



HP 9825 Desktop Computer I/O Control Reference

Manual Part No. 09825-90210
Microfiche No. 09825-99210



Hewlett-Packard Computer Systems Division
Herrenberger Straße 110/130, 7030 Böblingen, West Germany

The manual printing date and part number indicate its current edition. The printing date changes when a new edition is printed. (Minor corrections and updates which are incorporated at reprint do not cause the date to change.) The manual part number changes when extensive technical changes are incorporated.

May 1980...First Edition.

November 1980...Second Edition. Revised pages: B-1 thru B-23, C-1 thru C-10, D-9, D-10.

May 1985...Second Edition.

NOTICE

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

HEWLETT-PACKARD MAKES NO WARRANTY OF ANY KIND WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Hewlett-Packard shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance or use of this material.

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

HP Computer Museum
www.hpmuseum.net

For research and education purposes only.



Your I/O Control Reference

This reference gathers all current user documentation on controlling peripherals with the 9825 Desktop Computer. All operations available with the General I/O, Extended I/O, 9862 Plotter and 9872 (HP-GL) Plotter ROMs are covered here. Space is also provided for keeping the various operating notes and installation manuals supplied with HP interface cards and peripheral devices. A list of those manuals currently available is under each appropriate tabbed divider.

The 9825 Interfacing Concepts guide (09825-90060) accompanies this reference when part of the 9825B Manual Kit. Please keep the guide under the Interfacing References tab and refer there for details on selecting an interfacing card or writing a special I/O driver.

This reference replaces these previous 9825A manuals:

- General I/O Programming (09825-90024)
- Extended I/O Programming (09825-90025)
- 9862A Plotter Programming (09825-90023)
- 9872A Plotter Programming (09825-90026)

The Systems Programming ROM (98224A or 9825T) also provides I/O operations. See the Systems Programming chapter of your Operating and Programming Reference. If you have a 9825A Computer, refer to the Systems Programming manual (09825-90027).

We welcome your comments on this and other HP desktop computer manuals. You're invited to fill out a reply card at the back of the binder. Please don't hesitate to use a reply card if you find an error in a manual. If all reply cards are missing, direct your comments to:

Hewlett-Packard Company
PL 97 User Documentation
3404 East Harmony Road
Fort Collins, Colorado 80525

Programming Requirements

The information in this reference assumes you are familiar with programming in HPL, as covered in the 9825 Operating & Programming Reference.


Controlling external devices requires an I/O ROM, either the General I/O ROM or a specific Plotter ROM. The 9825A requires plug-in ROM cards, which are available in various ROM combinations (for example, General I/O and Extended I/O). The 9825B has the General I/O, Extended I/O, and both Plotter ROMs built in. All operations described in this reference are available with the standard 9825B Computer.

Memory Usage

Each I/O ROM uses some read/write memory when installed in the computer:

- General I/O ROM ... 56 bytes.
- Extended I/O ROM ... 94 bytes.
- 9862A Plotter ROM ... 70 bytes.
- 9872 (HP-GL) ROM ... 104 bytes.

This loss is reflected in the Total Memory size indicated under the 9825B Computer's paper-access lid. The Total Memory indicated on the 9825A does not reflect this loss.

To determine the actual memory size currently available, press **RESET** and **LIST** . The second number displayed is the current number of bytes available for program and data

storage.

Reference Preview

Chapter 1: Formatted I/O Operations

Covers the statements which handle data in specialized formats: write (wrt), read (red), format (fmt), conversion (conv) and list programs (list).

Chapter 2: HP-IB Control

Introduces the HP Interface Bus and describes all operations available for controlling and exchanging data with devices connected via the HP-IB.

Chapter 3: Binary I/O Operations

Explains the 16 binary statements and functions available with the General I/O and Extended I/O ROMS.

Chapter 4: More I/O

These Extended I/O operations are covered: the Autostart routine, error trapping (on err), time out (time), conversion tables (ctbl), parity checking (par) and interface control statements.

Chapter 5: Interrupt Control

The 9825 interrupt scheme is introduced and ways are shown for designing and controlling programmable interrupts.

Chapter 6: Buffered I/O

The I/O buffering structure is explained, followed by methods to set up and use various types of data buffers.

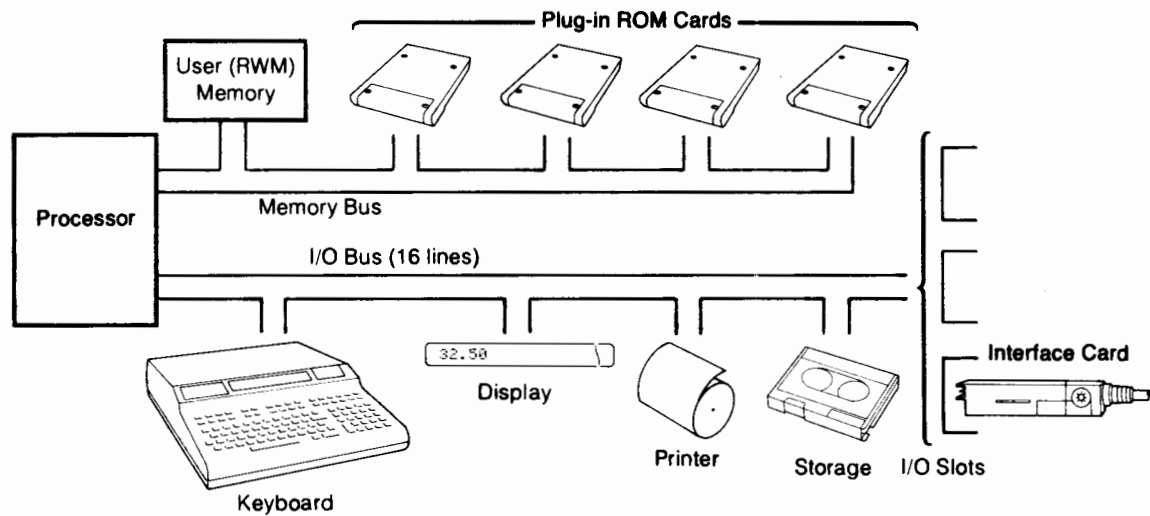
Chapter 7: Plotter Control

The 9862 and 9872 Plotter ROM operations are covered here. The 9872 ROM also allows controlling many other graphics devices responding to a standard set of HP graphics language (HP-GL) commands.

You'll find all reference tables, HPL syntax, and error codes in appendices at the back of this binder. A copy of the 9825 Technical Specifications is under the Interfacing References tab.

For a complete Table of Contents for each chapter, look under the appropriate tabbed divider.

The computer's I/O, or Input/Output, scheme is shown in the next figure. As shown, all data transfer between the computer processor, internal devices and peripheral devices is handled by an internal I/O bus. Each of these devices has been assigned a unique address, as explained later. All incoming data passes through the processor before it's stored in memory. Note that the term "I/O" is used with respect to the computer.



The 9825 Input/Output Scheme

Each external device must be connected to the computer via an appropriate interface card and cable. The card plugs into any I/O slot on the computer's back panel. A switch on the card allows setting it (and the peripheral device) to respond to a unique address, or select code.

Plug-in ROM cards provide extensions to the HPL language and plug into slots under the computer's keyboard. After installing a ROM card and switching the computer on, the ROM is an addressed part of the computer memory. (The 9825B already has many of these ROMs

built into its memory.) Since internal devices (printer, display and keyboard) and external

devices are connected via the same I/O bus, the internal devices respond to some General I/O operations in addition to their dedicated commands (prt, disp, etc.). These additional operations are covered in chapter 1.

Select Code

As just described, each external device is connected to the calculator via the same I/O bus. Since all external devices are “party-lined” on the same bus, each device is assigned a unique address, or select code, so that the correct device responds to each I/O operation.

For all external peripherals, the select code is an integer number from 2 through 15, which is specified in each I/O operation and decoded by the corresponding interface card. Each interface card has a switch permitting the user to set any one of many different codes. A list of recommended codes is in the Reference Tables appendix.

Each internal peripheral has a fixed select code which is automatically specified by standard calculator statements (display, print, etc.). Both the display and the keyboard respond to select code 0, the tape drive responds to select code 1, and the printer responds to select code 16.

The select code can be specified in the form of a constant, a variable, or an expression. Each of these write statements is addressed to the device responding to select code 9 \blacktriangleright

```
0: wrt 9, A, B, C
1: wrt S, R$(S = 9)
2: wrt 3r1, A, B(r1 = 3)
```

For some operations, other numbers can be combined with the select code to form a single expression. A general select code syntax is:

```
cc[dd[ee]][.f]  cc = one or two-digit select code.
                 dd = optional HP-IB address code (must be two digits). See chapter 2.
                 ee = optional HP-IB extended address code (use with Extended I/O
                     ROM). See chapter 2.
                 .f = format number (read and write statements only).
```


Input-Output Format

The I/O bus connecting the processor with internal and external peripherals contains 16 data lines. Data is transmitted in a 16-bit parallel, character-serial fashion. The I/O operations described in chapter 1 send and receive data in standard 8-bit (US)ASCII code. The computer sends and receives one 8-bit character at a time. The parity (most-significant) bit is not used with formatted I/O operations. You'll find a brief introduction to binary coding and conversion in chapter 3. It's important to know the I/O formats available with each interface card; see the card's installation and service manual for details.

Peripheral Interrupts

A peripheral interrupt is a signal from an external device requiring attention. For example, a voltmeter has taken a data reading and indicates that it's ready to transfer data to the computer. This capability is available when using certain interface cards (e.g., 98032A or 98034A HP-IB) in conjunction with Extended I/O operations. The interrupt control scheme and techniques are covered in chapter 5. More information can be found in the Interfacing Concepts guide. Peripheral interrupts cannot be controlled when only the General I/O ROM is available.

Syntax Conventions

The following general rules apply to the statements and functions described in this reference.

`dot matrix` All key words and characters in dot matrix must be entered as shown.

[] All syntax times in brackets are optional, as explained in the accompanying text.

... An ellipsis indicates that the preceding items can be replaced.

All General I/O and Extended I/O functions and statements can be executed from either the

keyboard or a program. A summary of HPL syntax is near the back of this binder.

Chapter 1

Table of Contents



Write Statement (wrt)	1-3
Delimiters	1-3
Free-field Output Format	1-4
Read Statement (red)	1-5
Free-field Input Format	1-6
Format Statements (fmt)	1-8
The Format Syntax	1-8
Data Output Specifications	1-9
Text Character Fields	1-11
Output Edit Fields	1-11
Output Edit Specifications	1-11
Printer Character Set	1-14
Display Character Set	1-17
Single Character Output	1-17
Text in Format and Write Statements	1-18
Data Input Specifications	1-19
Input Edit Specifications	1-21
Conversion Statement (conv)	1-23
List Statement (list)	1-23

Notes

Chapter 1

Formatted I/O Operations

This chapter describes the General I/O statements which handle data in specialized formats. If you are controlling HP 9800-series peripherals, additional instructions are provided in an operating note furnished with each HP 98032A Interface Card.

Write Statement

```
WRITE select code [ , format no.][ , expression or text, [ , expression or text2...]]
```

The write statement outputs characters, signs, and decimal point of each item to the specified peripheral. Each item in the list can be a numeric expression, text, or a string name (when the String ROM is in use). The value of each item is output in a free-field format unless a format statement is in effect. The format number can be an integer for 0 to 9 and references a similarly numbered format statement; format statements are described later.

Delimiters

A delimiter is a character that is used either to separate one expression from another inside the list or to terminate the list. The space (Δ) and the carriage-return line-feed (CR/LF) are delimiters that are automatically output during the execution of each write statement. The space is used to separate items within the list, and the CR/LF is used to terminate the list.

Free-Field Output Format

The free-field output format is automatically set whenever the calculator is switched on or **RESET** is pressed. Free-field is also set whenever run or erase all commands are executed. Each write statement references the free-field format until an appropriate format statement is executed; then the specifications in the format statement override free-field.

The free-field format causes each numeric expression to be output, right-justified, in an 18-character field. A CR/LF is given after every fourth expression output and after the last parameter is output. The form in which expressions appear is determined by the current fixed (**f x d**) setting. Characters within quotes and strings are output as "free text", which means that the 18-character fields are not used.

Here are some examples using free-field. The output device used here is an HP 9866B Printer, connected via a 98032A Interface set to select code 6. For more examples, see "Format Statements" in this chapter.

First, press **RESET**.

Then load and run this program \blacklozenge

The output is shown below.

```
0: fxd 4;10+A
1: wrt 6,A,A/9,
  A/A1e4
2: end
```

```
      10.0000      1.1111      0.0001 CR/LF
-----
18-char. field  18-char. field  18-char. field
```

Now change line 1 and run the program again.

```
0: fxd 4;10+A
1: wrt 6,A,A↑2,
  A↑4,A↑6,A↑8
2: end
```

```
      10.0000      100.0000      10000.0000      1000000.0000
1000000000.0000 CR/LF                                     CR/LF
```

Notice that a CR/LF is automatically output after the last item and after each four items.

To output text between numeric expressions, change line 1 again and run the program ♦

```
0: fxd 4;10→A
1: wrt 6;A;"mete
rs";A1e2;"cm";
A1e4;"mm"
2: end
```

```
10.0000meters 1000.0000cm 100000.0000mm CR/LF
w = 18 w = 18 w = 18
```

As shown in the printout, text is output between numeric fields with free-field. For complete control of numeric and text outputs, use format statements.

Read Statement


```
read select code[*, format no.]: variable1 [*, variable2...]
```

The read statement inputs and stores data from a specified peripheral¹. The number of variables in the list indicates how many data items to read. String variable names (but not substrings) can be used if the String ROM is in use. Each numeric data item can consist of the digits 0 through 9, plus and minus signs, a decimal point, and an "E" character (upper and lower case). All other characters are treated as input delimiters. The data item itself can assume the same form as any number entered from the keyboard.

The format number can be used to reference any of ten format statements. If a format number is not specified, and if a format statement has not been previously executed, a free-field input format is automatically used.

¹The calculator keyboard can not be used to input data with a read statement. Use an enter statement instead.

Free-Field Input Format

The free-field input format is set whenever the calculator is switched on or  is pressed. Free-field is also set whenever run or erase all commands are executed. Using free-field allows reading numeric data in virtually any form, provided that each item is followed by at least one delimiter (non-numeric character). For each variable in the read list, the calculator ignores all leading non-numeric characters until a numeric character is read. Then, after reading the data item, reading any non-numeric character terminates and stores the data item. Also, reading a LF terminates the entire read operation.

When free-field is used, the computer cannot input non-numeric characters unless a string variable (String ROM) is specified. Then all characters are input until the dimensioned string length is filled. Reading a LF (during free-field) automatically terminates reading a string. Also see "Format Statements" later in this chapter, and the String Variables chapter of the Operating and Programming Reference.

Here is a brief description of the delimiters used in the free-field input format:

- If the first character is a comma, the corresponding variable in the read statement is skipped and flag 13 is set. The skipped variable keeps its original value.
- All non-numeric characters preceding a data item are ignored. If a continuous string of the same non-numeric character is received, each variable in the read list is skipped and flag 13 is set after 2^{16} characters are read.
- An HT (horizontal tab) character causes the calculator to skip all characters until a LF is read.
- Whenever a LF is read (and it does not correspond to a preceding HT) the read statement is terminated and flag 13 is set.
- An upper- or lower-case "E" character, when part of any of these forms, causes the

preceding data item to be multiplied by the power of 10 indicated:

(data item)E(one or two digits)
(data item)E(+ or – and one or two digits)
(data item)E(space and one or two digits)

For example, any of these data items will be read as the number "1234" (Δ indicates space):

```
1.234E3
1.234E $\Delta$ 3
1.234E+3
1234000E-3
```

- Spaces read within numeric characters are ignored.
- The CR (carriage return) is always ignored (skipped over) during a read operation.

The following program can be used to input and print the data items on this ASCII-coded paper tape \blacklozenge

1.23,2.34,3.45,4.56,,5.67 (CR) (LF)



The program assumes that the tape reader interface is set to select code 3. Run the program twice to obtain both printouts.

```
0: 0+r1+r2+r3
1: red 0:r1,r2,
   r3
2: fxd 2
3: prt r1,r2,r3
4: spc 2!end
```

Notice that the read operation is terminated when the required number of items have been read, or when a LF is read. In the second printout, r2 was skipped and so retained its original value (the run command clears all variables).

```
1.23
2.34
3.45
```

```
4.56
0.00
5.67
```

As another example, use the previous program to read this tape \blacklozenge

12.81,13.35,(HT) 200.5,(LF) 25E5 ,



The computer ignores all data between HT and LF characters. More examples using read statements are in the next section and in Chapter 2.

```
12.80
13.35
2500000.00
```


Format Statements

Use of format statements provides more flexible and complete control of write and read statements. A format statement must be executed before the I/O statement referencing it and provides a list of specifications for use by the I/O statement. Then, as the I/O statement is executed, it references the last-executed format statement rather than free-field.

The Format Syntax

```
fnt[format no.][spec1[spec2...]]
```

The format number can be used to identify the statement for successive write or read statements. Each format number can be an integer constant from 0 to 9; if not specified, 0 is assumed. When the calculator is reset (power on, **RESET**, run, or erase all) format number 0 is automatically assigned to specify free-field. Also, the syntax `fnt[format no.]` assigns the format number to free-field.

For example, if the numbered format statements shown here appear in a program, then the write statements in lines 6 and 9 reference format no. 2, and the read statement in line 7 references format no. 9. The last two I/O statements reference free-field, however, since they do not specify a format no. and there is neither a format 0 nor an unnumbered format statement.

Each successive format statement replaces any previous, similarly-numbered format statement. A format statement with no parameters sets free-field. In the example sequence on the right, the first unnumbered format statement specifies free-field for the read statement. The write statements in lines 16 and 18 reference the second unnumbered format (line 14). The last two write statements reference the third unnumbered format (line 19).

Note that if an unnumbered format statement is used, all subsequent input or output operations that do not reference a numbered format statement will reference the unnumbered format statement. To avoid undesired referencing, all format statements should be numbered.

```
0: fnt 2,f10.2
1: fnt 5,e15.0
2: fnt 9,f6
.
.
.
6: wrt 6.2,A,B
7: red 3.9,r1
8: for I=1 to 9
9: wrt 16.2,rI
10: next I
11: wrt 16,A,B
12: red 7,A$

13: fnt ;red 3,
    A,B,C,D,E
14: fnt c50
15: fnt 1,5f10.0
16: wrt 6,A$
17: wrt 6.1,A,B,
    C,D,E
18: wrt 6,B$
19: fnt "end",5/
20: wrt 6
21: wrt 16;end
```

Data Output Specifications

Data specs determine the form in which each data item is output. Most data specs determine whether a number is output in fixed point or floating point, the number of digits to the right of the decimal point, and the character field width in which the number appears.

These data specs are available:



- [r] `f w. d` Specifies fixed-point format.
- [r] `e w. d` Specifies exponential (floating-point) format.
- [r] `fzw. d` Specifies fixed-point format with leading zeros in each field. Negative numbers are not allowed with this format.
- [r] `b` Specifies binary output of single characters. Examples begin on page 1-12.
- [r] `cw` Specifies a character field-width for either text or a string variable.

- `r` is an optional repeat factor (see examples). If `r` is omitted, 1 is assumed.
- `w` indicates total field width (in characters). If `w` is omitted, leading spaces are deleted from the field.
- `d` indicates the number of digits (0 through 11) to the right of the decimal point. If `d` is omitted or if `d` is greater than 11, the current fixed or float setting is used.
- `w`, `d` and `r` parameters must be positive integer constants.

For example, the data spec `f8.2` specifies a fixed-point number with two digits to the right of the decimal point. The number appears (right-justified) in an eight-character field. If `d` is 0, the decimal point is not output. A number output under a data spec is always rounded according to the number of decimal places specified. The free-field format is equivalent to `4f18`.

Some guidelines should be observed in selecting `w` and `d`. The minus sign, decimal point, and exponent are part of each number and must fit in the field width specified by `w`. For floating-point outputs, `w` should be greater than or equal to `d+7`. If the calculator cannot output the data in the field width available, the field is filled with `$$`.

The following examples of formatted output use the calculator's internal printer as the output device (select code 16).

1-10 Formatted I/O Operations

Lines 1 and 2 output three numbers. Since the entire format statement is referenced for each number, a CR/LF is output after each.

```
0: 10→A;.5→B;
   16→C
1: fmt f10.2
2: wrt C,A,B,
   AB↑2
```

```
10.00
 0.50
 2.50
```

w = 10

Lines 3 and 4 output the same numbers as above, but use of a repeat factor matches the data spec to the output list – so all numbers appear on the same line.

```
3: fmt 3f5.2
4: wrt C,A,B,
   AB↑2
```

```
10.00 0.50 2.50
```

w = 5 w = 5 w = 5

This sequence outputs numbers in various formats. Line 10 increments the format number, so that all formats are referenced. Outputs are shown below.

```
5: fmt f8.1,e8.1
6: fmt 1,4f4.0
7: fmt 2,4f.0
8: fmt 3,fz16.2
9: wrt C,A,2A↑2,
   3A↑3,4A↑4
10: if (C+.1→C) <
    16.41jnp -1
11: end
```

Format 0: Format 0 is referenced twice in line 9. Note that an upper-case E precedes the exponent.

```
10.0 2.0E 02
3000.0 4.0E 04
```

Format 1: Notice that \$s fill the last field since the number is too large for the field width specified.

```
10 2003000$$$$
```

Format 2: Notice that omitting w deletes leading spaces from each field.

```
10200300040000
```

Format 3: The last format outputs the same numbers in a single column with leading zeros in each field.

```
0000000000010.00
00000000000200.00
000000000003000.00
0000000000040000.00
```

Text Character Fields

The `c` data spec specifies a fixed character field for corresponding text or a string variable in the write statement. The characters are output right-justified in the field. If the field is too small, it's filled with `$s`.

