13/6A
Section 13
Petaling Jaya, **Selangor**
Tel 54994 54916
Telex MA 37605

Protel Engineering
P O Box 1917
Lot 259, Satok Road
Kuching, **Sarawak**
Tel 2400
Cable PROTEL ENG

MOZAMBIQUE
A N Goncalves, Lta
162, 1 Apt 14 Av O Luis
Caixa Postal 107
Lourenco Marques
Tel 27091, 27114
Telex 6-203 NEGON Mo
Cable NEGON

NEW ZEALAND
Hewlett-Packard (N Z) Ltd
P O Box 9443
Courtenay Place
Wellington
Tel 877-199
Cable HEWPACK Wellington

Hewlett-Packard (N Z) Ltd
Pakuranga Professional Centre
267 Pakuranga Highway
Box 51092
Pakurenga
Tel 569-651
Cable HEWPACK,Auckland

Analytical/Medical Only
Medical Supplies N Z Ltd
Scientific Division
79 Carlton Gore Rd , Newmarket
P O Box 1234
Auckland
Tel 75-289
Cable OENTAL Auckland

Analytical/Medical Only
Medical Supplies N Z Ltd
P O Box 1994
147-161 Tory St
Wellington
Tel 850-799
Telex 3858
Cable OENTAL, Wellington

Analytical/Medical Only
Medical Supplies N Z Ltd
P O Box 309
239 Stanmore Road
Christchurch
Tel 892-019
Cable DENTAL, Christchurch

Analytical/Medical Only
Medical Supplies N Z Ltd
303 Great King Street
P O Box 233
Dunedin
Tel 88-817
Cable DENTAL, Dunedin

NIGERIA
The Electronics
Instrumentations Ltd
N6B/770 Oyo Road
Oluseun House
P M B 5402
Ibadan
Tel 61577
Telex 31231 TEIL Nigeria
Cable THETEIL Ibadan

The Electronics Instrumentations Ltd
144 Agege Motor Road, Mushin
P O Box 6645
Lagos
Cable THETEIL Lagos

PAKISTAN
Mushko & Company, Ltd
Oosman Chambers
Abdullah Haroon Road
Karachi-3
Tel 511027 512927
Telex 2894
Cable COOPERATOR Karachi

Mushko & Company Ltd
38B, Satellite Town
Rawalpindi
Tel 41924
Cable FEMUS Rawalpindi

PHILIPPINES
The Online Advanced
Systems Corporation
Rico House
Amorsolo cor Herrera Str
Legaspi Village, Makati
Metro **Manila**
Tel: 85-35-81, 85-34-91
Telex 3274 ONLINE

RHODESIA
Field Technical Sales
45 Kelvin Road North
P O Box 3458
Salisbury
Tel 705231 (5 lines)
Telex RH 4122

SINGAPORE
Hewlett-Packard Singapore
(Pte.) Ltd
1150 Depot Road
Alexandra P O Box 58
Singapore 4
Tel 270-2355
Telex HPSG RS 21486
Cable HEWPACK, Singapore

SOUTH AFRICA
Hewlett-Packard South Africa
(Pty), Ltd
Private Bag Wendywood
Sandton, Transvaal 2144
Hewlett-Packard Centre
Daphne Street, Wendywood,
Sandton, Transvaal 2144
Tel 802-10408
Telex 8-4782
Cable HEWPACK JOHANNESBURG

Service Department
Hewlett-Packard South Africa
(Pty), Ltd
P O Box 39325
Gramley, Sandton, 2018
451 Wynberg Extension 3
Sandton, 2001
Tel 636-8188/9
Telex 8-2391

Hewlett-Packard South Africa
(Pty), Ltd
P O Box 120
Howard Place, Cape Province 7450
Pine Park Centre, Forest Drive
Pinelands, Cape Province, 7405
Tel 53-7955 thu 9
Telex 57-0006

Service Department
Hewlett-Packard South Africa
(Pty.), Ltd
P O Box 37099
Overport, Durban 4067
Braby House
641 Ridge Road
Durban, 4001
Tel 88-7478
Telex: 6-7954

TAIWAN
Hewlett-Packard Far East Ltd .
Taiwan Branch
39 Chung Hsiao West Road
Sec 1, 7th Floor
Taipei
Tel 3819160-4
Cable HEWPACK TAIPEI