In the example on the right, the first two lines of text fill the field; the third line is too short; the last line is too long.

```
0: fmt c10,f5.1
1: wrt 16,"Funct
   ion =",A
2: wrt 16,"Polar
   ity =",C
3: wrt 16,"Phase
   =",B
4: wrt 16,"Avera
   ge Value =",D+
   E>E
```

```
Function = 0.0
Polarity = 0.0
Phase = 0.0
$$$$$$$$$ 0.0
```

w = 10

Output Edit Specifications

These edit specs are used to control the placement of output data and to output character strings:

- [r] \times Outputs a blank character space.
- [r] \checkmark Outputs a CR/LF.
- [r] "TEXT" Outputs the ASCII characters within quotes. See the following examples and the Appendix.
- Ξ Suppresses the automatic CR/LF output after each write statement.

Any combination of specs can appear in the same format statement; each spec must be separated by a comma. Most of the specs can be duplicated `r` number of times by using the repeat factor.

For example, referencing this statement causes the first fixed-point field to appear twice, followed by four character spaces, and then another fixed-point field.

```
0: fmt 2f6.2,4x,
   f10.1
```

1-12 Formatted I/O Operations

The program on the right shows how to format a three-column table using the internal printer. You can use the same method, of course, with external output printers.

Suppose that the entered values are 5, 9, 15, 25, and 64. When formats 0, 1 and 2 are referenced in line 6, CR/LFs and the column headers are output. Notice that spaces are used to position the characters, and that a `b` data spec causes the binary value of decimal 29 to be output, printing the character `z`. (The complete character set is shown later.) Then line 7 references format 3 for the numeric outputs. Here, spacing is determined by the data field widths.

```

0: 1→A
1: ent r1,r2,r3,
   r4,r5
2: fmt 2/
3: fmt 1,2x,"n",
   3x,"n",b,3x,"1/
   n"
4: fmt 2,16"- "
5: fmt 3,f3.0,
   f6.0,f6.3
6: wrt 16;wrt
   16.1,29;wrt
   16.2
7: wrt 16.3,rA,
   rA↑2,1/rA;jmp
   (A+1→A)=6
8: wrt 16;end

```

n	n ²	1/n
5	25	0.200
9	81	0.111
15	225	0.067
25	625	0.040
64	4096	0.016

The next program, which produces a table of trigonometric values, uses a 9866B Printer. Although the program is similar in programming technique to the previous one, here are some exceptions (a sample printout is on the next page):

- Spacing between table columns is achieved by placing blocks of spaces between each number, rather than by adjusting the width of each data field.
- Notice the use of text (in line 5) to enclose the radians entries.

```

0: fmt 1,5/iwrt
  6.1
1: fmt 30x,"Trig
  onometric Table
  ",2/iwrt 6
2: deg;sf 14;
  0→A→X
3: fmt 2,6x,"Deg
  rees (Radians)"
  ,12x,"Sine",
  10x,"Cosine",
  12x,"Tangent",/
4: wrt 6.2
5: fmt 3,9x,f3.0
  ,"(",f4.2,")",
  15x,f6.3,9x,
  f6.3,10x,e10.3
6: wrt 6.3,X,πX/
  180,sin(X),cos(
  X),tan(X)
7: A+1→A
8: if A=3;0→A;
  fmt iwrt 6
9: if 180>X;X+
  10→X;sto 6
10: wrt 6.1
11: end

```

Trigonometric Table

Degrees (Radians)	Sine	Cosine	Tangent
0 (0.00)	0.000	1.000	0.000E 00
10 (0.17)	0.174	0.985	1.763E-01
20 (0.35)	0.342	0.940	3.640E-01
30 (0.52)	0.500	0.866	5.774E-01
40 (0.70)	0.643	0.766	8.391E-01
50 (0.87)	0.766	0.643	1.192E 00
60 (1.05)	0.866	0.500	1.732E 00
70 (1.22)	0.940	0.342	2.747E 00
80 (1.40)	0.985	0.174	5.671E 00
90 (1.57)	1.000	0.000	9.999E 99
100 (1.75)	0.985	-0.174	-5.671E 00
110 (1.92)	0.940	-0.342	-2.747E 00
120 (2.09)	0.866	-0.500	-1.732E 00
130 (2.27)	0.766	-0.643	-1.192E 00
140 (2.44)	0.643	-0.766	-8.391E-01
150 (2.62)	0.500	-0.866	-5.774E-01
160 (2.79)	0.342	-0.940	-3.640E-01
170 (2.97)	0.174	-0.985	-1.763E-01
180 (3.14)	0.000	-1.000	0.000E 00

Printer Character Set

The program on page 1-12 shows how to print a new character on the internal printer. As shown in the next program, the `b` data spec can be used to print any character in the printer's character set.

The program on the right prints numbers 0 through 127. It also outputs each number's binary-equivalent value, causing the printer to output its entire character set. A complete printout is shown.

```

0: wrt 16, "CHARA
  CTER SET"
1: fmt f3.0,6x,b
2: wrt 16,I,I
3: if (I+1>I)<12
  8: jmp -1
4: spc 2:end
  
```

Internal Printer Character Set

CHARACTER SET	
0	4
1	6
2	X
3	N
4	o
5	6
6	7
7	7
8	Δ
9	σ
10	(line feed)
11	λ
12	υ
13	(carriage return)
14	γ
15	θ
16	θ
17	Ω
18	δ
19	θ
20	α
21	θ
22	θ
23	θ
24	θ
25	θ
26	θ
27	θ
28	e
29	z
30	f

31	*
32	(space)
33	!
34	."
35	#
36	\$
37	%
38	&
39	'
40	(
41)
42	*
43	+
44	,
45	-
46	.
47	/
48	0
49	1
50	2
51	3
52	4
53	5
54	6
55	7
56	8
57	9
58	:
59	;
60	<
61	=
62	>
63	?

64	@	95	T
65	A	96	t
66	B	97	a
67	B	98	b
68	C	99	c
69	D	100	d
70	E	101	e
71	F	102	f
72	G	103	g
73	H	104	h
74	I	105	i
75	J	106	j
76	K	107	k
77	L	108	l
78	M	109	m
79	N	110	n
80	O	111	o
81	P	112	p
82	Q	113	q
83	R	114	r
84	S	115	s
85	T	116	t
86	U	117	u
87	V	118	v
88	W	119	w
89	X	120	x
90	Y	121	y
91	Z	122	z
92	[123	{
93]	124	
94	^	125	~
		126	
		127	



Referring to the printout, notice that most of the characters and decimal numbers correspond to ASCII codes (see the table in the Appendix). But notice that two codes, 10 and 13, do not generate printer characters. Instead, 10 causes the printer to linefeed and 13 is ignored.

1-16 Formatted I/O Operations

Another method of listing the printer character set is to print a string variable containing character values from 0 through 127. Here's the program and printout.

```

0: dim A$(128)
1: 0->I
2: char(I)->A$(I+
  1)
3: if (I+1->I)<12
  8) jmp -1
4: prt A$
5: end

```

```

      10  13
↓      ↓
40xNαβΓñΔσλμτϕ
θΩδΑαΑά0ó00€²£¥
!"#$%&'()*+,-./
0123456789:;<=>?
@ABCDEFGHIJKLMNO
PQRSTUVWXYZ[r]↑_
'abcdefghijklmno
parstuvwxyzñ|→Σ†

```

Notice that with a print statement, the printer generates characters for string values 10 and 13. But if the string is output to the printer using a write statement (see the next line and printout), the printer does a line feed for 10 and ignores 13.

```

4: wrt 16,A$

```

```

40xNαβΓñΔσ
λμτϕθΩδΑαΑά0ó00€
↑↑
12 14

```

This last printout shows another important point: The internal printer does line feeds during a print statement to accommodate long strings or text, but it does not automatically do line feeds for write or write binary statements. The second line in this last printout was printed because of the line feed automatically output after the write statement.

Similarly, line feeds must be sent to the display when using write or write binary statements. If not, output data will be lost. For example, execute this line:

```
wrt 0,"Keeper"
```

Keeper

Now add `fmt z;` and execute the line again:

```
fmt z;wrt 0,"Keeper"
```

␣

Text in Format and Write Statements

As shown in the preceding examples, text to be output can be included in format or write statements. For programming convenience, however, it's recommended that text **not be placed**: 1) in both format and write statements for a given output, and 2) as the last item in a write statement. The following examples show why.

Suppose that we wish to output this line on an external printer ("DM" indicates Deutsche Marks):

```
Sales =    10DM for the last    4 months.
```

└──────────┘
└───┘
w = 6
w = 3

This sequence could be used with no problem (assume that A = 10 and B = 4):

```
0: fmt "Sales =",f6.0,"DM for the last ",f3.0," months."
1: wrt 6,A,B
```

Notice that all text is in the format statement. If all text were placed in the write statement:

```
0: fmt f6.0,f3.0
1: wrt 6,"Sales =",A,"DM for the last ",B," months."
```

This output would result:

```
Sales =    10DM for the last    4
months.
```

Since there were more output items than format specs, a CR/LF was output after the last spec was referenced. Then "months" was output, followed by another CR/LF, at the end of the write statement.

Now suppose that we wish to output the same line, but have each write statement determine the units: "DM...months" on one line and "K\$...days" on another line. Here is one method and its results:

```
0: fmt "Sales =",f6.0," for the last ",f3.0,c
1: wrt 6,A,"DM",B," months."
2: wrt 6,A,"k$",B," days."
```

```
Sales =    10 for the last DM    4 months.
Sales =    10 for the last k$    4 days.
```

The problem of extra CR/LFs is avoided here by referencing data spec `c`. But notice that text preceding each spec is output **before** text preceding each variable. So the units DM and K\$ are in the wrong place! To output the units in the correct order, use more `c` specs:

```
0: fmt "Sales =",f6.0,c," for the last ",f3.0,c
1: wrt 6,A,"DM",B," months."
2: wrt 6,A,"k$",B," days."
```

```
Sales =    10DM for the last    4 months.
Sales =    10k$ for the last    4 days.
```

When using text in format and write statements, remember:

- Place all "fixed" text in the format statement.
- Place all "variable" text in the write statement and reference `c` data specs.
- Text (and all other edit specs) preceding each data spec are always output **before** text preceding the corresponding item in the write statement.

Data Input Specifications


Data specs can be used to determine which characters are input from a data input string, and in what form the data will appear. When a format statement is referenced by a read statement, the read operation is not terminated until a LF character is read (unless the edit spec `z` is used). A general data input spec syntax is:

```
[r]f [w]
```

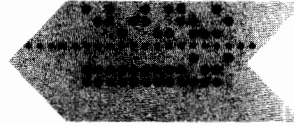
- `r` is the number of consecutive times the spec is to be used (if `r` is 1 it may be omitted). `r` and `w` must be integer constants.
- `w` is the width of the data field to be read. Omitting `w` specifies free-field read for the corresponding item(s).

A data spec like `f10` calls for reading ten numeric characters; all non-numeric characters which precede a numeric are counted but not entered. If an "E" is read within a numeric field, a number of the form `1e dd` is entered.

The following examples use an HP 9883A Tape Reader via a 98032A Interface set to select code 3. The paper tapes shown here are coded in ASCII.

To read this tape containing three data items  run this program.

123423453.45 (LF)



```
0: fmt 3f4
1: red 3,A,B,C
2: fxd 2
3: prt A,B,C
4: end
```


```
1234.00
2345.00
 3.45
```

Notice that four characters were read for each data item.

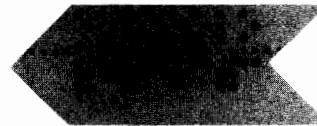
Now delete line 0, and run the program to read the same tape again.

```
123423453.45
 0.00
 0.00
```

Referencing free-field inputs the entire data string into the first variable; reading the LF terminates the read operation, skipping the last two variables.

To read this tape  run this program.

12 (CR) (LF) 34 (CR) (LF) 56 (CR) (LF) 78 (CR) (LF)



```
0: 1→A;fmt f3
1: red 3,rA;jmp
  (A+1→A)>4
2: fxd 0
3: prt r1,r2,r3,
  r4
4: end
```

```
12
34
56
78
```

Since there is a LF character after each data item, the read statement had to be repeated to input each item.

Input Edit Specifications

The following edit specs can be used to increase input format flexibility. An example use of each spec follows.

<code>z</code>	Cancel LF as terminator.
<code>[r]x</code>	Skip character.
<code>[r] /</code>	Skip data item.
<code>[r]cw</code>	String field width.

The `z` spec causes the calculator to read only the number of characters specified in the rest of the format statement. The read operation is automatically terminated after the characters are read, without the need for a LF character.

For example, this tape has 9 two-digit numbers, but no LF character \blacktriangleright

122334455667788990



Run this program to read and print all nine items.

```

0: 1→A
1: fmt z,f2
2: red 3,rA;jmp
   (A+1→A)>9
3: 1→A;fxd 0
4: prt rA;jmp
   (A+1→A)>9
5: end

```

12
23
34
45
56
67
78
89
90

The `x` spec causes the calculator to skip (not count) `r` number of characters. For example, to read and skip alternate items on the previous tape, run this program. Notice that the entire format statement is referenced for each data entry.

```

0: fmt z,f2,2x
1: red 3,A,B,C,
   D,E
2: fxd 0
3: prt A,B,C,D,
   E
4: end

```

12
34
56
78
90

1-22 Formatted I/O Operations

The `/` spec causes the calculator to skip all data which precedes `r` number of CR/LF characters.

For example, to read only the first and last items on this tape ▶
run the program shown below.

12 (CR) (LF) 34 (CR) (LF) 56 (CR) (LF) 78 (CR) (LF)



```
0: fmt f2,3/,f2
1: red 3,A,B
2: fxd 0
3: prt A,B
4: end
```

12
78

The `c` spec indicates the number of characters to input into a dimensioned string variable. All characters are entered until either the dimensioned string is filled, or `w` characters are read. When `c` is used, reading a LF does not terminate a string input.

Reading into `s` string variable is a convenient way of storing all characters in a data item, including non-numeric. Later, any character or portion of the string can be evaluated using String Variable operations

For example, the HP 98033A BCD Interface inputs data from a measurement device, such as an HP 3480 Digital Voltmeter, and transfers ASCII-coded data to the calculator in this 16-character format:

(sign) D₁ D₂ D₃ D₄ D₅ D₆ D₇ D₈ E (sign) D₉, (overload) D₁₀(LF)

Since a delimiter follows each data item, it's easy to read the numeric items by using a read statement with free-field format.

```
0: red 3,r1,r2
```

By reading into a string variable, however, you can evaluate any portion (substring) of the string later in the program. Running this sequence stores the data reading in `A`, the overload numeric character in `B`, and the function code in `C`.

```
1: dim A$(20)
2: fmt c16
3: red 3,A$
4: val(A$[1,12])
  →A
5: val(A$[14,
  14])→B
6: val(A$[15,
  15])→C
```

See the String Variables chapter of the Operating and Programming Reference for other string operations.

Conversion Statement

```
conv [code1 ; code2 [ ; code3 ; code4 ] ...]
```

The conversion statement sets up a character replacement table for use with read and write statements. Up to 10 pairs of decimal codes can be specified at a time. Each new conversion statement cancels the previous table and sets up a new one. A conversion statement with no parameters cancels any previous table.

In this program sequence ↯

line 8 sets up a conversion table which changes the spaces (decimal 32) output between numbers in line 10 to asterisks. Then line 11 cancels the previous table.

```
8: conv 32,42
9: fmt 3f10.2
10: wrt 6,A,B,C
11: conv
```

As another example, line 6 on the right specifies that whenever an ASCII "/" (decimal 47) is read in line 7, it will be converted to an ASCII "HT". Also, whenever an ASCII "NULL"

is read, it will be converted to an ASCII "CR" (decimal 13). Changing these input delimiters causes the calculator to skip all characters read between a / and a LF, and ignore all NULL characters. Input delimiters are described on page 1-6

```
6: conv 47,9,0,
13
7: red 3,A$
```

List Statement

A select code parameter can be used with the list statement when the General I/O ROM is present, enabling program listing on a peripheral output device. The new list syntax is:

```
List[#select code][ ; line nos.]
```

The optional line numbers remain as described in the operating and programming manual.

A listing of the program described on page 1-13 is shown next, output to a 9866B Printer via a 98032A Interface set to select code 6. Referring to the listing, notice that three CR/LFs are automatically output before and after the listing.

1-24 Formatted I/O Operations

```
0: fmt 1,5;/wrt 6.1
1: fmt 30x,"Trigonometric Table",2;/wrt 6
2: deg;sf 14;0→A→X
3: fmt 2,6x,"Degrees (Radians)",12x,"Sine",10x,"Cosine",12x,"Tangent",/
4: wrt 6.2
5: fmt 3,9x,f3.0,"(",f4.2,")",15x,f6.3,9x,f6.3,10x,e10.3
6: wrt 6.3;X;πX/180,sin(X),cos(X),tan(X)
7: A+1→A
8: if A=3;0→A;fmt /wrt 6
9: if 180>X;X+10→X;eto 6
10: wrt 6.1
11: end
*24841 ←checksum
```

If not needed, the CR/LFs and checksum can be suppressed by adding a decimal point and a non-zero digit to the select code. For example, use this statement to list the last half of the program shown above:

```
list#6.1,5
```

```
5: fmt 3,9x,f3.0,"(",f4.2,")",15x,f6.3,9x,f6.3,10x,e10.3
6: wrt 6.3;X;πX/180,sin(X),cos(X),tan(X)
7: A+1→A
8: if A=3;0→A;fmt /wrt 6
9: if 180>X;X+10→X;eto 6
10: wrt 6.1
11: end
```

The checksum and CR/LFs are not suppressed when this list statement is used with the internal printer (select code 16).

Chapter 2

Table of Contents



Overview of the HP-IB	2-3
Bus Messages	2-4
The HP-IB Card	2-6
HP-IB Addresses	2-6
Transfer Parameters	2-8
Non-active Controller Address	2-9
Device Statement (dev)	2-9
Multiple Listeners	2-10
Data Messages	2-11
Sending Data Messages	2-11
Receiving Data Messages	2-15
Trigger Message (trg)	2-16
Clear Message (clr)	2-17
Remote Message (rem)	2-18
Local Message (lcl)	2-19
Local Lockout Message (llo)	2-19
Clear Lockout/Set Local Message (lcl)	2-20
Service Requests and Polling	2-20
Sending the Require Service Message (rqs)	2-21
Receiving the Require Service Message	2-21
Serial Polling	2-22
Sending the Status Byte Message	2-22
Receiving the Status Byte Message	2-22
Parallel Polling (pol)	2-25
Sending the Status Bit Message	2-24
Receiving the Status Bit Message (pol)	2-25
Poll Configure Statement (polc)	2-26
Poll Unconfigure Statement (polu)	2-26
Pass Control Message (pct)	2-26
Abort Message (cli)	2-27
Sample Application	2-28
Command Statement (cmd)	2-31
Equate Statement (equ)	2-33
Extended Read Status (rds)	2-34
The HP Interface Bus (an overview)	2-37
HP-IB Lines and Operations	2-37
Interface Functions	2-46

Notes

Chapter 2

HP-IB Control

This chapter describes how to control and exchange data with instruments via the HP Interface Bus (HP-IB). Also included here is a brief description of the HP-IB and the 98034A Interface Card. For more information on the bus card, refer to its installation and service manual.

The HP-IB is Hewlett-Packard's implementation of IEEE standard 488-1975. A copy of this standard can be ordered from the IEEE Standard's Office; 345 East 47th Street; New York, N.Y. 10017. A brief technical description of the HP-IB is at the back of this chapter.

Overview of the HP-IB

The HP Interface Bus has a serial-byte bus structure which permits bi-directional communication between many instruments. When a controller such as a calculator is used, up to 14 additional HP-IB compatible devices can be controlled via one interface card.

Instruments can be controlled or programmed and data can be transmitted between devices on the bus. This is possible since each instrument connected to the bus has the potential of being a **talker** (send data) or a **listener** (receive data). Each instrument has a unique talk and/or listen address by which a **controller** communicates with the instrument. A unique handshake technique allows the communication to take place at a speed determined only by the specific instruments being addressed. Slower devices will not slow down the communication speed of the bus when they are not addressed.

In addition to the talker, listener, and controller functions, one device can be assigned the role of **system controller** on the bus. This instrument communicates with every other instrument and can halt and reset all bus operation at any time. The calculator is normally set to be the system controller; the function is enabled on the 98034A Interface Card.

Bus Messages

The communication capabilities of each device on the HP-IB can be exercised by using the messages described here. The General I/O ROM permits using three messages (Data, Remote, and Abort) for addressing one instrument at a time (the computer is assumed to be the system controller). The addition of an Extended I/O ROM, however, enables all 12 bus messages for complete bus capability. Messages can be transferred between device \leftrightarrow device, device \leftrightarrow controller and controller \leftrightarrow controller.

The 12 bus messages are categorized and listed below. A more complete description of each message is given later.

Device Communication:

- **Data** – The data characters transferred between devices by a calculator instruction (such as red, wrt, or cmd).

Device Control:

- **Trigger** – Causes a device or group of devices to simultaneously initiate a device-dependent action.
- **Clear** – Initializes device-dependent functions to a predefined state.
- **Remote** – Switches selected devices to remote operation, allowing parameters and device characteristics to be controlled by Data messages.
- **Local** – Causes selected devices to revert to manual control for future parameter modifications.
- **Local Lockout** – Prevents the device operator from switching the unit to manual control.
- **Clear Lockout and Set Local** – Removes all devices from local lockout mode and causes all devices to revert to local.

Interrupt and Device Status:

- **Require Service** – Asynchronously indicates a device's need for interaction with the controlling device.
- **Status Byte** – Presents device-dependent status information; one bit indicates whether or not the device currently requires service. The remaining 7 bits indicate status defined by the device.
- **Status Bit** – A single bit of device-dependent status which may be logically combined with status bit messages from eight devices.

System Control:

- **Pass Control** – Causes bus management responsibilities to pass from the sending device to the receiving device.
- **Abort** – Stops all communication and causes control to pass back to the system controller, independent of the device currently in control.

To determine which messages are needed to control and exchange data with each device, first review the programming requirements for the device as explained in its operating manual. Then find the appropriate bus message syntax in this chapter. Remember that most instruments must be set to Remote before they will respond to other bus messages, and that the Data message is used to transfer control characters and data between devices.

If the operating manual does not describe which bus messages are required by that device, you can determine which messages are required by knowing which HP-IB interface functions are implemented on the device. See the back of this chapter.

The following table summarizes the bus operations available with the General I/O and Extended I/O ROMs. Each message and operation is further described in the remainder of this chapter. The table on page 2-27 summarizes computer response to bus messages when it is not in active control of the bus.

Sample HP-IB Operations with the 9825 Computer

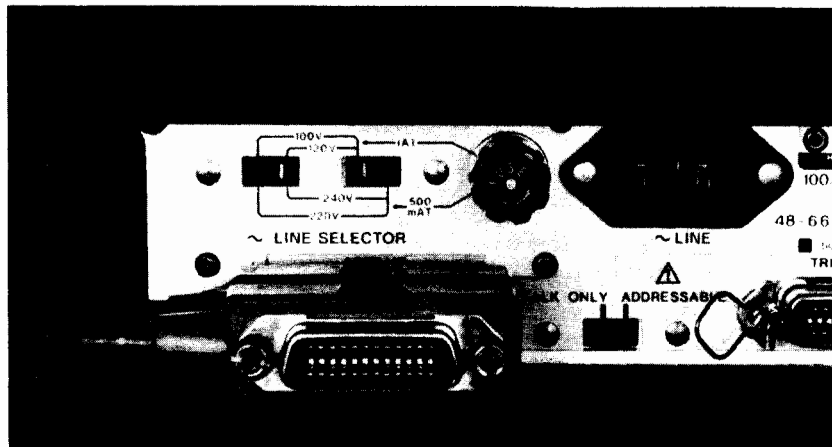
Message Name	Description	Sample Operations ¹
Data	Output text and variables to single devices.	wrt 701,"total=",A wrt "Printer",A,B,C
	Output single characters.	wtb 701,H
	Input data from a device.	red 711,A#,B#
	Input single characters.	rdb(711)→A
	Specify device address and send data in the form of ASCII characters.	cnd 7,"?U-", "L10" cnd "dvm", "L10"
	Output data to multiple listeners.	wrt "dvm,printer", "L10" cnd "?UK", "L10"
Trigger	Transfer data from device to device.	cnd "?K"
	Send a Group Execute Trigger to all instruments.	tra 7
Clear	Send a GET to selected devices.	tra 711
	Clear all devices.	clr 7
Remote	Clear selected device.	clr 711
	Enable remote mode on all devices. Switching the calculator on also sends a Remote message.	ren 7
Local	Set remote mode on selected device.	ren 711
	Return selected device to local mode.	lcl 722
Local Lockout	Prevent all devices from returning to local mode.	llo 7
Clear Lockout/Set Local	Set local mode and disable local lockout on all devices. (****) also sends this message.	lcl 7
Pass Control	Transfer bus control to a selected device.	pct 723
Require Service Status Byte	Request Service from the controller and send an 8-bit status byte for response to a Serial Poll.	ras 7,105
Status Bit	Bit and logic level for responses to a Parallel Poll.	ras 7,64
Abort	Clear all bus operations and return control to the original system controller. (****) also sends an abort message.	cli 7

¹In each case, a device name can be assigned and substituted for the select code parameter. See "The Device Statement".

The HP-IB Card

The HP 98034A Interface provides HP-IB capability for the HP 9825 Calculator. The bus card buffers all data and control messages between the calculator and instruments on the bus. The interface is preset to respond to select code 7.

Each instrument on the bus is connected through a 24-wire cable with a piggy-back connector on each end. Cables are available in 1, 2, and 4 meter lengths. Total cable length for a system can be up to 20 meters. The next photo shows two bus cables connected to an HP 3490A.



Connecting Bus Cables

HP-IB Addresses

The General I/O ROM provides simplified control of instruments via the HP-IB by using a select-code parameter containing a three- or four-digit integer. The first one or two digits specify the bus card select code, while the last two digits represent the address of the instrument on the bus.

Instruments having HP-IB capability are assigned unique 7-bit ASCII characters for talker and listener addresses.¹ A controlling instrument, such as the calculator, uses the address characters to indicate which instrument is to talk (send data) or listen (receive data). For example, here are the addresses usually assigned to some instruments:

Instrument	HP-IB Address	
	Talker	Listener
98034A Interface	U	5
3490A Multimeter	V	6
9871A (Opt. 001) Printer	A	!
59309A Digital Clock	P	0

¹Bus addresses for most devices can be changed, if needed; see the manual furnished with each instrument for details.

Now, using the ASCII table in the Appendix, convert the five least-significant bits of each character's binary form to a decimal value:

Instrument	Address	
	Character	5-bit Value
98034A Interface	U	21
	5	21
3490A Multimeter	V	22
	6	22
9871A Printer	!	01
	A	01
59309A Clock	P	16
	0	16



Notice that the 5-bit value for talker-listener instruments is the same number.

These numbers are used as the HP-IB address code in select-code parameters of General I/O operations. The HP-IB address code must always contain two digits; if the 5-bit value is a one-digit number (e.g., 9), a leading zero must be used (e.g., 09).

As examples, these write statements are addressed to a 9871A Printer via a bus card set to select code 7

```

98034A select code
    |
    v
printer address
0: wrt 701,A,B,
C
1: fmt 5,2c30
2: wrt 701.5,A$,
Q$
    |
    v
format number
    
```


Transfer Parameters

Transfer parameters specify each message's origin (sender) and destination (receiver) on the bus. For most messages, the calculator, as controller, specifies the device sending the message and the device or devices receiving the message.

The select code parameter is used to specify transfer parameters in the same form described in the previous section. Here is a review of the general syntax:

`cc[dd][.f]`

- `cc` = one or two digit select code of interface card.
- `dd` = HP-IB address from 00 thru 31¹ (must be two digits).
- `.f` = format number (read and write statements only).

The address code within each select code parameter specifies the appropriate address (origin or destination) of the device on the bus. Except for the command (`cmd`) statement, all I/O operations which have an address code automatically transmit the computer's preset talker/listener address and the specified address code in the appropriate order on the bus. When the computer is the active controller, this address sequence is preceded by the bus Unlisten command to clear all listeners previously set. For example, `wrt 711.3...` sends this address sequence before sending data:

- Unlisten command
- Calculator talk address
- Listen address 11 (on bus 7)

Extended Bus Addressing

When communicating among devices which use extended addressing on the HP-IB, the extended address can be specified by adding two more digits to the select code parameter. Here is the complete syntax:

`cc[dd[ee]][.f]`

- `ee` = extended address, from 00 thru 31 (must be two digits).

Extended addressing is provided by the bus definition (see IEEE Std. 488-1975). The primary address of a bus device is followed by another byte of addressing information. This byte has an allowed range of 00 thru 31, with the Secondary Command Group (SCG) bits (bits 5 and 6)

¹Address 31 is a special address. See next page.

set. This optional byte is automatically sent by the Extended I/O ROM when the two additional digits are specified in the select code parameter. For example, `wrt 70321.3,...` outputs this address sequence:

- Unlisten command
- Calculator's preset talk address
- Device listen address 03 (on bus 7)
- Secondary device address 21.

As controller on the bus, the calculator has the ability to send secondary addresses. As a device on the bus (not controller), the calculator does not respond to a secondary address.

Non-Active Controller Address

When a device is not the active controller on the bus it can not address other devices to talk or listen. If the calculator specifies a device address from 00 thru 30 in an I/O operation when it is not in active control, an error message will occur. The calculator can still send Data and other messages when addressed to talk or listen, however, by specifying device address 31. For example, this sequence sends a Data message (contents of variables A, B, and C) when the calculator is addressed to talk:

```
5: rds(7)→A; if bit(6,A)=0; gto +0
6: wrt 731,A,B,C
```

The calculator continually executes line 5 until it is addressed to talk. The read status function is described at the end of this chapter.

Device Statement

```
dev "name 1" ; select code 1 [device address]
    [ ; "name 2" ; select code 2 [device address]...]
```

The device statement sets up a name/address list for peripheral devices. Once each name is set up, it can be used in place of the select code parameter in each I/O operation. Each new device statement adds the new names to the previous list; each name can have only one select code parameter assigned at a time. The device list is erased when the calculator is reset (☐), run command, erase command, or switch power on). This statement can be used with any interface with a select code of 2 thru 15, but the optional device address is allowed only with the HP-IB interface.

In this example sequence, line 6 sets up the names "printer", "dvm", and "punch". Then lines 7 thru 11 use the device names instead of numeric select codes. Notice that the tape punch (select code 3) is assigned a name, even though it is not a bus device.

```

6: dev "printer",701,"dvm",711,"punch",3
7: wrt "dvm","R3F0T1E"
8: red "dvm",A
9: wrt "punch",A
10: if A=64;jmp -3
11: wrt "printer","value =",A

```

Each select code parameter can be a positive integer from 2 thru 15. If a format number other than 0 is to be specified, it can be specified in the read or write statement by using the syntax:

"device name . format no."

For example, line 13 in this sequence references format 1, while line 14 references format 0.

```

12: fmt 1,"R3F",b,"T1E"
13: wrt "dvm.1",F
14: red "dvm",A

```

Some of the example programs in this manual were output using an HP 9871A Printer. To output program listings via the bus, the list statement is used. For example, this statement was used to output the sequence shown above:

```
list#706.1, 12, 14
```

Multiple Listeners

More than one listener can be specified for Data messages and certain other messages by using device names, separated by commas, in place of the select code parameter in each statement. For write operations, the calculator is set to the talker and all names in the list are set to listeners. For read operations, the first name in the list is set to the talker and the calculator and all other names in the list are set to listeners. This method assumes that the calculator is to be either the talker or the listener in the operation. If a talker and one or more listeners is to be set up without the calculator participating in the transfer, the command statement must be used.

For example, this program sequence first defines device names, and then simultaneously outputs the variables A, B, and C to a voltmeter and an HP 9871A Printer. The next line outputs the string A\$ to the printer and the HP 59304A Display. The last line inputs a reading from the voltmeter, and also sends it to the printer and the display.

```

43: dev "printer",701,"dvm",722,"display",724
44: wrt "printer,dvm",A,B,C
45: wrt "printer,display",A$;wait 1000
46: red "dvm,printer,display",D

```

Multiple device names are not allowed within these bus statements: `trg`, `clr`, `cli`, `rem`, `llo`, `lcl`, `pct`, `polc`, and `polu`. Instead, either execute the statement repeatedly (using one device name at a time), or use the method shown on page 2-17 to send the message simultaneously.

Data Messages

The primary reason for the existence of the interface bus is to transmit Data messages. It is the Data message that exchanges device-dependent data among bus devices. Data messages are one or more 8-bit bytes (characters) sent over the bus from one Talker to one or more Listeners. For example, a Data message from the controller might send a function code to set a frequency counter to the frequency mode. The message might be a pair of ASCII characters such as "F2". Another Data message might be an ASCII string of alpha prefixes and numbers that are the measured value. A Data message thus may impart control information, or a measured value, or a command of some sort. Any Data message is multi-valued. That is, it will be transmitted over the 8 data lines of the interface. A Data message can have either an implicit termination or an explicit termination, for example, carriage return/line feed.

Sending Data Messages

The following I/O operations are used to send Data messages from the calculator to the bus:

I/O Statement	Described on Page(s)
wrt (write)	2-12 and 1-5
wtb (write binary)	3-3
list (list program)	1-23
tfr (transfer)	6-8
cmd (command)	2-31

The select code parameter previously explained is used to specify the listener address(es).

The write statement (`wrt`) can reference a format statement for controlling the output of the expressions. A Carriage Return/Line Feed (CR/LF) is sent at the end of the write statement unless `edit spec z` is specified in a format statement. No CR/LF is sent after the write binary statement (`wtb`).

Instruments designated as listeners on the bus are controlled by using write and write binary statements. The address-code parameter just described must be used in each I/O operation.

Here is a short program which prints a listing of r-variables using a 9871A (Opt. 001) Printer. The printer bus address is set to "!" (address code 01). A sample printout and analysis of the program follow.

```

0: 0+A+X;10+B;          4: wrt 6,"
   20+C.                r",A,rA,"
1: fmt 5/,30x,          r",B,rB,"
   "Data Register       r",C,rC
   Listing",/;wrt
   701
2: fmt 3"ΔΔΔΔΔΔ        5: A+1+A;B+1+B;
   Reg. ΔΔΔΔΔConen    C+1+C;X+1+X
   ts ΔΔΔΔ",/;wrt     6: if X=5;0+X;
   701                wrt 701
3: fmt f.0,e18.8       7: if A<=9;jmp -
   ,f.0,e18.8,f.0,    3
   e18.8              8: fmt 5;/wrt
                       701
                       9: end

```

Data Register Listing

Reg.	Contents	Reg.	Contents	Reg.	Contents
r0	0.00000000E 00	r10	3.44827586E-01	r20	5.00000000E-01
r1	0.00000000E 00	r11	5.00000000E-01	r21	4.76190476E-01
r2	0.00000000E 00	r12	8.33333333E-01	r22	4.54545455E-01
r3	3.33333333E 00	r13	7.69230769E-01	r23	4.34782609E-01
r4	2.50000000E 00	r14	7.14285714E-01	r24	4.16666667E-01
r5	2.00000000E 00	r15	6.66666667E-01	r25	4.00000000E-01
r6	1.66666667E 00	r16	6.25000000E-01	r26	3.84615385E-01
r7	1.42857143E 00	r17	5.88235294E-01	r27	3.70370370E-01
r8	1.25000000E 00	r18	5.55555556E-01	r28	3.57142857E-01
r9	1.11111111E 00	r19	5.26315789E-01	r29	3.44827586E-01

Here is a brief analysis of the program:

- Line 1 – Causes the printer to advance paper five lines and print the program title. Notice that the title is centered on the paper by sending a block of spaces before the title.
- Line 2 – Causes the column headings to be printed. Here, spacing is achieved by including spaces (ΔΔ) within the text format specs.

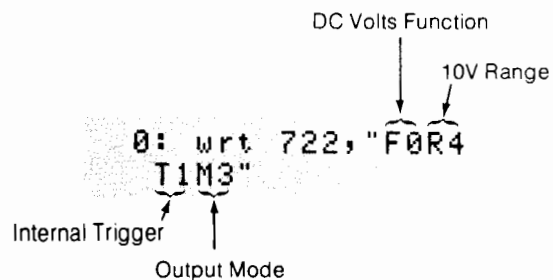
- Lines 3 and 4 – Specify the form of each table entry and which numbers (variables) are printed in each line of the table. The magnitude of each width (w) parameter in the format statement (line 3) was selected to position the corresponding table column under the appropriate heading.
- Line 5 – Increments program counters.
- Line 6 – Causes the printer to space one line after each five lines printed.
- Line 7 – Determines the maximum number of table entries.
- Line 8 – Causes the printer to space five lines.

As another example, the HP 3490A Multimeter can be connected via the bus and controlled by using strings of ASCII characters. Here is a list of control characters:

HP 3490A Control Characters

Char.	Function	Char.	Function
R	Range Program Identifier	T	Trigger Source Program Identifier
1	10,000 k Ω ; Test 7	0	Internal Sample Rate
2	1,000 k Ω ; 1000 V; Test 6	1	Immediate Internal
3	100 k Ω ; 100 V; Test 5	2	Next External Trigger
4	10 k Ω ; 10 V; Test 4	3	None
5	1 k Ω ; 1 V; Test 3	M	Mode of Operation Program Identifier
6	.1 k Ω ; 1 V; Test 2	0	Addressed Multi with No Output
7	Autorange; Test 1	1	Addressed Multi with Output
F	Function Program Identifier	2	Addressed Single with No Output
0	DC Volts	3	Addressed Single with Output
1	K Ohms	4	Interrupt Multi with No Output
2	AC Volts	5	Interrupt Multi with Output
3	Test	6	Interrupt Single with No Output
S	Sample/Hold Program Identifier	7	Interrupt Single with Output
0	Sample/Hold Off	E	Execute Mode of Operation Program
1	Sample/Hold Off		
2	Track/Hold		
3	Acquire/Hold		

The line on the right shows one method of setting 3490A range, function, etc., and then initiating a data sample. A read statement could be used to input the resulting data.



Here is a line which sets an HP 3330A Synthesizer to an output frequency of 100.0 Hz. Its listen address is "\$" (decimal code 04).

```
1: wrt 704, "L100"
    ↑      ↑
    0="    Value
    Hertz
```

Sometimes it may be necessary to program a device using a variable stored in the calculator. Perhaps the variable contains the desired output frequency or amplitude for a signal source. This example outputs the same characters as line 1 above. The data spec `f.1` suppresses leading spaces, which the 3330A cannot accept.

```
2: fmt "L", f.1,
   "="
3: wrt 704, AA = 100.0
```

Receiving Data Messages

When the calculator is a listener, the following I/O operations are used to receive data messages from the bus:

I/O Operation	Described on Page(s)
red (read)	below and 1-7
rdb (read binary)	3-4
rds (read status)	2-22 and 3-5
tfr (transfer)	6-8
pol (parallel poll)	2-25

The read statement (`red`) can reference a format statement to control the incoming data. Read binary (`rdb`) and read status (`rds`) are both functions, which means that they must appear as part of a statement (such as `rdb(7)→A` or `dsp rdb(7)`) in order to be stored as part of a program. Read binary (`rdb`) reads only one byte (8 bits on the HP-IB) at a time.

For most applications, read statements and the free-field format are used to input data via the HP-IB. Many devices output leading non-numeric characters in data messages; format statements must be referenced to read those characters. Chapter 1 describes the data and edit specs usable with format statements.

Examples best illustrate the technique for receiving data via the bus. For example, the HP 3490A Multimeter may send a data message of this form when programmed for DC volts and the 10V range (Δ indicates a space):

N Δ D C + 0 8 3 4 6 2 E - 4 $\text{\textcircled{CR}}$ $\text{\textcircled{LF}}$



By using the sequence on the right, the calculator skips the four leading non-numeric characters and reads the numeric portion of the string.

```
5: fmt 4x,f
6: red 722.8,A
```

By using string variables, either of these sequences will input the same number shown above. The first sequence has the advantage of retaining all characters in the input string; in either case the leading characters can be checked later in the program.

```
7: dim A$[20]
8: fmt c16
9: red 722,A$
10: val(A$[5]) $\rightarrow$ A
```

The **b** data spec can be used to input non-numeric characters. For example, suppose that this sequence is used to read the 3490A data string shown above. Afterwards, variables A through E will contain \blacklozenge .

```
7: dim A$[4]
8: fmt c4,f
9: red A$,A
```

```
0: fmt 4b,f
1: red 722,A,B,
  C,D,E
```

```
A = 78 (N)
B = 32 ( $\Delta$ )
C = 68 (D)
D = 67 (C)
E = 8.3462
```

Now consider this more-complex data message transmitted from an HP 3570A Network Analyzer:

- 0 3 4 . 4 8 , - 0 8 8 . 9 3 $\text{\textcircled{CR}}$ $\text{\textcircled{LF}}$

This sequence can be used to input and print the readings. The 3570A talker address is "A" (indicated by decimal code 01).

```
6: fmt ifxd 2
7: red 701,A,P
8: prt "Amplitud
  e=",A,"Phase=",
  P
```

Here is a printout \blacklozenge

```
Amplitude=-34.48
Phase=      -88.93
```


As a final example, here is a program which inputs, stores, and computes the average of 50 data samples taken using a 3490A Multimeter.

```

0: dim A$(50,
  20);0→A;1→I→N
1: wrt 722,"FOR6
  TIM3"
2: red 722,A$(I)
3: wrt 722
4: if (I+1→I)<51
  ;jmp -2
5: A+val(A$(N,
  5))→A
6: if (N+1→N)<51
  ;jmp -1
7: prt A/50
8: end

```

- Line 0 – Sets up a 50-element string array and initializes three variables.
- Line 1 – Programs 3490A remote controls and instructs it to take a data sample.
- Line 2 – Inputs the data sample into a string array element.
- Line 3 – Instructs the 3490A to take another data sample.
- Line 4 – Exits the data input loop when 50 samples are stored.
- Lines 5 - 7 – Compute and print an average value from the data.

Trigger Message

`tr@ select code [device address]`

The Trigger message is always sent from a controller to a selected device or set of devices. The purpose of the Trigger message is to initiate some action, for example, to trigger a digital voltmeter to perform its measurement cycle, or a digital voltage source to go to a new setting. Neither the Trigger message nor the interface indicates what a device does when it receives this message. The action taken is entirely up to the device designer.

Specifying only the interface card's select code (e.g., `tr@ 7`) outputs a Trigger message to all devices currently addressed to listen on the bus. Including a device address (e.g., `tr@ 703`) triggers that device only. Multiple device names cannot be used to specify multiple listeners with the trigger statement.