Hewlett-Packard Far East Ltd
Taiwan Branch
68-2, Chung Cheng 3rd Road
Kaohsiung
Tel (07) 242318-Kaohsiung

Analytical Only
San Kwang Instruments Co , Ltd ,
No 20, Yung Sui Road
Taipei
Tel 3715171-4 (5 lines)
Telex 22894 SANKWANG
Cable SANKWANG TAIPEI

TANZANIA
Medical Only
International Aeradio (E A.), Ltd
P O Box 861
Dar es Salaam
Tel 21251 Ext 265
Telex 41030

THAILAND
UNIMESA Co., Ltd
Elcom Research Building
2538 Sukumvit Ave
Bangkok
Tel 3932387, 3930338
Cable UNIMESA Bangkok

UGANDA
Medical Only
International Aeradio(E A), Ltd.,
P O Box 2577
Kampala
Tel 54388
Cable INTAERIO Kampala

ZAMBIA
R J Tilbury (Zambia) Ltd
P O Box 2792
Lusaka
Tel 73793
Cable ARJAYTEE, Lusaka

OTHER AREAS NOT LISTED, CONTACT:
Hewlett-Packard Intercontinental
3200 Hillview Ave
Palo Alto, California 94304
Tel (415) 493-1501
TWX 910-373-1267
Cable HEWPACK Palo Alto

CANADA

ALBERTA
Hewlett-Packard (Canada) Ltd
11620A - 168th Street
EdmontonT5M 3T9
Tel: (403) 452-3670
TWX 610-831-2431

Hewlett-Packard (Canada) Ltd
210,7220 Fisher St S E
Calgary T2H 2H8
Tel (403) 253-2713
Twx 610-821-6141

BRITISH COLUMBIA
Hewlett-Packard (Canada) Ltd
837 E Cordova Street
Vancouver V6A 3R2
Tel: (604) 254-0531
TWX: 610-922-5059

MANITOBA
Hewlett-Packard (Canada) Ltd
513 Century St
St James
Winnipeg R3H DL8
Tel (204) 786-7581
TWX: 610-671-3531

NOVA SCOTIA
Hewlett-Packard (Canada) Ltd
800 Windmill Road
Dartmouth B3B 1L1
Tel (902) 469-7820
TWX 610-271-4482 HFX

ONTARIO
Hewlett-Packard (Canada) Ltd
1020 Morrison Dr
Ottawa K2H 8K7
Tel (613) 820-6483
TWX 610-563-1636

Hewlett-Packard (Canada) Ltd
6877 Goreway Drive
Mississauga L4V 1M8
Tel (416) 678-9430
TWX 610-492-4246

QUEBEC
Hewlett-Packard (Canada) Ltd
275 Hymus Blvd
Pointe Claire H9R 1G7
Tel (514) 697-4232
TWX 610-422-3022
TLX 05-821521 HPCL

FOR CANADIAN AREAS NOT LISTED:
Contact Hewlett-Packard (Canada)
Ltd in Mississauga

CENTRAL AND SOUTH AMERICA

ARGENTINA
Hewlett-Packard Argentina
S A
Av Leandro N Alem 822 - 12
1001**Buenos Aires**
Tel 31-6063,4,5,6 and 7
Telex 122443 AR CIGY
Cable HEWPACK ARG

BOLIVIA
Casa Kavlin S A
Calle Potosi 1130
P D Box 500
La Paz
Tel 41530,53221
Telex CWC BX 5298,ITT 3560082
Cable KAVLIN

BRAZIL
Hewlett-Packard do Brasil
I e C Ltda
Avenida Rio Negro, 980
Alphaville
06400**Barueri** SP
Tel 429-3222

Hewlett-Packard do Brasil
I e C Ltda
Rua Padre Chagas, 32
90000-**Pórto Alegre**-RS
Tel (0512) 22-2998, 22-5621
Cable HEWPACK Porto Alegre

Hewlett-Packard do Brasil
I E C, Ltda
Rua Siqueira Campos, 53
Copacabana
20000-**Rio de Janeiro**
Tel 257-80-94-DDD (021)
Telex 391-212-1905 HEWP-BR
Cable HEWPACK
Rio de Janeiro

CHILE
Calcagni y Metcalfe Ltda
Alameda 580-Of 807
Casilla 2118
Santiago, 1
Tel 398613
Telex 3520001 CALMET
Cable, CALMET Santiago