Here's a sequence that presets functions on a frequency counter and a voltmeter, and then outputs a Trigger message (line 51) to simultaneously initiate action on both devices. Line 52 then inputs the data for the DVM. The device names have been defined earlier in the program.

```

49: wrt "count","I2E8E?G?52"
50: wrt "dvm","TOF1M3E"
51: wtb "count,dvm";trg 7
52: red "dvm",A,B

```

Notice that line 51 first outputs the device addresses to specify them as listeners, and then sends the Trigger message to both devices.

Some devices do not respond to the Trigger message but still have "trigger" capability. In most cases they can be triggered by receiving an appropriate ASCII character via a Data message.

For example, this sequence inputs 50 data readings from an HP 3490A Multimeter. Line 2 presets the meter functions and executes a reading by sending the character E. Line 4 re-executes the preset functions for another reading.

```

0: dim A$[50,20];0→A;1→I→N
1: dev "dmm",722
2: wrt "dmm","FOR6T1M3E"
3: red "dmm",A$[I]
4: wrt "dmm","E"
5: if (I+1→I)<51;jmp -2

```

Clear Message

`clr select code [device address]`

The purpose of the Clear message is to provide a way to initialize devices to some predefined state. A Clear message can be sent either to all devices or to a selected set of devices. Only the controller can send a Clear message. The message is single-valued in the sense that it is either true or not true.

Specifying only the interface select code (such as `clr 7`) outputs a Device Clear (DCL) command to all devices addressed to listen on the bus. Specifying an individual device address (such as `clr 711`), however, outputs a Selected Device Clear (SDC) command to reset only the specified device.

For example, these lines send (simultaneously) the Clear message to the devices named "gen" and "clock":

```
0: dev "gen",706,"clock",720
1: wtb "gen,clock"
2: clr 7
```

But this statement sends the Clear message only to the clock:


```
11: clr "clock"
```

Remote Message

`rem select code [device address]`

The Remote message causes devices on the bus to switch from local, front-panel control to remote program control. It is single-valued, true or false. A device in remote control may be set to remain unresponsive to some or all of its panel controls.

To put a device into remote state the device must be addressed to listen while the REN (Remote Enable) line is true.

The Remote message is automatically output whenever the calculator is switched on or  is pressed. To prevent a device from being switched back to local by a front panel switch, use the Local Lockout statement (llo).

In the following example, the remote message is sent to the digital multimeter at select code 7, device address 22.

```
0: dev "dmm",722
1: rem "dmm"
```

All devices on the bus can respond to remote enable (REN) and are addressed to listen, are set to remote by this line:

```
2: rem 7
```

Local Message

`lcl` select code with device address

The Local message always originates with a controller and is sent to selected devices with the purpose of returning them 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 good systems practice to return all devices to local control upon the conclusion of automatic operations.

When an interface select code with a device address is specified, a Local message (Go To Local – GTL) is output to the specified device only. The `lcl` statement also sends the Clear Lockout/Set Local message (as explained later) when only the interface select code is specified.

The following lines send the Local message to the digital multimeter at select code 7, device address 11.

```
0: dev "dmm",711
1: lcl "dmm"
```

Local Lockout Message

`llo` select code

This message prevents an operator from returning a device to local control from its front panel. Since it always originates with the controller and is issued to all devices, transfer parameters are implied and need not be stated explicitly. As long as the Local Lockout message is in effect, no device can be returned to local control except through the controller itself, thus maintaining system integrity. In effect, this message locks out the "local" push-button present on most device front panels. This message prevents a casual passer-by from interfering with systems operations by pressing buttons.

To cancel local lockout, send a Clear Lockout/Set Local message (`lcl`). The Abort message (`cli`) does not cancel local lockout.

It is a good practice, especially when devices that are connected to the bus are used for other purposes, to send the Local Lockout message when they are used by the bus. The following line sets local lockout:

```
0: llo 7
```

Clear Lockout/Set Local Message

```
lcl select code
```

This compound message returns all devices to local, front panel control and simultaneously clears the Local Lockout message. It is used instead of the Local message when the controller, in an earlier action, issued the Local Lockout message.

Executing the local statement without a device address sends the Clear Lockout/Set Local message (REN), which sets all devices to local operation and cancels local lockout if it is in effect.

As an example, if the Local Lockout message is sent in line 0, and all bus activity is complete by line 30, the Clear Lockout/Set Local message is sent to return front panel control to all bus devices:

```
0: llo 7
  •
  •
  •
31: lcl 7
```

Service Requests and Polling

Service Requests and polling provide an additional means of communications between the calculator (controller) and other devices on the bus. A device may use the Require Service message (rqs) to ask for the attention of the controller. The controller could then use polling to find out the status or condition of a device on the bus. Typically, the controller uses polling to locate the source of a service request, and then the cause. Polling, however, is not limited to situations involving service requests.

Two polling methods are available: serial polling and parallel polling. A device responds to a serial poll by sending a Status Byte message (a value between 0 and 255) containing up to 8 bits of status information. A device responds to a parallel poll by sending a Status Bit message, which places one pre-selected bit of data on the bus. Use of the parallel poll allows the controller to quickly check up to eight devices at once, while use of the serial poll enables receiving a full byte (8 bits) of information from one device at a time. Each method is further described in the following pages.

Every bus-compatible device that is designed to use the service request should also respond to a serial and/or parallel poll. However, a device can be designed to respond to polling even though it does not use service request.

The operating manual for each device describes whether it can transmit Require Service messages and how the device responds to a poll.

Sending the Require Service Message

`rqs` select code, status byte

The Require Service message originates with devices other than the controller. The Purpose of the message is to let a device alert the controller to the device's need for some action by the controller. The Require Service message provides a system with an additional level of communications outside and asynchronous to the run-of-the-mill interchanges.

When the computer is not the active controller (either it passed control to another device or the System Controller switch on the interface is OFF), it can transmit the Require Service message and respond to both types of polls.

The status byte specifies an 8-bit byte (number between 0 and 255) to be sent in response to a serial poll, as explained in the next section. Bit 6 (decimal 64) is the SRQ bit which must be set if the Require Service message is sent. To clear the SRQ line, send a zero for the status byte. The SRQ line is also cleared if the controller serial polls the calculator. To send the Require Service message, the following program segment could be used:

```
0: pct 705
  .
  .
20: rqs 7,64
```

Receiving the Require Service Message

When the calculator is the system controller, it can be programmed to identify the source of a request and to service the requesting device(s); or the calculator can completely ignore all service requests. In most cases, however, a Require Service message indicates that the calculator should take some action to maintaining proper system operations. The calculator can use serial or parallel polling to identify the source of a service request and reveal the cause. The calculator can then service the device.

The Require Service message controls the bus management line SRQ. The calculator can check the status of this line to see whether a service request is present. All devices on the bus use the same line to request service. So when the calculator detects a service request, one or more devices may be the source.

Serial Polling

The serial poll is so named because the controller polls devices one at a time, in sequence, rather than all at once. When serial -polled, a device transmits a single byte of information to indicate its status. This transmission is called the Status Byte message. For example, a status byte may indicate that a device is overloaded (power supply), a device output has stabilized at a new level (signal generator), or a device has requested manual service (any of several types of devices). Once the controller has serviced each device that has been requesting service, the SRQ line is cleared (assuming no new requests are received).

Sending the Status Byte Message

The Status Byte message is sent at the request of the controller and is usually a response to the controller's poll taken to discover which device or devices are sending the Require Service message. The byte is specified via the require service (rqs) statement previously described. Bit 6 (decimal 64) must be set if the calculator requires service. The remaining bits may optionally be set to transmit other status information. In the usual case the message is directed to the controller for its interpretation and possible action. However, there is no restriction. Any device with the talker function can send the Status Byte message to any other devices with the listener function.

The status byte is stored in the 98034A Interface Card until the calculator is polled via the bus. The interface card automatically responds to a serial poll by sending the byte as a Status Byte message.

Receiving the Status Byte Message

To serial poll for checking the presence of a service request use the read status (rds) function to check interface card status. Bit 7 (decimal 128) indicates when the SRQ line is true.

For example, this program line checks for a service request:

```
7: if bit(7,rds(7));gto 15
```

If a service request is present, the program branches to line 15 to conduct a polling operation. The bit function is described in Chapter 3. Also, interrupt control can be used to automatically detect a service request and branch to a service routine which polls the bus. See Chapter 5 for details. The Extended I/O ROM increases the capability of the read status function, permitting you to conduct a serial poll by specifying the device address in the select code parameter. For example, executing this statement: `rds(711)+A` conducts a serial poll on a device with decimal address 11 and returns its status byte to A.

As another example, assume that a bus system has two devices that can send Require Service messages. One device has talk address X (decimal 24) and the other has talk address Y (decimal 25). When polled, each device returns status byte 64 (decimal) to indicate that it requires service (only bit 6 is set true). Otherwise, 0 is returned.¹

Here's a sequence that checks for a service request (lines 5-6), and then conducts a serial poll when a request is seen (lines 7 and 8). Then the program automatically branches to a display statement to indicate the device requesting service.

```

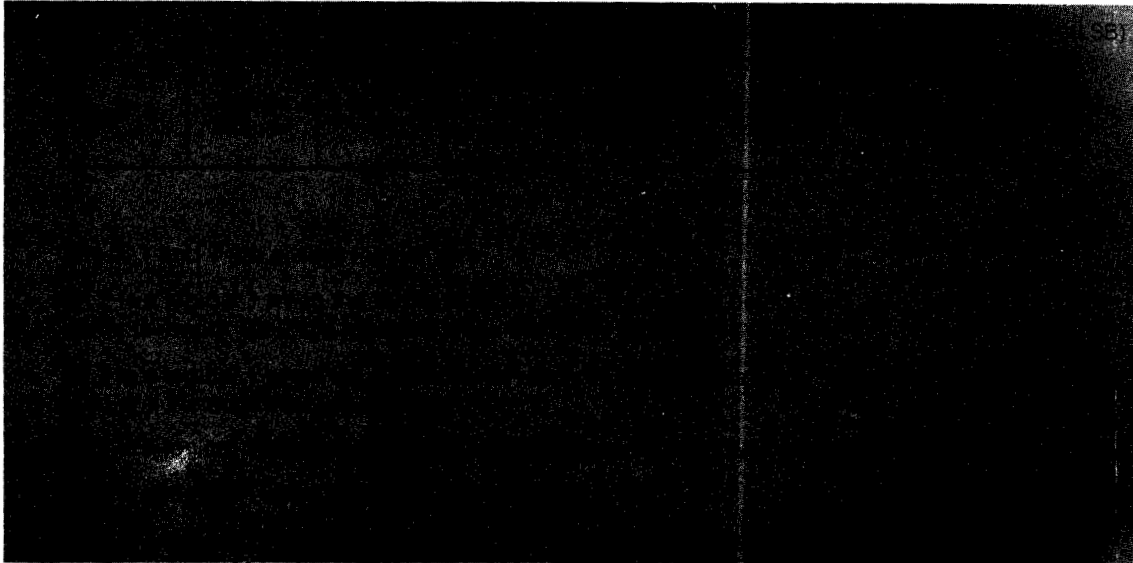
5: rds(7)+A
6: if bit(7,A)=0;gto 12
7: if bit(6,rds(724));gsb 10
8: if bit(6,rds(725));gsb 11
9: gto 5
10: dsp "SERVICE DEVICE X";stp ;ret
11: dsp "SERVICE DEVICE Y";stp ;ret

```

This sequence assumes that each device requires manual servicing (e.g., change printer paper) and that device X gets preference when both request service at the same time.

¹For convenience, the calculator assumes that bus status bits are numbered 0 thru 7 (0 is least-significant bit). Other devices may assume that the bits are numbered 1 thru 8 (1 is least-significant bit); see each manual for details.

Some devices return more than the service request bit (bit 6) in the Status Byte message. For example, here is the status byte sent by an HP 9871A (option 001) Printer:



9871A Status Byte¹

The sequence shown here conducts a serial poll (printer address is 01). If the printer has sent a Require Service message (bit 6 is 1), lines 4 thru 6 check other status bits and report conditions.

```

2: rds(701)+A
3: if bit(6,A)=0;gto 7
4: if bit(2,A);dsp "PRINTER BUFFER FULL";stp
5: if bit("xxx0lxxx",A);dsp "PRINTER NOT READY";wait 100;gto 2
6: if bit(5,A);dsp "PRINTER COVER OFF";stp

```

Parallel Polling

Sending the Status Bit Message

When the interface responds to a parallel poll, it sends the status bit message. The `rqs` statement sets or clears the status bit on the interface. When bit 6 of the status byte is set, as in `rqs 7: 64`, the status bit is set. To clear the status bit, clear bit 6, as in `rqs 7: 0`. The status byte is erased when the interface responds to a parallel poll.

Two switches on the interface card are associated with the Status Bit message. A rotary switch with ten positions corresponding to one of 8 bit positions (switch positions 9 and 10 are null positions) determines the location of the bit in the byte. A slide switch can be used to reverse the logic sense of the status bit (positive or negative true logic).

¹For convenience, the calculator assumes that bus status bits are numbered 0 thru 7 (0 is least-significant bit). Other devices may assume that the bits are numbered 1 thru 8 (1 is least-significant bit); see each manual for details.

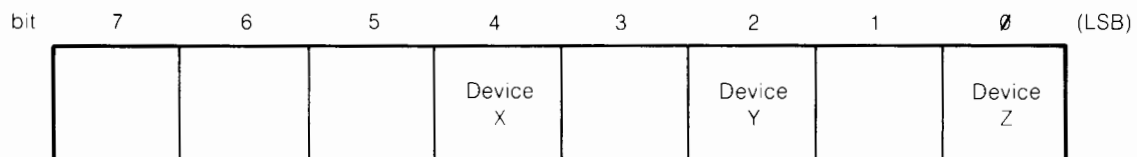
Receiving the Status Bit Message

```
pol (select code)
```

Parallel polling enables the calculator to check the status of up to eight devices at a time. This is possible since each device with parallel poll capability is pre-programmed to output one status bit when parallel polled. The bit is output as the Status Bit message. The status bit for each device indicates either that it has sent a Require Service message or that a predefined condition exists (e.g., a printer is out of paper). The poll function executes a parallel poll:

The poll function causes all devices to output their Status Bit messages simultaneously. The function then returns the combined byte for the calculator to analyze. The calculator can now quickly see which device(s) require service and take appropriate action.

As an example, suppose that three devices named X, Y, and Z can respond to a parallel poll. Each device is assigned to output a different Status Bit message when polled as shown here:



A logical 1 for devices X and Y indicate that they have sent a Require Service message. A logical 1 for device Z, however, indicates that it is out of paper. So for this system, the calculator can respond to the Device Z status bit by displaying `PRINTER OUT OF PAPER`, but it must perform a serial poll to determine the exact status of devices X and Y.

Here is a sequence which checks for a Require Service message on the bus (lines 0 and 1), and parallel polls all devices when a message is seen. The program then determines which device requires service (lines 3-5) and branches to the appropriate service routine.

```
0: rds(7)→A
1: if bit(7,A)=0;gto 7
2: pol(7)→B
3: if bit(4,B);gsb "srvcX"
4: if bit(2,B);gsb "srvcY"
5: if bit(0,B);gsb "srvcZ"
6: gto 0
```

In this sequence, notice that device X gets the highest priority service, while device Z (the printer) gets serviced last. The entire sequence is repeated until all service requests are cleared.

Poll Configure Statement

```
polc select code with device address ; status byte
```

Some devices having parallel poll capability can be programmed remotely to output a given status bit. The poll configure (polc) and poll unconfigure (polu) statements permit the calculator to set and clear status bits on these devices.

The poll configure statement sends the Parallel Poll Configure (PPC) command to set the device specified to send the specified status bit when parallel polled. For example, the statement: `polc 724,16` programs the device at address 24 to send status bit 5 (decimal 16) in response to a parallel poll.

NOTE

Bit 3 determines sense, and bits 0, 1, 2 determine the response line.

Poll Unconfigure Statement

```
polu select code [device address]
```

The poll unconfigure statement clears all programmed status bits on compatible devices. If the select code parameter contains a device address (e.g., `polu 725`) a Parallel Poll Disable (PPD) command clears only the specified device. If the select code does not contain a device address, however, a Parallel Poll Unconfigure (PPU) command clears all compatible devices on the bus.

Pass Control Message

```
pct select code with device address
```

The pass control statement sends a Pass Control message to transfer bus controller responsibility to another device which can assume active control of the bus.

After passing control the calculator can be addressed from the new active controller, and can send a Require Service message and respond to both serial and parallel polls, as described earlier. If the other controller cannot pass control back to the calculator, the Abort message (cli) must be used to halt all bus operation and return control to the calculator. An error occurs if a device address is not specified.

NOTE

Do not execute the pass control statement while a transfer (tfr) operation is being executed on the HP-IB. To check if a transfer operation is active, execute rds ("buffer name"). If -1 is returned, then the transfer is active.


**Calculator Response when Not Active Controller**

Message	Response
Data	Can branch to I/O routines when addressed to Talk or Listen. ¹
Trigger	No response.
Clear	Can branch to "clear" routine, by monitoring interface card status. ¹
Remote	No Response.
Local	
Local Lockout	
Clear Lockout/Set Local	
Require Service	
Status Byte	Status Byte sent in response to serial poll.
Status Bit	Status Bit sent in response to parallel poll.
Pass Control	Can branch to "controller" routines when addressed to control. ¹
Abort	The 98034A interface halts all bus operations, clears status bits and regains active control if preset to be System Controller.

¹See "Extended Read Status" and Chapter 5.

Abort Message

```
cli select code
```

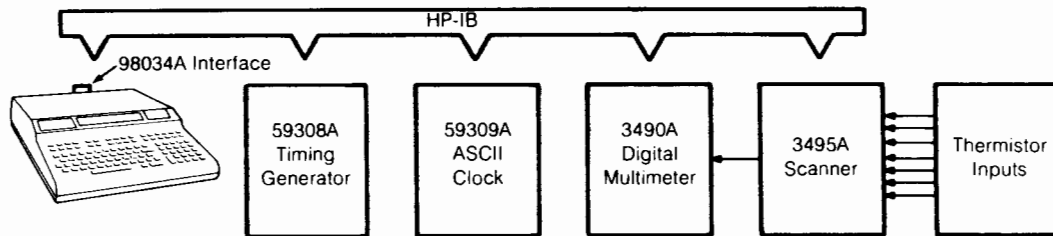
The clear interface statement sends the Abort message to halt all bus operations and return bus control to the calculator. Pressing  also outputs the Abort message.

The Abort message can be sent only when the calculator is preset as the System Controller; if not, error E9 occurs. The System Controller Switch is on the 98034A Interface.

Sample Application

Systems designed around the 9825 can be used in many areas for a wide variety of applications. The block diagram shown below outlines a typical HP-IB configuration. One possible use for this system would be to study temperature variations in a flowing stream or some other body of water near a power plant or factory in order to ascertain certain pollution effects.

For this simple application, temperatures are measured by thermistors, which output voltages read via the digital multimeter. Various channels of the scanner correspond to individual thermistor or temperature inputs. At pre-entered intervals, the calculator commands the scanner to rotate channels and reads the corresponding voltage equated to temperature from the multimeter. The calculator then prints the date, time and temperature readings on its internal printer. A listing and analysis of the program are on the following pages.



A Typical HP-IB System

```

0: dim T[5]
1: dev "timer",706,"scan",709,"clock",720,"dmm",722
2: ent "ENTER SAMPLE INTERVAL IN SECONDS",N;jmp N>10 and N<1000
3: fmt 1,"T",fz3.0,"E6ASR",z
4: fmt 2,2/,2fz2.0,/,fz2.0,":",fz2.0,":",fz2.0
5: fmt 3,/, "Channel",2x,"Temp(C)",/,16"-
6: fmt 4,3x,f1.0,5x,f7.4
7: fmt 5,"C",fz2.0,"E",z
8: fmt 6,"R7F1S1TIM3E",z
9: wrt "timer.1",N;wtb "clock","C"
10:
11: red "clock",T;l+1
12: 100frc(T/100)+T[I];int(T/100)+T;jmp (I+1+1)=6
13: wrt 16.2,"Date: ",T[5],"/",T[4],"Time: ",T[3],T[2],T[1]
14: wrt 16.3
15: l+C
16:
17: wrt "scan.5",C
18: wrt "dmm.6";red "dmm",D
19: 3807/(log(D)+9.39)-273+D
20: wrt 16.4,C,D;if (C+l+C)<8;gto 16
21: if bit(6,rds("timer"))=1;trg "timer,clock";gto 10
22: jmp -1
23: end

```

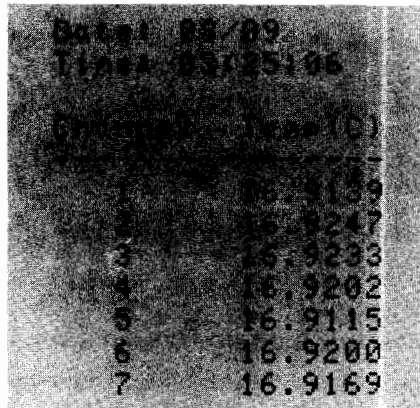
- Line 0 – Dimensions a simple array for date/time reading.
- Line 1 – Defines device names for the I/O devices.
- Line 2 – Allows the user to enter the sample interval time (range: $10 < N < 1000$).
- Lines 3 thru 8 – Are formats for the timing generator, printer, scanner, multimeter and clock.
- Line 9 – Programs the timer and triggers a time interval. The timer will output a Require Service message when the interval has elapsed. The write binary statement triggers the clock to store the current time.
- Lines 10 and 16 – Are null lines added to separate the "read" routine in the program listing.

- Lines 11 and 12 – Take the current date/time reading and separate the data. The HP 59309A Clock outputs the reading in this format:

MMDDhhmmss (CR) (LF)

MM = Month
 DD = Day
 hh = Hour
 mm = Minute
 ss = Second

- Line 13 – Prints the current date and time.
- Line 17 – Sets the scanner to the channel indicated by variable C.
- Lines 18 and 19 – Program the multimeter, input a data reading, and convert the voltage reading to degrees Celsius.
- Line 20 – Prints the channel number and temperature. The remainder of the line continues the reading routine until all 7 channels are read. A sample printout is shown below.



- Lines 21 and 22 – Monitor the bus, waiting for a Require Service message from the timer which indicates that the current time interval has elapsed. When this occurs, the trigger statement simultaneously restarts the timer and causes the clock to store the current time. Then the program branches to the read routine.

Although this is a simple example of bus operation, it shows the power available for controlling bus systems. In this example, the calculator would spend considerable time at lines 21 and 22 waiting for a long timing interval. Instead, line 21 could branch the program to a data reduction routine, or other time-consuming operation, while waiting to take the next set of data samples.

Command Statement

The command statement allows direct addressing of one or more devices on the bus by using ASCII characters to specify talker or listener addresses and data.

```
cmd select code , "address characters" [ , "data characters" ]
      or
cmd "device name(s)" or select code [ , "data characters" ]
```

As shown above, when a select code is specified, the address parameter is used to specify the talker and listener addresses for the Data message. But, as shown in the second syntax, device names or a select code can be used in place of the select code and address-characters parameter. (The device statement is described at the start of this chapter.) In either case, the data characters are output to the addressed listeners. Also, an equate name can be used in place of the data characters. The equate statement is described later.

The command statement is provided for compatibility with HP 9820A, 9821A, and 9830A calculator programs. Use of this statement is completely described in the HP-IB User's Guides: for 9820A/9821A Calculators, specify part number 59300-90001; for 9830A/B Calculator, specify part number 59300-90002.

When comparing this command statement with the ones available for the 9820A, 9821A, and 9830A/B Calculators, notice that a select code parameter is now required, since the 98034A Interface Card has a variable select code. Also notice that only one set of address-data parameters can now be included in each statement.

For example, this sequence could be used to send the data message "L100.0=" to program an HP 3330A Synthesizer responding to address \$ (decimal 04):

```
cmd 7, "?U$", "L100.0="
```

Clear all listeners ————— ↑ ↑ ↑ ————— Listen address (3330A)
Talker address (computer)

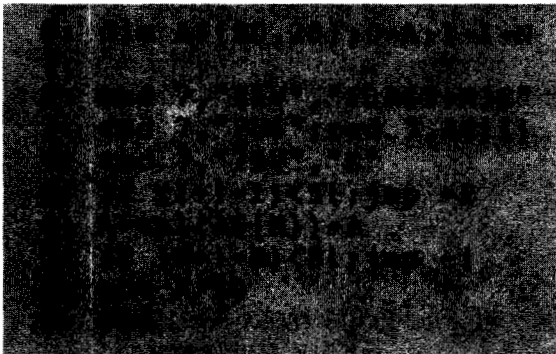
If the device name were defined as "GEN" in the device (dev) statement, this sequence could be used to send the same data message:

```
cmd "GEN", "L100.0="
```

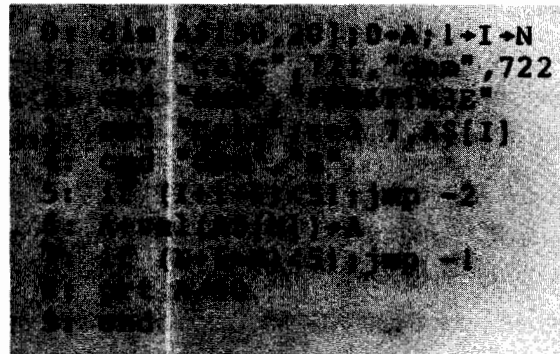

The ? character should be included at the start of each address parameter to clear (unlisten) all listeners on the bus. This instruction is automatically sent, however, when other I/O operations (wrt, red, tfr, etc.) are used on the bus, or when a device name is used in the command statement, as shown above.

Here are two versions of a program that inputs, stores and computes the average of 50 data samples from an HP 3490A Multimeter. The program shown on the left uses ASCII characters to specify talker/listener addresses, while the one on the right uses device names. A brief analysis of the program follows.

Using ASCII Characters



Using Device Names



- Line 0 – Sets up a 50-element string array and initializes three variables.
- Line 1 – Sets up a device name table for the program on the right.
- Line 2 – Programs the 3490A to the 1 volt range and instructs it to take a data sample.
- Line 3 – Inputs the data sample into a string array element.
- Line 4 – Instructs the 3490A to take another data sample.
- Line 5 – Exits the data input loop when 50 samples are stored.
- Lines 6 thru 8 – Compute and print an average value from the data.

Another version of this program, which does not use command statements, is on page 2-16. For another program using the command statements, see the following pages.



Equate Statement

```
equ "name 1" : data string 1 [ : "name 2" : data string 2...]
```

The equate statement sets up a list of names and data character strings for use with the command statement. Each name can then be used in place of the data parameter in command statements to output the associated string of ASCII characters. The data string can be either a sequence of text or a string variable name. Each successive equate statement adds the new names to the equate list; the same name, however, cannot be used for more than one string at a time. The equate list is cleared when the calculator is reset (**RESET**), run, erase, or power on).

For example, the following program is identical to the one on page 2-29 except that command statements are used in most cases to control devices on the bus. The equate statement in line 4 allows command statements to reference equate names "time" and "100 kohm", rather than specify the exact output strings. Using a string variable name as the data string for "time" allows the string to be entered by the user (lines 2 and 3) before it is "equated" in line 4. Notice that once the string is equated, however, it cannot be altered or deleted until the computer is reset. The rest of the program is as described earlier.

```

0: dim T[5],A[9]: "T      E6ASR" *AS
1: dev "timer",706,"scan",709,"clock",720",720,"amm",722
2: ent "ENTER SAMPLE INTERVAL IN SECONDS",N;jmp N>10 and N<1000
3: str(N)-AS[2,4]
4: equ "time",AS,"100kohm","R7S1T1M3E"
5: fmt 1,2/,2fz2.0,/,fz2.0,":" fz2.0," fz2.0
6: fmt 2,/, "Channel",2x,"Temp(C)",/,16"-
7: fmt 3,f4.0,f12.4
8: fmt 4,"C",fz2.0,"E"
9: cmd"timer","time";cmd"clock","C"
10:
11: red "clock",T[1]-T
12: 100*(N/100)-T[1];int N/100)-T;jmp (T[1]-1)=6
13: int 16,2,"Data:",T[1],",",T[2],T[3],T[4],T[5]
14: int 16,2
15:
16:
17: int "scan",C
18: int "amm",100kohm";red"amm",D
19: 100/(log(D)+4.34)-273+6
20: int 16,3,C,D,f12.4,C,f12.4
21: int 16,6,rd("timer");int "timer,clock";goto 10
22: jmp -1
23: end

```

Extended Read Status

`rds (select code [: variable 1[: variable 2[: variable 3]]) => variable 4`

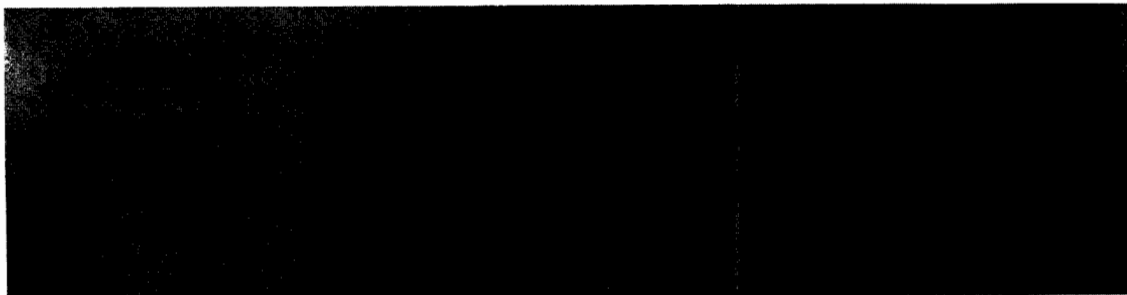
or

`rds (select code with device address)`

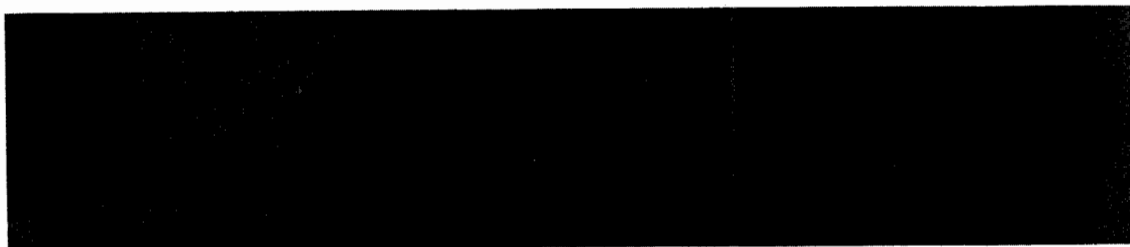
The Extended I/O ROM enables the read status function either to return up to four interface status bytes or conduct a serial poll.

When only the interface select code is specified (2 thru 15), up to four variables can be specified to return status bytes from the 98034A Interface. The interface status bytes are shown on the next pages. The first status byte is returned to variable₁, the second status byte is returned to variable₂, and so on.

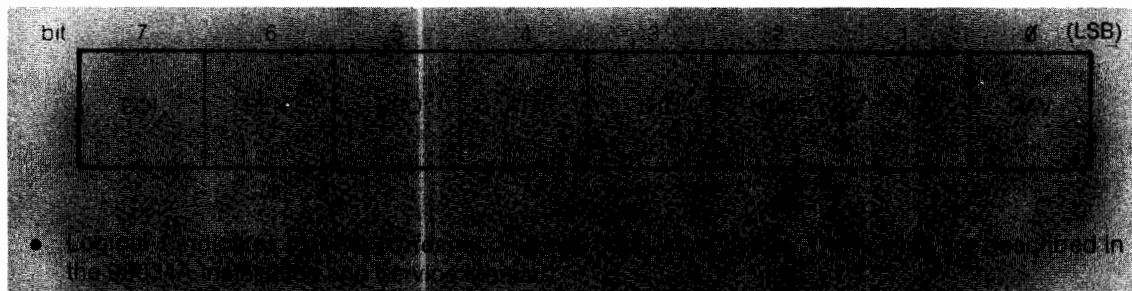
When a device address is specified in the select code parameter (as in the second syntax), a serial poll is automatically conducted on the specified device. The function then returns the status byte. Serial polling is described earlier in this chapter.



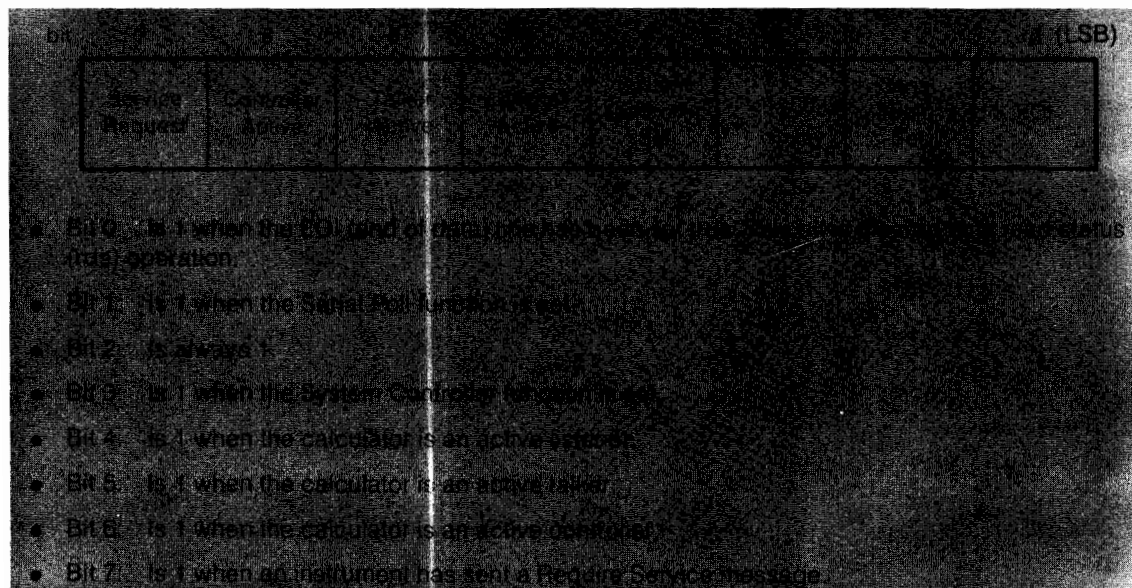
First Status Byte



Second Status Byte



Third Status Byte



Fourth Status Byte

Notes

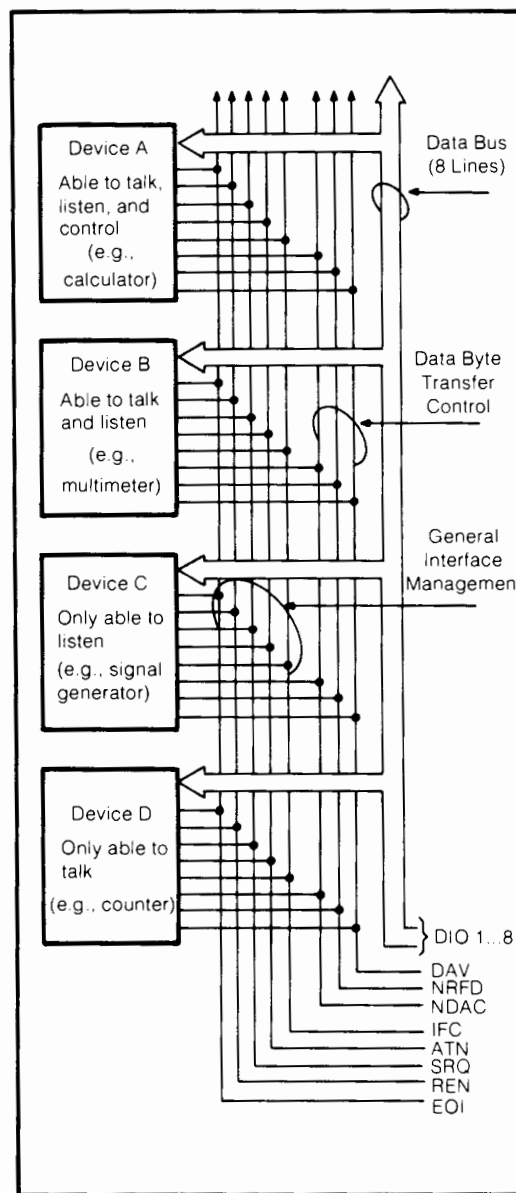
The HP Interface Bus

This section offers a brief overview of the HP-IB hardware and control scheme. You need not read this section, since complete control of the bus is available by using the bus messages described earlier. If a manual for another device on the bus does not describe operation via the bus messages already described, however, this information will help you to determine which messages are needed to control that device.

HP-IB Lines and Operations

The HP Interface Bus transfers data and commands between the components of an instrumentation system on 16 signal lines. The interface functions for each system component are performed within the component so only passive cabling is needed to connect the system. The cables connect all instruments, controllers, and other components of the system in parallel to the signal lines.

The eight Data I/O lines (DIO1 thru DIO8) are reserved for the transfer of data and other messages in a byte-serial, bit-parallel manner. Data and message transfer is asynchronous, coordinated by the three handshake lines: Data Valid (DAV), Not Ready For Data (NRFD), and Not Data Accepted (NDAC). The other five lines are for management of bus activity. See the figure on the right.



HP-IB Signal Lines

Devices connected to the bus may be talkers, listeners, or controllers. The controller dictates the role of each of the other devices by setting the ATN (attention) line true and sending talk or listen addresses on the data lines. Addresses are set into each device at the time of system configuration either by switches built into the device or by jumpers on a PC board. While the ATN line is true, all devices must listen to the data lines. When the ATN line is false, only devices that have been addressed will actively send or receive data. All others ignore the data lines.

Several listeners can be active simultaneously but only one talker can be active at a time. Whenever a talk address is put on the data lines (while ATN is true), all other talkers will be automatically unaddressed.

Information is transmitted on the data lines under sequential control of the three handshake lines (DAV, NRFD and NDAC). No step in the sequence can be initiated until the previous step is completed. Information transfer can proceed as fast as devices can respond, but no faster than allowed by the slowest device presently addressed as active. This permits several devices to receive the same message byte concurrently.

The ATN line is one of the five bus management lines. When ATN is true, addresses and universal commands are transmitted on only seven of the data lines using the ASCII code. When ATN is false, any code of 8 bits or less understood by both talker and listener(s) may be used.

The IFC (interface clear) line places the interface system in a known quiescent state via the Abort message.

The REN (remote enable) line is used with the Remote, Local, and Clear Lockout/Set Local messages to select either local or remote control of each device.

Any active device can set the SRQ (service request) line true via the Require Service message. This indicates to the controller that some device on the bus wants attention, say a counter that has just completed a time-interval measurement and wants to transmit the reading to a printer.

The EOI (end or identify) line is used by a device to indicate the end of a multiple-byte transfer sequence. When a controller sets both the ATN and EOI lines true, each device capable of a parallel poll indicates its current status on the DIO line assigned to it.

In the interest of cost-effectiveness, it is not necessary for every device to be capable of responding to all the lines. Each can be designed to respond only to those lines that are pertinent to its function on the bus.

The operation of the interface is generally controlled by one device equipped to act as controller. The interface uses a group of commands to direct the other instruments on the bus in carrying out their functions of talking and listening.

The controller has two ways of sending interface messages. Multi-line messages, which cannot exist concurrently with other multi-line messages, are sent over the eight data lines and the three handshake lines. Uni-line messages are transferred over the five individual lines of the management bus.

The commands serve several different purposes:

- Addresses, or talk and listen commands, select the instruments that will transmit and accept data. They are all multi-line messages.
- Universal commands cause every instrument equipped to do so to perform a specific interface operation. They include multi-line messages and three uni-line commands: interface clear (IFC), remote enable (REN), and attention (ATN).
- Addressed commands are similar to universal commands, except that they affect only those devices that are addressed and are all multi-line commands. An instrument responds to an addressed command, however, only after an address has already told it to be talker or listener.
- Secondary commands are multi-line messages that are always used in series with an address, universal command, or addressed command (also referred to as primary commands) to form a longer version of each. Thus they extend the code space when necessary.

To address an instrument, the controller uses seven of the eight data-bus lines. This allows instruments using the ASCII 7-bit code to act as controllers. As shown in the table, five bits are available for addresses, so a total of 31 addresses are available in one byte. If all secondary commands are used to extend this into a two-byte addressing capability, 961 addresses become available (31 addresses in the second byte for each of the 31 in the first byte).

Command and Address Codes

Code Form							Meaning
X	0	0	A ₅	A ₄	A ₃	A ₂ A ₁	Universal Commands
X	0	1	A ₅	A ₄	A ₃	A ₂ A ₁	Listen Addresses
			except				
X	0	1	1	1	1	1 1	Unlisten Command
X	1	0	A ₅	A ₄	A ₃	A ₂ A ₁	Talk Addresses
			except				
X	1	0	1	1	1	1 1	Untalk Command
X	1	1	A ₅	A ₄	A ₃	A ₂ A ₁	Secondary Commands
			except				
X	1	1	1	1	1	1 1	Ignored

Code used when attention (ATN) is true (low).

X = don't care

Interface Functions

Interface functions provide the physical capability to communicate via HP-IB. These functions are defined in the IEEE Standard 488-1975. This standard, which is the designer's guide to the bus, defines each interface function in terms of state diagrams that express all possible interactions.

Bus capability is grouped under 10 interface functions, for example: Talker, Listener, Controller, Remote/Local. The following table lists the functions.

HP-IB Interface Functions

Mnemonic	Interface Function Name
SH	Source Handshake
AH	Acceptor Handshake
T	Talker (or TE = Extended Talker)*
L	Listener (or LE = Extended Listener)*
SR	Service Request
RL	Remote Local
PP	Parallel Poll
DC	Device Clear
DT	Device Trigger
C	Any Controller
C _N	A specific Controller (for example: C _A , C _B ...)
C _S	The System Controller

*Extended talkers and listeners use a two-byte address. Otherwise, they are the same as Talker and Listener.

Since interface functions are the physical agency through which bus messages are implemented, each device must implement one or more functions to enable it to send or receive a given bus message.

The following table lists the functions required to implement each bus message. Each device's operating manual lists the functions implemented by that device. Some devices, such as the 98034A Interface, list the functions implemented directly on the device.