COLOMBIA
Instrumentacidn
Henrik A Langebaek & Kier S A
Carrera 7 No 48-75
Apartado Aéreo 6287
Bogotá, I D E
Tel 69-88-77
Cable AARIS Bogotá
Telex 044-400

COSTA RICA
Científica Costarricense S A
Avenida 2 Calle 5
San Pedro de Montes de Oca
Apartado 10159
San Jose
Tel 24-38-20, 24-08-19
Telex 2367 GALGUR CR
Cable GALGUR

ECUADOR
Calculators Only
Computadoras y Equipos
Electrdnicos
P D Box 6423 CCI
Eloy Alfaro #1824,3 Piso
Quito
Tel 453482
Telex 02-2113 Sagita Ed
Cable Sagita-Quito

EL SALVADOR
Instrumentacion y Procesamiento
Electronico de el Salvador
Bulevar de los Heroes 11-48
San Salvador
Tel 252787

GUATEMALA
IPESA
Avenida La Reforma 3-48
Zona 9
Guatemale City
Tel 63627 64786
Telex 4192 Teletro Gu

MEXICO
Hewlett-Packard Mexicana
S A de C V
Av Periférico Sur No 6501
Tepepan, Xochimilco
Mexico 23, D F
Tel 905-676-4600

Hewlett-Packard Mexicana,
S A de C V
Ave Constitucidn No 2184
Monterrey, N L
Tel 48-71-32, 48-71-84
Telex 038-410

NICARAGUA
Roberto Terán G
Apartado Postal 689
Edificio Terán
Managua
Tel 25114, 23412,23454
Cable ROTERAN Managua

PANAMA
Electrdnico Balboa, S A
P O. Box 4929
Calle Samuel Lewis
Cuidad de Panama
Tel 64-2700
Telex 3483103 Curunda,
Canal Zone
Cable ELECTRON Panama

PERU
Compañla Electro Médica S A
Los Flamencos 145
San Isidro Casilla 1030
Lima 1
Tel 41-4325
Cable: ELMED Lima

PUERTO RICO
Hewlett-Packard Inter-Americas
Puerto Rico Branch Office
Calle 272,
No 203 Urb Country Club
Carolina 00924
Tel (809) 762-7255
Telex 345 0514

URUGUAY
Pablo Ferrando S A
Comercial e Industrial
Avenida Italia 2877
Casilla de Correo 370
Montevideo
Tel 40-3102
Cable RADIUM Montevideo

VENEZUELA
Hewlett-Packard de Venezuela
C A
P D Box 50933
Caracas 105
Los Ruices Norte
3a Transversal
Edificio Segre
Caracas 107
Tel 35-00-11 (20 lines)
Telex 25146 HEWPACK
Cable HEWPACK Caracas

FOR AREAS NOT LISTED, CONTACT:
Hewlett-Packard
Inter-Americas
3200 Hillview Ave
Palo Alto, California 94304
Tel (415) 493-1501
TWX 910-373-1260
Cable HEWPACK Palo Alto
Telex 034-8300, 034-8493

System 45 I/O ROM Error Messages

150 Improper select code.

151 A negative select code was specified that does not match present bus addressing.

152 Parity error.

153 Either insufficient input data to satisfy enter list or attempt to ENTER from source into source.

154 Integer overflow, or ENTER count greater than 32767 bytes or 16383 words.

155 Invalid interface register number. (Can only specify 4-7)

156 Improper expression type in READIO, WRITEIO, or STATUS list.

157 No line-feed was found to satisy ENTER image specifier or no line-feed record delimiter was found in 512 characters of input.

158 Improper image specifier or nesting image specifiers more than 4 levels deep.

159 Numeric data was not received for numeric enter list item.

160 Repetition of input character more than 32768 times.

161 Attempted to create CONVERT table or EOL sequence for source or destination variable which is locally defined in a subprogram.

162 Attempted to delete a nonexistent CONVERT table or EOL sequence.

163 I/O error, such as interface card not present, device timeout, or interface or peripheral failure. (Interface FLAG line = 0).

164 Transfer type specified is incorrect type for interface card.

165 An FHS or DMA type NOFORMAT transfer specifies a count that exceeds the size of the variable, or an image specifier indicates more characters than will fit in the specified variable.

166 A NOFORMAT FHS or DMA type transfer does not start on an odd-numbered character position, such as A$[3]..

167 Interface status error or an EOI was received on an HP-IB interface before ENTER list or image specification was satisfied.