Functions Used By Each Bus Message

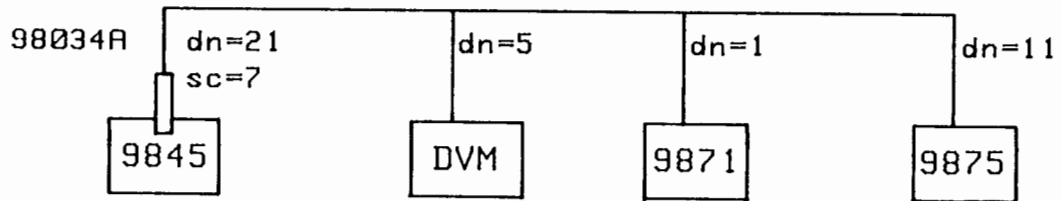
Bus Message	Functions Required
Data	T→L* (SH, AH)
Trigger	C→DT* (L, SH, AH)
Clear	C→DC* (L, SH, AH)
Remote	C _S →RL* (SH, AH)
Local	C→RL* (L, SH, AH)
Local Lockout	C→RL* (SH, AH)
Clear Lockout/Set Local	C _S →RL*
Require Service	SR*→C
Status Byte	T→L* (SH, AH)
Status Bit	PP*→C
Pass Control	C _A →C _B (T, SH, AH)
Abort	C _S →T,L*C

sender function→receiver function(s) (support functions)

*Since more than one device can receive (or send) this message simultaneously, each device must have the function indicated by an *

Notes

HP-IB DIAGRAM



I030

HP-IB ADDRESSING

PURPOSE:

To address peripherals connected to the HP-IB Interface.

FORM:

Replace <s.c.> with <s.c.><d.n.>

EXAMPLES:

OUTPUT 701;A\$
Dvm=705
ENTER Dvm;X,Y

I031

HP-IB DEVICE NUMBERS

RULES:

Each device on the HP-IB has its own device number.

There are switches on each device so you can assign a unique device number to each peripheral.

I032

HP-IB SYSTEM CONTROLLER

RULES:

Each bus system can have only one System Controller, although it can have several devices CAPABLE of being controllers.

A switch on the HP-IB Interface Card can be manually set to make the 9845 the System Controller.

The System Controller can take priority control of the bus when it is not the active controller.

I033

HP-IB ACTIVE CONTROLLER

RULES:

The current Active Controller can pass control to another device.

There can be only one Active Controller in the bus system at any time.

The Active Controller addresses devices as talkers and listeners.



I034

HP-IB TALKERS AND LISTENERS

RULES:

A device is a TALKER if it can send information on the bus when it has been addressed.

A device is a LISTENER if it can receive information from the bus when it has been addressed.

EXAMPLES:

TALKERS	LISTENERS	TALKERS/ LISTENERS
-Signal generators	-Printers	-Plotter/Digitizers
-Voltmeters	-Plotters	-Mass storage media
-Digitizers	-Tape punches	-Programmable DVM's

I035

MULTIPLE LISTENERS

PURPOSE:

To send the same information to multiple devices simultaneously.

FORM:

Replace <s.c.> with <s.c.>,<s.c.>[,<s.c.>....]

EXAMPLE:

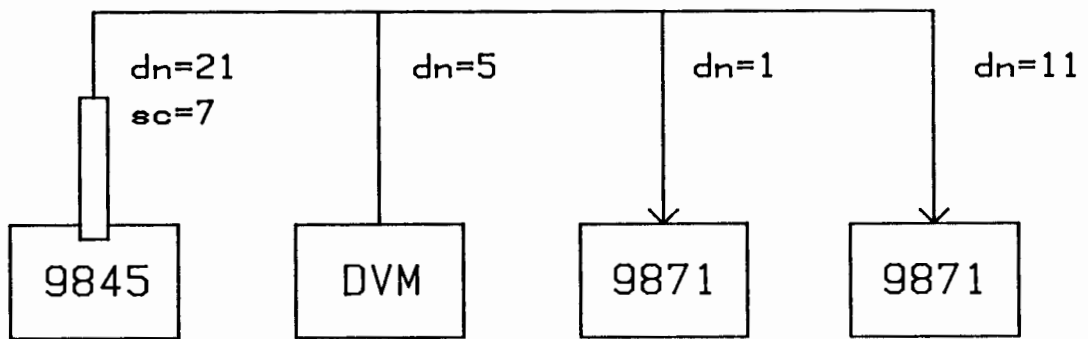
OUTPUT 701,711;A\$

USES:

2 OR 3 Printers producing the same outputs.

I036

MULTIPLE LISTENERS DIAGRAM



AUTOMATIC HP-IB CONTROL

EXAMPLES:

ENTER 704; X, Y
OUTPUT 711; A\$, B\$

RULES:

The System 45 as System Controller automatically sends all of the necessary information to the bus to declare the current talker and listener.

Automatic Control is the easiest way to use the HP-IB.

When the System 45 is NOT the only controller in the system, special statements are needed.

DEVICE DEPENDENT SETTINGS

PURPOSE:

Some devices on the bus may need special set-up instructions before they can participate in the bus system.

EXAMPLE:

Set DVM range before taking readings.
OUTPUT 711 USING "#,K";"R1"
ENTER 711;X,Y

RULES:

The user must set the switches on the front panel of the instrument or write OUTPUT statements to get the instrument ready.

I039

HP9874A DIGITIZER PROGRAM

To digitize points as fast as possible and
display the coordinates on the digitizer display:

```
10 OUTPUT 706; "IN"           !Initialize the 9874
20 Digitize: OUTPUT 706; "OC"  !Outputs coordinates
30 ENTER 706;Xcoord,Ycoord,Pen,Annot !Enter a point
40 IMAGE "LB",DDDDDD,"",",DDDDDD !Format for display
50 OUTPUT 706 USING 40;Xcoord,Ycoord !Write to display
60 GOTO Digitize
70 END
```



I038.4

HP9875A TAPE DRIVE PROGRAM

To store string data in file 2 on the tape:

```
10 DIM A$(100)
20 OUTPUT 704; "RW"      ! Command to rewind tape
30 OUTPUT 704; "MF5,4"  ! Mark 5 files of 1024 bytes
40 INPUT "Enter data:",A$ ! Input string from keyboard
50 OUTPUT 704; "SF2"    ! Command to store in file 2
60 OUTPUT 704; A$       ! Sends data to the 9875A
70 END
```

I038.6

HP-IB INTERFACE STATUS

EXAMPLES:

STATUS 7;A
STATUS 7;A,B,C,D

RULE:

Variables returned for HP-IB Interface Card represent status bytes 4,1,2 and 3, respectively.

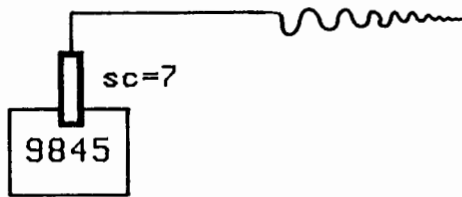
HP-IB DEVICE STATUS

EXAMPLE:

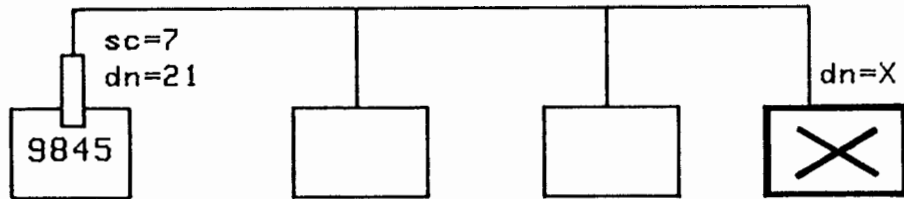
STATUS 711;A

I040

HP-IB STATUS



DEVICE STATUS



I041

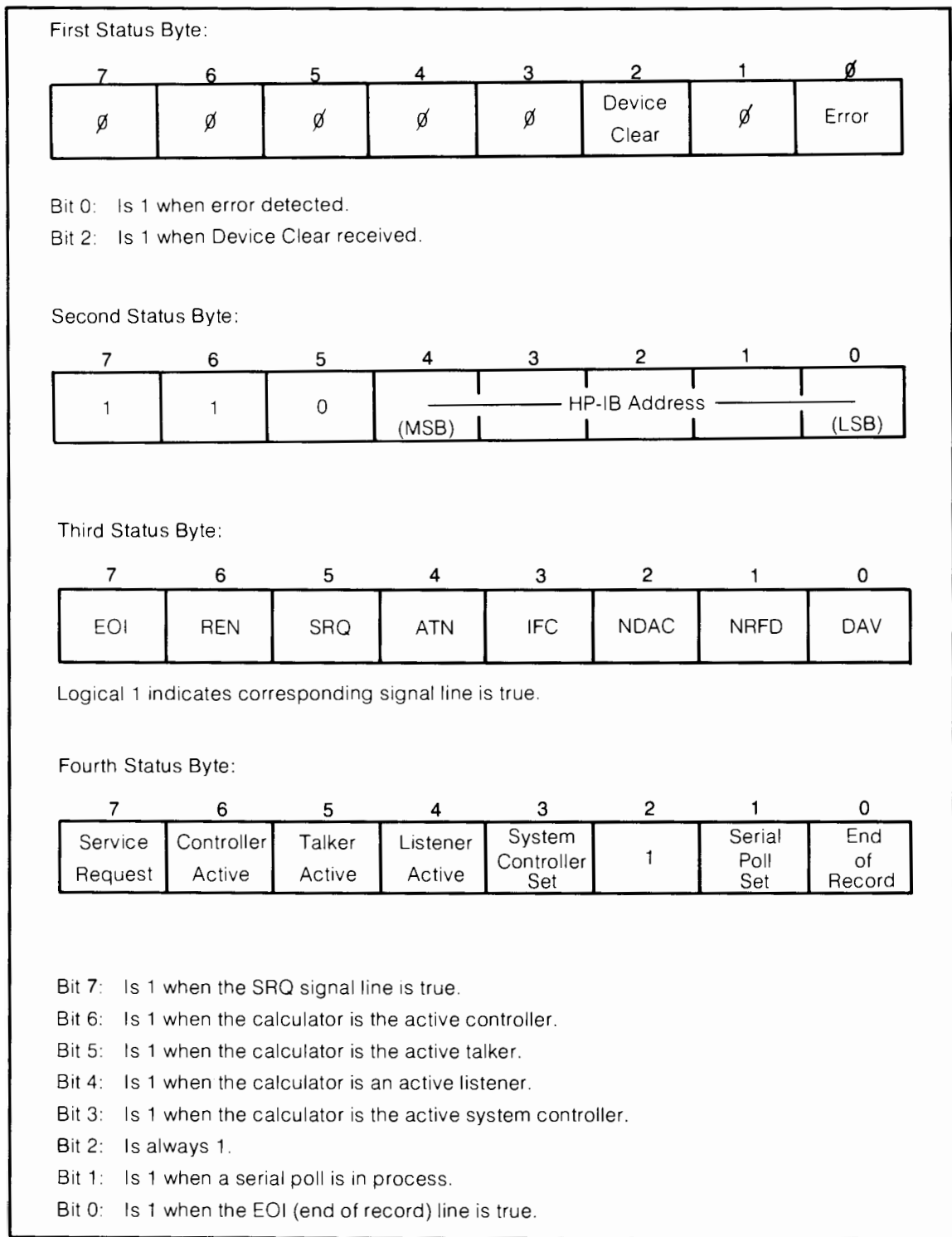


Figure 7. Interface Status Bytes

STATUS 7; A, B, C, D,
 ↑ ↑ ↑ ↑
 4th 1st 2nd 3rd

PROGRAM TO INTERPRET THE HP-IB
INTERFACE DEVICE ADDRESS

```
10 STATUS 7;A,B,C,D  
61 PRINT "Device address is ";BINEOR(2^7+2^6,C)
```

BINEOR:

```
11000000    2^7+2^6  
11010101    Status byte C
```

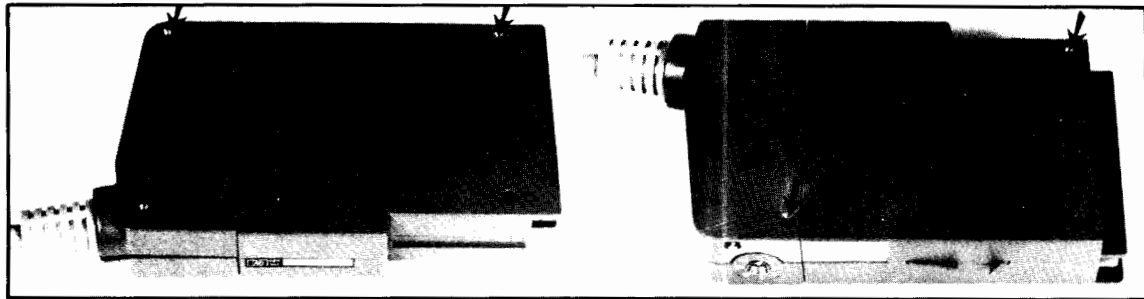
```
-----  
10101 ----> 21    Device address  
           2      10
```

I043

EXERCISE

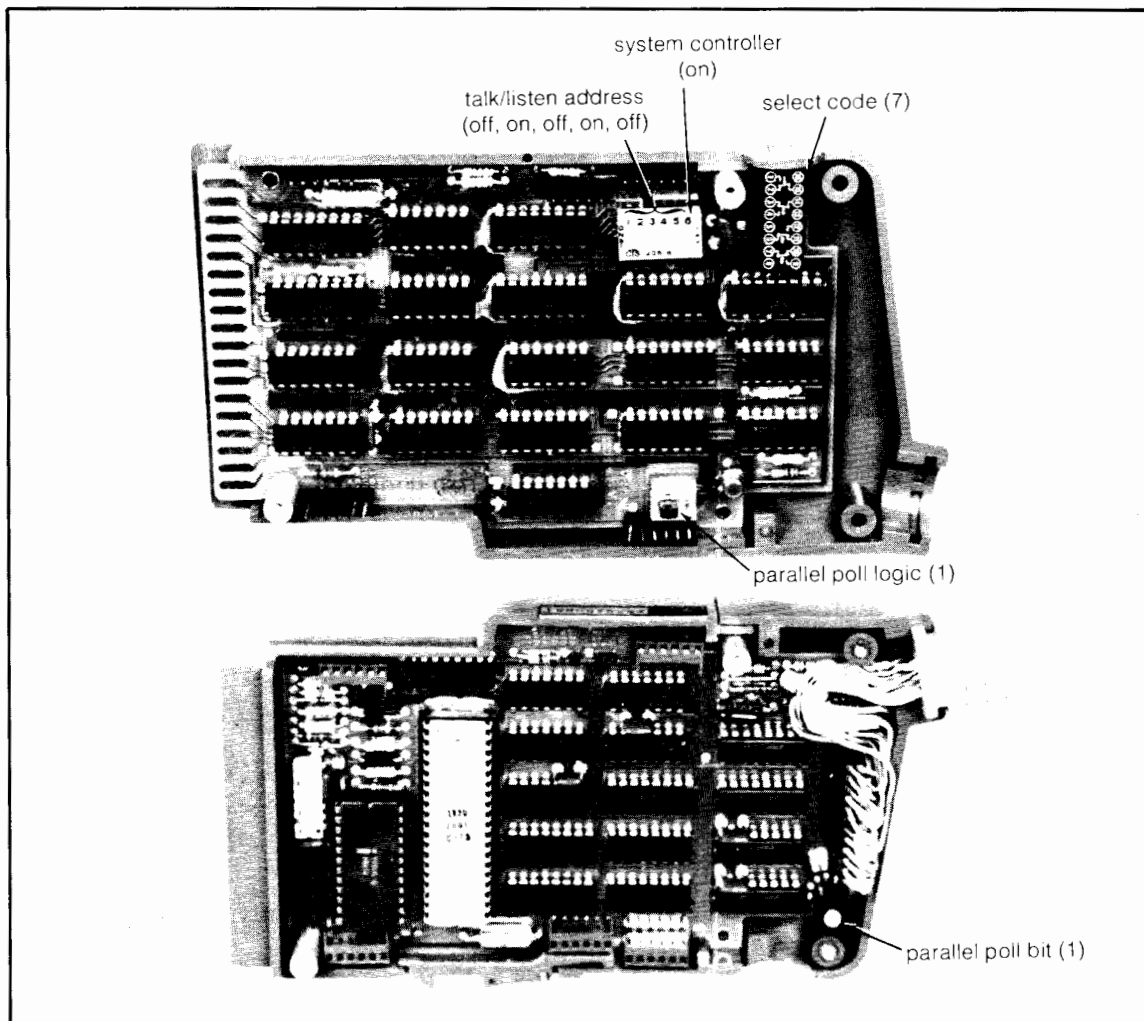
1. Read the STATUS bytes for the HP-IB Interface Card.
2. Compute the HP-IB Device Address assigned to the System 45.

I042



A. Remove only the four screws shown above.

B. Flip the card over and remove these two screws.



C. Separate the case halves and position them as shown.

Figure 4. Opening the Interface Case

SWITCHES INSIDE THE HP-IB CARD

For System Controller on Device Address 21:

Switch settings

(1) (2) (3) (4) (5) (6)
OFF ON OFF ON OFF ON

For NON-Controller on Device Address 20:

Switch settings

(1) (2) (3) (4) (5) (6)
ON ON OFF ON OFF OFF



I044.6

EXERCISE

Connect two System 45's in an HP-IB system: one must be System Controller and the other is the non-controller. The Device Addresses for the two systems must be different. Therefore, we'll assign the System Controller to Device Address 21, and the non-controller to Device Address 20.

1. Following the diagram, open the interface card with a screwdriver.
2. Set the switches for the talk/listen address and system controller as needed.
3. Re-check the device address and the system controller information after re-connecting the interface to the System 45.

I044

Changing Talk/Listen Addresses

The bus interface is set to talk address "U" and listen address "5" at the factory. These may be changed to any talk/listen pair of characters listed in the next table by setting switch S3 (1 through 5) on the A1 circuit board. Setting each slide to the "ON" position corresponds to a "0" in the table.

Table 5. Available Bus Addresses and Codes

Address Characters		Address Switch Settings					Address Codes	
Listen	Talk	(5)	(4)	(3)	(2)	(1)	decimal	octal
SP	@	0	0	0	0	0	0	0
!	A	0	0	0	0	1	1	1
"	B	0	0	0	1	0	2	2
#	C	0	0	0	1	1	3	3
\$	D	0	0	1	0	0	4	4
%	E	0	0	1	0	1	5	5
&	F	0	0	1	1	0	6	6
'	G	0	0	1	1	1	7	7
(H	0	1	0	0	0	8	10
)	I	0	1	0	0	1	9	11
*	J	0	1	0	1	0	10	12
+	K	0	1	0	1	1	11	13
,	L	0	1	1	0	0	12	14
-	M	0	1	1	0	1	13	15
.	N	0	1	1	1	0	14	16
/	O	0	1	1	1	1	15	17
0	P	1	0	0	0	0	16	20
1	Q	1	0	0	0	1	17	21
2	R	1	0	0	1	0	18	22
3	S	1	0	0	1	1	19	23
4	T	1	0	1	0	0	20	24
5	U	1	0	1	0	1	21	25 ← preset
6	V	1	0	1	1	0	22	26
7	W	1	0	1	1	1	23	27
8	X	1	1	0	0	0	24	30
9	Y	1	1	0	0	1	25	31
:	Z	1	1	0	1	0	26	32
;	[1	1	0	1	1	27	33
<	/	1	1	1	0	0	28	34
=]	1	1	1	0	1	29	35
>	^	1	1	1	1	0	30	36

SETTING HP-IB CARD SWITCHES

For NON-Controller on Device Address 20:

- a.) System Controller (Switch 6) = OFF
- b.) Device Address Switch (5) (4) (3) (2) (1)
 1 0 1 0 0
- c.) 1=OFF;0=ON OFF ON OFF ON ON
- d.) Switch settings as they appear on the card:
 (1) (2) (3) (4) (5) (6)
 ON ON OFF ON OFF OFF (left to right)

I045.6

Chapter 3 Table of Contents



Write Binary Statement (wtb)	3-3
Read Binary Statement (rdb)	3-4
Read Status Function (rds)	3-5
KDP Status Bits	3-5
Tape Drive Status Bits	3-7
Interface Status	3-8
Write Control Statement (wtc)	3-9
Extended Binary Operations	3-10
Binary Representation	3-10
Decimal/Octal Modes (moct, mdec)	3-11
Decimal/Octal Conversions (dto, otd)	3-12
Binary AND Function (band)	3-12
Exclusive OR Function (eor)	3-12
Inclusive OR Function (ior)	3-13
Complement Function (cmp)	3-13
Rotate Function (rot)	3-13
Shift Function (shf)	3-14
Add Function (add)	3-15
Bit Function (bit)	3-15
Binary Coding and Conversions	3-17

Notes

Chapter 3

Binary I/O Operations

Binary I/O operations are available to read and write individual data characters, and to transmit or receive control information using interface status lines. Binary I/O operations do not reference format or conversion statements. The data I/O modes and status line meanings are determined by the interface card; refer to each card's installation and service manual for details.

Write Binary Statement

```
wtb select code ; expression or text1 [ ; expression or text2...]
```

This statement outputs the 16-bit, binary-equivalent result of each expression or each character of text. The usable range for each expression is an integer from 0 through 32767.

Here is a short program which uses write binary to print the same title and internal-printer character set as shown on page 1-14. A portion of the printout is also shown below. Notice that a line feed (decimal 10) must be sent at the end of each statement to print the line (line feeds are not automatically sent after binary output operations).

```
0: 0→A
1: wtb 16,"CHARA
  CTER SET",10
2: wtb 16,A,10
3: if (A+1→A)<12
  8;jmp -1
4: end
```

```
CHARACTER SET
◀
◀
X
N
α
S
Γ
π
Δ
α
```

3-4 Binary I/O Operations


The HP 9871A Printer has a variety of functions that are controlled by using sequences of decimal codes. Here are program segments that set the top of form (decimal codes 27 and 84) in line 6 and form length (27,70, int (n/64),int n) in line 7. For a form length of 8 inches, n is 768. Line 15 outputs a form feed instruction. The printer interface is set to select code 6.

```
6: wtb 6,27,84
7: wtb 6,27,70,
  int(768/64),768
  .
  .
  .
15: wtb 6,12
```

Read Binary Function

rdb (select code)

Read binary is a function that inputs one 16-bit character and stores its integer-decimal value. If the interface handles data in an 8-bit fashion, the eight most-significant bits are read as zeros.

For example, to read this ASCII-coded paper tape 

1234,2345,3.45 (LF)

Run the next program 

The program assumes that the HP 9883A Tape Reader's interface responds to select code 3.

```
0: fxd 0
1: prt rdb(3)→A
2: if A#10;jmp -
  1
3: end
```

Notice that the read binary function is programmed as part of the print statement. Line 2 continues the program until a LF is seen. Compare the list of decimal numbers with the ASCII characters in the Reference Tables appendix.

```
49
50
51
52
44
51
52
53
44
51
46
52
53
10
```

As an interesting exercise, execute this line ♦

```
wrt 16,rdb (0);jmp 0
```

Now press each key, except **RESET**, to print its decimal equivalent code. Notice that Live Keyboard is disabled while it is involved in an I/O operation. To halt the calculator, press **RESET**.

The decimal keycodes returned by using this method generally do not correspond to ASCII decimal codes. A complete set of decimal codes is in the Reference Tables appendix.



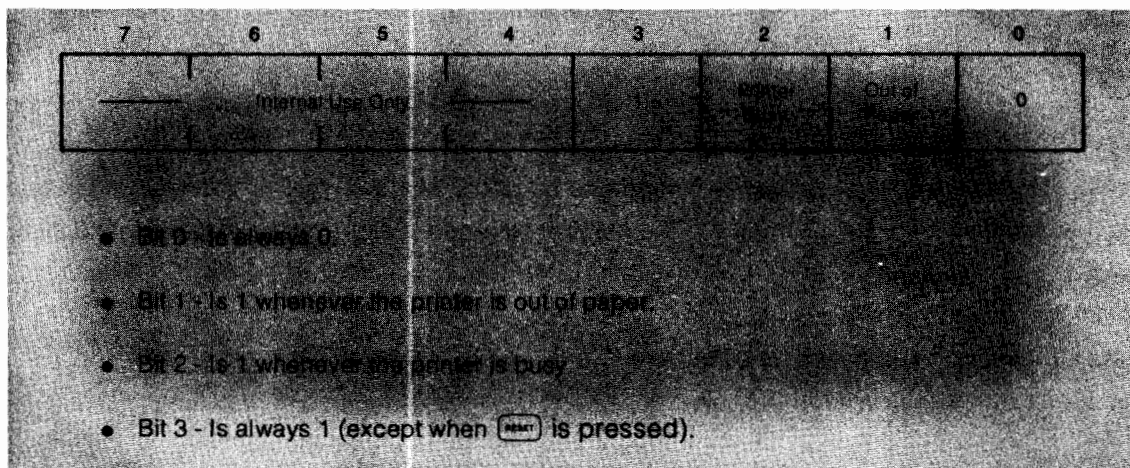
Read Status Function

```
rds (select code)
```

This function reads the current status information transmitted from the specified device and returns a decimal-equivalent number. The number of status bits and their meanings are described in each interface installation and service manual. The status information available from the HP 98032A Interface Card is described later in this section.

KDP Status Bits

Status information from the calculator's keyboard (K), display (D), and printer (P) is combined into an 8-bit byte. Although most of the bits are for internal use only, two bits have programming uses:



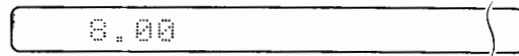
KDP Status Bits

3-6 Binary I/O Operations

So, if paper is loaded in the calculator, store and run this line (use either select code 0 or 16) ♦

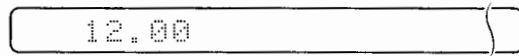
```
0: dsp rds(16);  
wait 100;jmp 0
```

The display should flash continually ♦



Bit 3 will always appear as "1" (binary 8) to the read status function.

Now press **PRT ALL**. Now the display is ♦ since the printer is busy in the print all mode. Press **PRT ALL** again to switch the mode off. Notice that status bit 2 (decimal 4) is added to the byte whenever the printer is busy.



The segment on the right shows how to avoid error 15, out of paper, by watching for the addition of status bit 1 (decimal 2). When a status byte of 10 is returned, the rest of line 0 is executed.

```
1: if rds(16)=10  
;dsp "out of  
paper→→";stp  
2: prt A$
```

The above method works, provided that the printer is not busy. The lines on the right check for both status bytes which indicate "out of paper".

```
3: if rds(16)=10  
;jmp 3  
4: if rds(16)=14  
;jmp 2  
5: jmp 2  
6: dsp "Out of  
Paper→→";stp  
7: prt A$
```

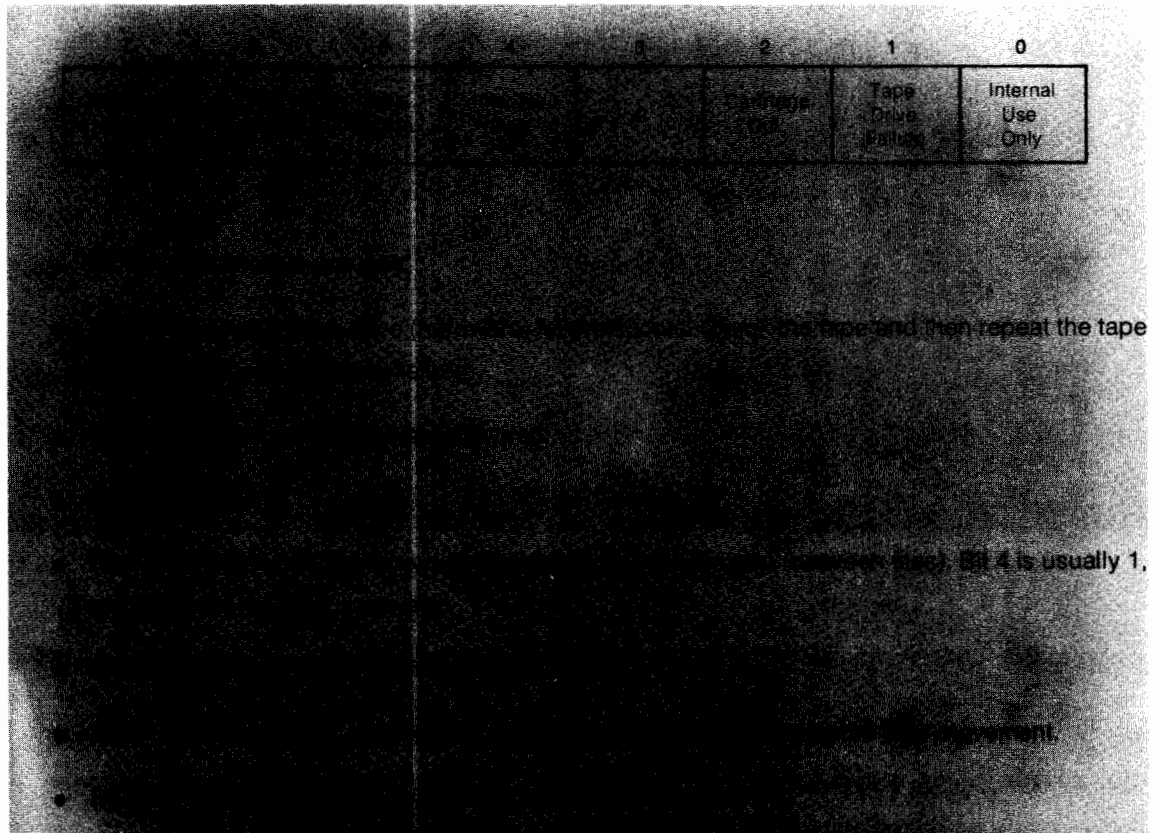
Fortunately, there's an easy way to isolate one bit of the status byte; this expression can be used to isolate bit 1: $\text{int}(n/2)\text{mod } 2$. The number returned is either 0 or 1, reflecting the state of bit 1.

Here's a program segment that's equivalent to the one shown above ♦

```
0: int(rds(16)/  
2)mod2→A  
1: if A=1;dsp  
"Out of Paper→→"  
→";stp  
2: prt A$
```

Tape-Drive Status Bits

The status bits available with the internal tape drive are shown below.



Tape Drive Status Bits

Here is a program sequence that checks for a protected tape before recording data. If bit 7 is 1 (decimal 128), "4--Protected Tape!" is displayed.

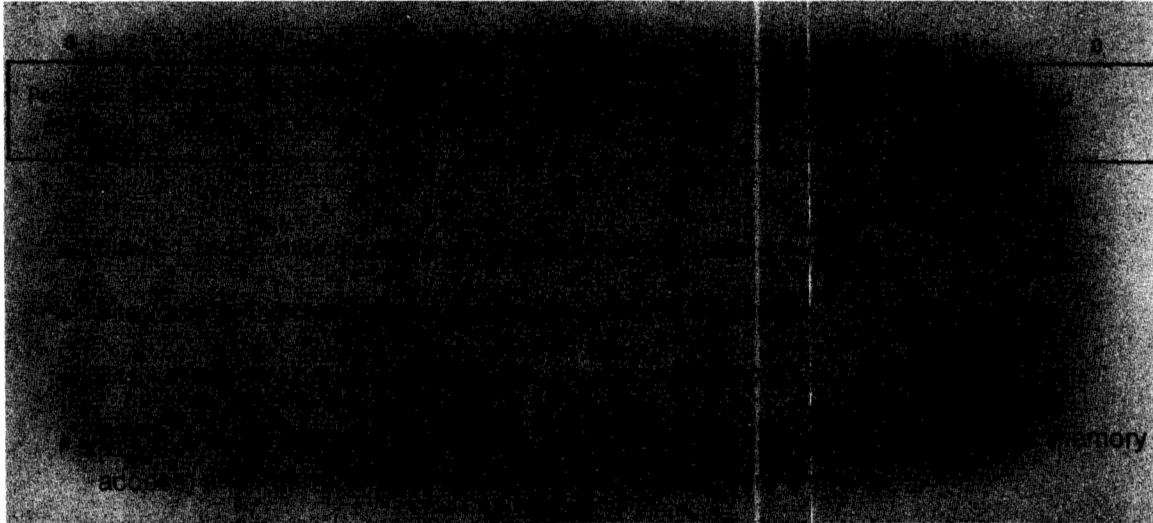
```

7: if rds(1) < 128
8: jmp 2
8: int b, "--Protected Tape!";
wrt 0,0istp
9: rcf 5,A[L,P]

```


Interface Status

The 98032A Interface has nine status bits which can be monitored using read status:



98032A Interface Status Bits

Refer to the 98032A manual for complete details on status meanings and status-input lines in use.

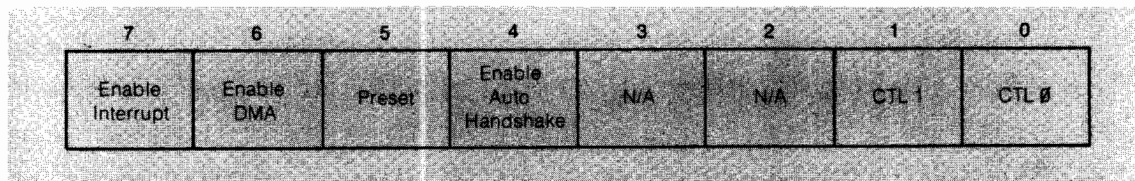
The 98032A Option 066 Interface Card uses bit 8 to indicate when the HP 9866B Printer is out of paper. The Peripheral Status bit remains at 1 (decimal 256) except when the printer is out of paper. Here is a program segment that checks for a 1 at bit 8 before sending data to the printer.

```
4: if rds(6)<257  
   jmp 2  
5: dsp "9866B  
   Out of Paper";  
   stp  
6: wrt 6,A$,B
```

Write Control Statement

`wtc select code: expression`

This statement outputs a binary number to control functions on most interface cards. One number is allowed with each statement. The available bits and their meanings for the 98032A Interface are shown below.



98032A Interface Control Bits

Control bits 0, 1, and 5 are used to drive interface output lines CTL0, CTL1, and Preset. CTL0 and CTL1 are optional peripheral control lines, while Preset is used to initialize the peripheral to its power-up state. A preset signal is automatically given when the calculator is switched on or when **RESET** is pressed. The interface ignores bits 2 and 3. Bits 4, 6, and 7 are usable only with Extended I/O operations. See the interface manual and the following chapters for more details.

Alternate Character Set

The 9825 calculator has an option which provides an alternate character set for the display and printer. The alternate character set is Katakana which is used in Japan.

If the Katakana option is installed, it can be selected by executing `wtc 0:1`. The normal character set is reselected by pressing **RESET** or executing `wtc 0:0`.

Without the optional alternate character set installed, `wtc 0:1` will disable the printer and the display. This should not be done when the calculator is in the print all mode, because the heating elements that produce the dots on the thermal paper may be left in the ON state, and they could be destroyed within seconds.

CAUTION

IF THE PRINTER IS ACTIVELY PRINTING, `wtc 0:1` SHOULD NOT BE EXECUTED BECAUSE IT COULD CAUSE THE PRINTER TO BURN OUT.

Executing `wait 500` before executing `wtc 0:1` will ensure that a print operation is complete before the printer and display are disabled; thus, avoiding the possibility of destroying the printer's heating elements.

Extended Binary Operations

The Extended I/O ROM has 12 functions and statements for manipulating and testing 16-bit binary values. For each function, the value can be any expression whose integer value is in the range of decimal -32768 thru 32767 . Fractional values are handled differently depending on which number mode is currently set. Fractional values are rounded up when the decimal mode (mdec) is set, but fractional values are truncated when the octal mode (moct) is set.

If the value of any parameter is outside of the above range, error E6 will result. If flag 14 is set, however, no error will occur; instead, flag 15 will be set to indicate the overflow and the 16-bit binary result will be used as is. Thus, with flag 14 set, the range of the parameters may be extended to 0 thru 65535 and treated as 16-bit positive binary values rather than the normal 16-bit 2's complement representation (described in the next section).

The binary functions described here should not be confused with the logical operators **and**, **or**, **xor**, and **not** which are described in the calculator operating and programming manual. Each of those operators is used to evaluate expressions and return a 0 or 1, depending on the Boolean operator.

Binary Representation

The 16-bit numbers used for the binary operations explained in this chapter are represented internally as binary numbers. To represent a negative value, the calculator stores the 2's complement of the value. Here is how to find the 2's complement for a value such as -37 decimal. First convert 37 to 16-bit binary (0 000 000 000 100 101). Then complement¹ the value (1 111 111 111 011 010). This intermediate value is the 1's complement of 37. To get the 2's complement, add 1 to the 1's complement. Thus, -37 would be represented as 1 111 111 111 011 011 or octal 177733.

¹To complement a binary number, convert the 0's to 1's and 1's to 0's.

Decimal/Octal Mode Statements

Extended I/O allows you to set the computer in either of two number modes, octal (base 8) or decimal (base 10). Octal notation is explained at the back of this chapter.

Set Octal Mode Syntax: `noct`

Set Decimal Mode Syntax: `ndec`

The currently set mode affects all Extended I/O binary functions, the General I/O binary operations, and all other operations which use 8- or 16-bit binary parameters. A complete list of parameters affected by the number mode follows below. The calculator is automatically set to the decimal mode when it is switched on.

For example, if the data byte 01011101 (binary) is to be read from a device on select code 3:

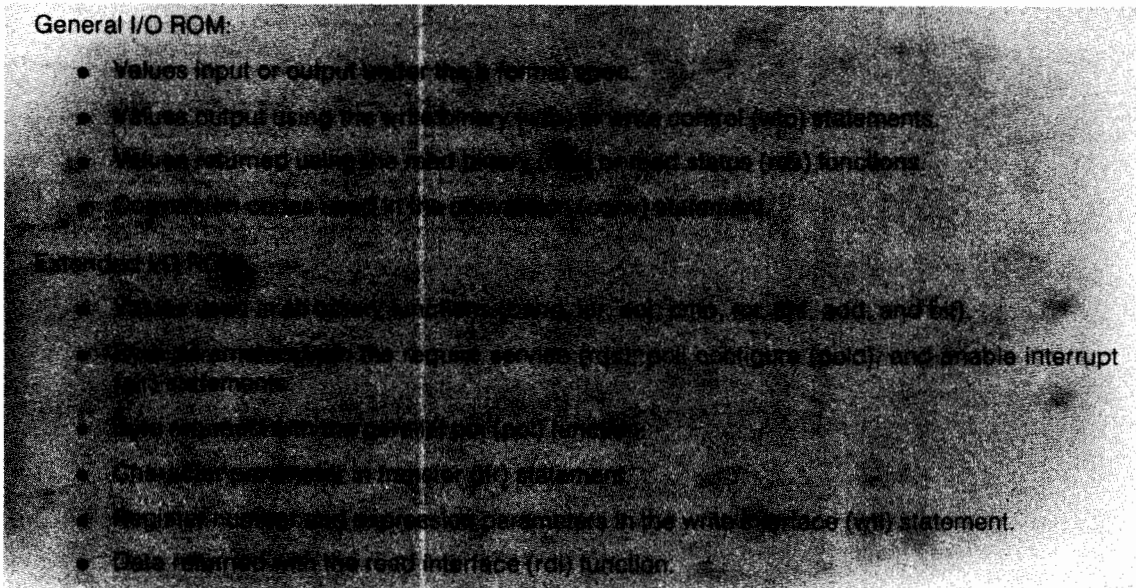
`noct;rdb(3)`

135.00

`ndec;rdb(3)`

93.00

I/O Parameters Affected by the Decimal or Octal Mode



Decimal/Octal Conversion Functions

Two functions are available for converting specified values from decimal to octal form. The working range is from decimal -32768 thru 32767 .

Decimal to Octal Conversion Syntax:

```
dto (expression)
```

Octal to Decimal Conversion Syntax:

```
otd (expression)
```

Binary AND Function

```
band (expressionA , expressionB)
```

The binary AND (**band**) function combines the given values, bit-by-bit, and returns the result.

The truth table for the logical AND operation is shown on the right.

A	B	band (A,B)
0	0	0
0	1	0
1	0	0
1	1	1

For example, to AND decimal 20 and 24, execute these lines:

```
mdec 20>A;dsp A
```

```
20
```

```
(0000 0000 0001 0100)
```

```
24>B;dsp B
```

```
24
```

```
(0000 0000 0001 1000)
```

```
band(A,B)
```

```
16
```

```
(0000 0000 0001 0000)
```

Exclusive OR Function

```
eor (expressionA , expressionB)
```

The exclusive OR (**eor**) function combines the values, bit-by-bit, in a logical exclusive OR operation and returns the result. The exclusive OR truth table is shown on the right.

A	B	eor (A,B)
0	0	0
0	1	1
1	0	1
1	1	0

Inclusive OR Function

`ior (expressionA , expressionB)`

The inclusive OR (`ior`) function combines the values, bit-by-bit, in a logical OR operation and returns the result. The logical OR truth table is shown on the right.

A	B	ior (A,B)
0	0	0
0	1	1
1	0	1
1	1	1

For example, to combine octal 57 and 21 in an inclusive OR operation:

```
noct!57→A!dsp A      57      (0 000 000 000 101 111)
21→B!dsp B           21      (0 000 000 000 010 001)
ior(A,B)              77      (0 000 000 000 111 111)
```



Complement Function

`cmp expression`

The complement (`cmp`) function takes the binary 1's complement of the 16-bit value and returns the result. For example, execute these lines:

```
noct!127→A!dsp A     127      (0 000 000 001 010 111)
cmpA                  177650   (1 111 111 110 101 000)
```

Rotate Function

`rot (expression ++ or -- no. of places)`

The rotate (`rot`) function right-rotates the 16-bit equivalent of the value by the specified number of places and returns the result. The value is rotated left when the number of places is negative. Bit 0 is rotated to bit 15 when the number of places is positive. Thus, no bits are lost when the value is rotated.

For example, execute these lines:

```
mdec;15→A;dsp A      15      (0000 0000 0000 1111)
rot (A,-7)            1920     (0000 0111 1000 0000)
rot (A,3)             -8191    (1110 0000 0000 0001)
```

Here is a sequence which inputs two 8-bit bytes, combines them into a 16-bit word, and prints the resulting value. This sequence could be combined into one statement:

```
20: rdb(3)→A;rdb(3)→B
21: rot(A,8)→A
22: prt ior(A,B)
```

```
prt ior(rot((rdb 3),8),rdb(3))
```

Shift Function

```
shf (expression; + or - no. of places)
```

The shift (**shf**) function shifts the 16-bit binary equivalent of the expression the specified number of places to the right. The value is shifted left when the number of places is negative. Bits shifted left of bit 15 or right at bit 0 are lost.

For example, set A = 255 (0000 0000 1111 1111) and execute these lines:

```
shf (A,5)             7      (0000 0000 0000 0111)
shf (A,-7)            32640   (0111 1111 1000 0000)
```

Here is a sequence which inputs a 16-bit word from a device on select code 3. The word is then printed in four 4-bit segments.

```
0: mdec
1: rdb(3)→A
2: prt "(MSB)",shf(A,12)
3: shf(A,-4)→B;prt shf(B,12)
4: shf(A,-8)→B;prt shf(B,12)
5: prt "(LSB)",band(A,15)
```

Add Function

```
add (expressionA , expressionB )
```

The add function adds the binary equivalents of the two expressions and returns the result. This function is identical to the calculator's + operation for decimal integers. The add function can be used in the octal mode, however, permitting the addition of octal values.

For example, if A = 37 and B = 2, execute these lines:

```
mdec;add(A,B) 39
```

```
moct;add(A,B) 41
```

Bit Function

The bit function is used to test one or more bits of a given value, and return either a 1 to indicate true (all bits match) or 0 to indicate false (no match).

Single Bit Test Syntax:

```
bit (bit position , expression )
```

Multi-bit Test Syntax:

```
bit ("mask " , expression )
```

When a numerical value is given for the first parameter, it indicates to test one bit (position 0 thru 15) in the value; 0 tests the least-significant bit. When the first parameter is text, each bit in the value is tested according to the corresponding character in text: the character 1 requires a 1-bit for a match, the character 0 requires a 0-bit for a match, and any other character indicates not to check that bit. If all specified bits match, a 1 is returned. Up to a 16-bit mask is allowed. If fewer than 16 characters are in the mask, they correspond to the least-significant bits in the value; in this case, any higher bits are not tested.

For example, set A = 65 (0000 0000 0100 0001) and test the eight least-significant bits of A using the mask "0100 0001":

```
mdec;bit("01000001",A) 1 (true)
```


3-16 Binary I/O Operations

Now set A = 66 (0000 0000 0100 0010) and test A using the same mask:

```
bit("01000001",A) 0 (false)
```

As another example, suppose that a tape reader is connected via a 98032A Interface (at select code 2) which sends this status byte in reply to the read status function:

8	7	6	5	4	3	2	1	0 (LSB)
Power On	x	x	1	0	x	x	Tape Loaded	End of Tape

Status bits 7, 6, 3 and 2 indicate interface status conditions. This program could be used to monitor the tape reader and input data only when the reader is powered up (bit 8) and a tape is loaded (bit 1). The first bit function checks only bits 1 and 8 by using a mask. The second bit function checks bit 0 to halt the program when the end of tape is reached.

```
0: dim AS(10,50)
1: rda(2)=A
2: if bit("100001",A)=0;jap -1
3: red 2,A$
4: if bit(0,A);jap -1
5: if (i-1)>50;goto 1
6: end
```

More examples using the bit function are in chapter 2.

Binary Coding and Conversions

Binary is a base 2 number system using only 1's and 0's. By giving the 1's and 0's positional value, any decimal number can be represented. For example, this diagram shows how decimal 41 = binary 101001:

Decimal		Binary					
10^1	10^0	2^5	2^4	2^3	2^2	2^1	2^0
↓	↓	↓	↓	↓	↓	↓	↓
10	1	32	16	8	4	2	1
4	1	1	0	1	0	0	1

Binary-Decimal Conversions

To convert from binary to decimal, the positional values for the 1's are added up. From the above example this would be:

$$2^5 + 2^3 + 2^0 = 32 + 8 + 1 = 41$$

To convert from decimal to binary, the decimal number is repeatedly divided by 2. The remainder is the binary equivalent. For example:

	Remainder (read up)
2 $\overline{)41}$	→ 1
2 $\overline{)20}$	→ 0
2 $\overline{)10}$	→ 0
2 $\overline{)5}$	→ 1
2 $\overline{)2}$	→ 0
2 $\overline{)1}$	→ 1

Octal-Binary Conversions

Octal is a base 8 number system. Octal numbers are often used since conversion from binary to octal and vice-versa is easy when electronic circuits are used.

To convert from binary to octal, the octal number is broken up into groups of three bits (starting from the right). Each 3 bit group represents an octal number.

For example, to convert binary 10110100011001 to octal:

Octal Number	1	4	0	7	2	6
	↓	↓	↓	↓	↓	↓
Binary Number	001	100	000	111	010	110

Notice that only values from 0 through 7 are used in octal.

To convert from octal to binary, the process is reversed:

Binary Number	10	110	100	011	001
	↓	↓	↓	↓	↓
Octal Number	2	6	4	3	1

ASCII

Binary is often used as a code to represent not only numbers, but also alphanumeric characters such as "A" or "," or "?" or "x" or "2". One of the most-common binary codes used is ASCII¹. ASCII is an eight-bit code, containing seven data bits and one parity bit. The plotter uses ASCII for most I/O operations. No parity bit is used. For example:

A complete list of ASCII characters and their octal and decimal representations is given next.

<u>Character</u>	<u>ASCII Binary Code</u>	<u>ASCII Decimal Code</u>
A	01000001	65
B	01000010	66
?	00111111	63

¹ American Standard Code for Information Interchange.

Chapter 4 Table of Contents

Autostart	4-3
Timeout Statement (time)	4-4
On Error Statement (on err)	4-4
Conversion Table Statement (ctbl)	4-6
Substring Conversion Tables	4-8
Parity Statement (par)	4-9
Conversion Protocol	4-10
Interface Control Operations	4-10
Write Interface Statement (wti)	4-11
Read Interface Function (rdi)	4-12
I/O Flag Function (iof)	4-12
I/O Status Function (ios)	4-12
I/O Drivers Example	4-13



Notes

Chapter 4

More I/O

This chapter describes additional Extended I/O operations.

Autostart

If a tape cartridge is installed when the computer is switched on, it automatically executes a load program 0 (`ldp0`) statement from track 0. The autostart routine permits the computer to automatically load and run a supervisory program, which in turn could define special function keys or load other programs without operator instructions. The autostart routine is also performed after a power failure, enabling the computer to automatically reload and restart a program.

The autostart routine may also be initiated by a remote controller on the HP-IB. This is described in Chapter 5 under "Abortive Interrupts".

As an example, suppose the calculator is being used in an environment where power interruptions occur frequently. File 0 on track 0 contains the following program:


```
0: ldm 1
```

File 1 contains a memory file. Periodically, the calculator executes the record memory (`rcm1`) statement, storing the memory in its present state. If the power is interrupted with the cartridge in the transport and the power comes back on, the system can be brought back up without having to start over.



Timeout Statement

`time limit in milliseconds`

The timeout statement specifies a maximum time in which the calculator will wait for a peripheral device to respond to an input or output operation.¹ (Normally, the calculator simply waits until the device becomes ready to send or accept data.) Whenever a device does not respond within the specified time interval, the calculator exits the I/O operation and displays error E4. The time limit can be up to 32767 milliseconds (about 32 seconds). If 0 is specified, the time limit is cancelled. When the timeout routine is in effect,  will not abort the I/O operation. The timeout routine can be cancelled, however, by resetting the calculator (erase command, run command or switch power on).

The timeout routine may be used in conjunction with error recovery to take alternate action if a peripheral is not responding. See the next section.

On Error Statement

`on err "label "`


The on error statement enables an error recovery routine and specifies a label to branch to when a calculator error occurs. Then, when an error is seen, the calculator does not halt and display the error message; instead, it branches to the specified label and assigns values to three read-only variables:

`rom` – The ROM in which the error occurred. 0 = mainframe; an ASCII-equivalent value indicates the letter for plug-in ROM errors (an ASCII-decimal table is in the Reference Tables appendix).

`ern` – The error number.

`erl` – The line in which the error occurred.

For example, if `on err "error"` is executed and then error E2 occurs in line 17, the calculator immediately exits the current line, branches to label "error", and assigns these values to the error recovery variables: `rom=69, ern=2, erl=17`.

The error recovery routine is cancelled after the calculator branches to the specified label. Another on error statement must be executed to re-enable the routine. Resetting the calculator (, run command, erase command, or switch power on) also cancels the routine.

¹I/O operations used with Buffered I/O (transfer statement) are not affected by the timeout routine. Buffered I/O is described in Chapter 6.



NOTE

The on err statement should not be placed in the first line of the error recovery routine; if it is, the calculator may continuously loop in the routine when an error occurs in that line.

Here is a short program which reads and prints data readings from a digital voltmeter at select code 3. Line 0 specifies a time limit of one second for each I/O operation. The on error statement before each I/O operation specifies a routine to branch to when an error is seen.

```

0: time 1000
1: on err "dvm error"
2: red 3,A,B,C
3: on err "alt-prt"
4: wrt 6,A,B,C
5: gto 1
6: "dvm error":dsp "DVM DOWN";stp ;gto 1
7: "alt-prt":
8: if rom=69 and ern=4;prt "TIMEOUT";gto 3
9: if rom=71 and ern=8;prt "PRINTER DOWN"
10: if rom=71 and ern=9;prt "CHECK INTERFACE"
11: prt A,B,C
12: gto 1
13: end

```

The "dvm error" routine displays DVM DOWN and halts the program whenever any error occurs in line 2. The "alt-prt" routine however, checks the error recovery variables and prints the error which occurred. Then it prints the three items and continues the program.

Notice in each case that the on error statement must be re-executed to reset the error recovery routine after an error occurs.

NOTE

The on error statement cannot be used to trap errors if the program is stopped for an enter statement. For example, in the following program you enter a string that exceeds two characters. When you press CONTINUE, Error S9 is displayed and the program stops. The program will not branch to error routine "E" because the error occurred on an enter statement.

```

0: dim B$(2)
1: on err "E"
2: ent "B$",B$
3: prt B$;stp
4: "E":dsp "Error";stp
5: end

```


Conversion Table Statement

```
ctbl [string variable name]
```

The conversion table statement assigns a pre-dimensioned string variable to act as a conversion table for all I/O operations. This statement sets up a conversion table completely separate from the conversion (conv) statement, which is intended for conversion of delimiters, etc. with read and write statements only. The ctbl statement can be used to set up a table of any length up to 256 characters, allowing conversion to or from foreign (non ASCII) code.

To use the conversion table statement, a string variable must first be dimensioned and filled with the ASCII characters to be converted. The position, or index, of each ASCII character in the string corresponds to the value of a foreign-code character. These positions are all offset by one, however, to allow conversion of a binary zero in the foreign-code set. Thus, the first character in the table corresponds to a foreign code of 0, the second character corresponds to a foreign code of 1, etc. Once the string is filled, the ctbl statement assigns the string as a conversion table.

For input, each data character is read from the peripheral and used as an index into the conversion table; the content of that location is then substituted for the input character. For output, the table contents are searched (starting from the first character) for the character being output; when it is found, the index of the character at that location is used as the output code. If the ASCII code being searched for is not found, the code is sent untransformed. For input conversion, if the character code read is larger than the size of the currently established conversion table, the code is not converted.

Only one conversion table at a time is active. Executing another ctbl statement cancels the former table and establishes the new one. A ctbl statement with no parameters cancels any previous conversion string. This table should only be activated for the duration of the I/O operation requiring the foreign code set; it should then be de-activated.

NOTE

I/O operations will also reference conversion (conv) and parity (par) statements in addition to referencing ctbl. Refer to "Conversion Protocol" later in this chapter.

For example, suppose that you wish to read and print sets of X-Y values from a paper tape punched in EIA¹ code. Each value is separated by a comma and each set of values is followed by a carriage return (CR). Here is the complete foreign code to be used:

Character	Decimal Equivalent	
	EIA	ASCII
0	32	48
1	1	49
2	2	50
3	19	51
4	4	52
5	21	53
6	22	54
7	7	55
8	8	56
9	25	57
,	59	44
.	107	46
Carr. Ret.	128(CR)	10(LF)

The ASCII decimal-equivalent values were found by using the ASCII table at the back of this manual. Notice that the ASCII line feed was entered instead of carriage return, since the calculator ignores CR but responds to LF as a terminating delimiter during free-field read.

Now a conversion table can be set up by using the table above. First, dimension a string variable having one more element than the largest foreign code value to be converted (decimal 128 in this case):

```
0: dim C$(129);1+1
```

Next, fill the string with spaces (ASCII decimal 32) so that other characters can be assigned to individual positions:

```
1: " " > C$(1,129)
```

Then store each ASCII character in the string position determined by the corresponding EIA decimal value. (Remember that the string position is the foreign code value plus 1.) Either of these methods can be used:

¹Electronic Industries Association standard.

```

2: char (48)→C$[33,33]
3: char (49)→C$[2,2]
4: char (50)→C$[3,3]
5: char (51)→C$[20,20]
6: char (52)→C$[5,5]
7: char (53)→C$[22,22]
8: char (54)→C$[23,23]
9: char (55)→C$[8,8]
10: char (56)→C$[9,9]
11: char (57)→C$[26,26]
12: char (44)→C$[60,60]
13: char (46)→C$[108,108]
14: char (10)→C$[129,129]

```

```

2: "0"→C$[33,33]
3: "1"→C$[2,2]
4: "2"→C$[3,3]
5: "3"→C$[20,20]
6: "4"→C$[5,5]
7: "5"→C$[22,22]
8: "6"→C$[23,23]
9: "7"→C$[8,8]
10: "8"→C$[9,9]
11: "9"→C$[26,26]
12: ", "→C$[60,60]
13: ". "→C$[108,108]
14: char (10)→C$[129,129]

```

Finally, the string can be assigned as a conversion table:

```
15: ctbl C$
```

With the above instructions, definition of the conversion table is complete. Since all remaining elements in the string are defined as spaces, the table will convert any EIA character not in the set to an ASCII space. The calculator, in turn, will ignore all spaces when reading with the free-field format.

Once the conversion table is assigned, all I/O operations which follow will reference it. For example, the next sequence will read ten sets of X-Y values, automatically reference the conversion table as each character is input, and print the converted data items. Line 21 cancels the table so that other I/O operations do not reference it.

```

16: I←I
17: red 3,X,Y
18: prt X,Y;spc 2
19: If (I+I+I)≤10;jmp -2
20: ctbl
21: end

```

This conversion table could also be used to output numeric data, converting **only** the characters in the string to EIA (see the previous table) and passing all other characters, unchanged, in ASCII. Note, however, that ASCII space (decimal 32) would be converted to binary 0 (ASCII NULL), since the first position in the string contains decimal 32.

Substring Conversion Tables

The conversion table need not be based on an entire string. The index of the conversion table is based on the string specified by the `ctbl` statement and not on the absolute locations of characters in the original string variable.

As an example, here is a sequence which dimensions and fills a string with the entire ASCII character set, but then sets up a conversion table using only ASCII A through Z (decimal 65 through 90):

```
0: dim A$(129)
1: char(I)+A$(I+1,I+1);jmp (I+1+I)>128
2: ctbl A$(66,91)
```

So a 26-character conversion table is set up so that:

A = 0, B = 1, C = 2, D = 3, ...

Parity Statement

`par` parity type

The parity statement enables a parity check routine for input data and a parity-bit generating routine for output data. The parity type specifies the routine:

- type 0 – parity disabled.
- type 1 – parity always 1.
- type 2 – parity even.
- type 3 – parity odd.

When parity is enabled, the output data is masked to 7 bits; then the specified parity bit is calculated and set as the 8th bit. Input data is checked for the proper parity type; error E7 is displayed if the parity bit is not correct.

The parity routine should only be active when using ASCII or another 7-bit code. If parity is active during 8-bit or higher data transfers, erroneous results will occur. If a parity type outside the above range is given, only the two least-significant bits of the binary representation of the given parity type will be used. Thus, `par 5` is the same as `par 1`. Execution of `par 0` turns off the parity routine.

For example, the following program sequence inputs 50 data items from a paper tape punched in ASCII with even parity. If an error occurs in line 1, the program jumps to the error recovery routine, disables parity, and checks the error recovery variables. If error E7 has occurred, the calculator displays BAD DATA. If any other error has occurred in line 1, however, the rest of the routine displays an error message.

```

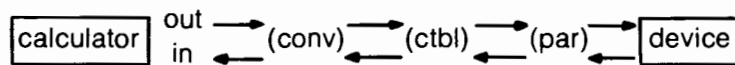
0: par 2;on err "error"
1: red 2,A;prt A;jmp (I+I->I)=50
2: gto 0
3: "error";par 0
4: if con=09 and err=7;dep "BAD DATA";beep;stp ;gto 0
5: con=0;if A=0;dep "
6: int 0,2f.0
7: wrt 0,"ERROR ",A,arn," IN LINE ",erl;beep;stp
8: end

```

Notice that parity is disabled at the beginning of the "error" routine, so that it will not be referenced by succeeding I/O operations.

Conversion Protocol

When more than one of the conversion routines: conversion (conv), conversion table (ctbl), or parity (par) are active at the same time, they are executed in the following order. Remember that conv is referenced by only read (red) and write (wrt) statements, while ctbl and par are referenced by all General I/O and Extended I/O ROM input/output operations.



Conversion Protocol

Remember also that ctbl and par are not applied to data being transferred between buffers and peripherals, as described in Chapter 6. These conversions are applied at the time that data is being written to, and read from the buffer using the normal read and write operations. So the buffer always contains an exact representation of the data that came from, or is going to, the peripheral.

Interface Control Operations

The following four operations allow direct transfer of data or status information between the calculator and the control registers on each interface card. It should be noted that, since these operations are fundamental I/O routines, the user must completely understand the function and I/O protocol of each interface card and peripheral device. If not, unexpected and/or unwanted results could occur!

CAUTION

UNEXPECTED (AND SOMETIMES UNWANTED) RESULTS CAN OCCUR WITH THE WTI STATEMENT IF THE USER DOES NOT FULLY UNDERSTAND THE REQUIREMENTS OF THE DEVICE BEING ADDRESSED.


Write Interface Statement

`wti 0; select code`

or

`wti register no.; expression`



The first syntax is used to specify a select code for successive write interface and read interface operations. The specified select code remains set until either another one is specified or the calculator is reset ( , run command, erase command, or switch power on).

The second syntax is used to output 16-bit binary values directly to the pre-specified interface. The register number may be an integer from 4 thru 7. The binary equivalent of the expression is sent to the specified register. A general description of the interface control registers is given below. For more details, refer to the appropriate interface installation and service manual.

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

Extreme care should be taken when the wti statement is used with the select code 0 or 1. These are the addresses of the internal display and tape cartridge, respectively, and require special I/O and data protocol.

Read Interface Function

`rdi (register number)`

This function returns the 16-bit binary **equivalent value** of the interface register specified. The select code currently set by a **previous write interface** statement determines the interface addressed. `rdi (0)` returns the currently set select code parameter.

I/O Flag Function

`iof (select code)`

This function returns a 1 or 0, indicating the state of the specified interface flag (FLG) line: 1 usually indicates that the peripheral is ready; 0 indicates that the peripheral is busy.

I/O Status Function

`ios (select code)`

This function returns a 1 or 0, indicating the state of the specified interface status (STS) line: 1 usually indicates that the peripheral is functioning; 0 indicates an error condition. Refer to the interface installation and service manual for more details.

I/O Drivers Example

Using the write interface (wti) statement, the read interface (rdi) function, the I/O flag (iof) function, and the I/O status (ios) function, the input and output drivers can be simulated for the 98032A and 98033A interfaces.

This program example imitates the output drivers:

```
0: "Output subroutine":  
1: ent "Select code?",S  
2: ent "Data?",D  
3: wti 0,S  
4: if ios=0;gsb "down"  
5: if iofS=0;jmp 0  
6: if bit(2,rdS);cmpD-D  
7: wti 4,D  
8: wti 7,0  
9: ret
```

This program example imitates the input drivers:

```
0: "Input subroutine":  
1: ent "Select code?",S  
2: wti 0,S  
3: if ios=0;gsb "down"  
4: if iofS=0;jmp 0  
5: rdi 4-D  
6: wti 7,0  
7: if iofS=0;jmp 0  
8: rdi 4-D  
9: if bit(3,rdi 5);cmpD-D  
10: ret
```


Notes

Chapter 5

Table of Contents



Introduction	5-3
The Programmable Interrupt Scheme	5-3
Vectored Interrupt	5-4
On Interrupt Statement (oni)	5-5
Enable Interrupt Statement (eir)	5-6
Interrupt Return Statement (iret)	5-7
Example Application	5-7
HP-IB Interrupt Control	5-8
Abortive Interrupts	5-11
Interface Control Bits	5-13
Interrupt Lockouts	5-14
Variables with Interrupt Service Routines	5-14

Notes

Chapter 5

Interrupt Control

Introduction

The Extended I/O ROM provides the 9825 Computer with the ability to run user-written programs in various interrupt modes. That is, normal program execution may be interrupted to perform other program lines at the request of external devices.

There are two types of interrupt capability: programmable and automatic. Programmable interrupt is available for you to write routines for controlling peripheral devices and transferring data via special interfaces, such as the HP-IB. Automatic interrupt is a built-in feature of certain I/O operations, providing them with a higher level of I/O control than a programmable interrupt scheme could allow.

This chapter describes the three statements which provide you with programmable interrupt capability: the on interrupt (oni), enable interrupt (eir), and interrupt return (iret) statements. Chapter 6, *Buffered I/O*, shows how automatic interrupt is used with the I/O buffer feature for handling data transfers in various formats. Automatic interrupt is also used by the calculator keyboard, and has priority over programmable interrupt routines.

The Programmable Interrupt Scheme

The program lines which perform an interrupt service task are called a "service routine". The oni statement is used to specify an interface card and the location in read/write memory of a service routine to be executed when that interface card interrupts the calculator. The eir statement is used to enable, or allow the interface card to interrupt when its peripheral device requires service. The conditions which actually determine when the interface will interrupt depend upon the interface card and the eir parameters, as described later.

When the interface card interrupts, the calculator "logs in" the request for service, disables the interface from further interrupts, and branches to the pre-specified service routine after completing the current program line. The service routine must be terminated with an iret statement, which returns program control to the line which would have been executed next if the interrupt hadn't occurred.

5-4 Interrupt Control

The general set up for an interrupt service routine is as follows:

```
oni (specify interface and label of service routine)
eir (enable interface card to interrupt)
    •
    •
    •
    (main program)
    •
    •
    •
"label": (service routine lines)  iret
```

The service routine can be any number of program lines needed to service the interrupting device. The last line must be terminated by an iret statement. The iret must not be executed except when accessed via interrupt control.

The calculator normally branches to each service routine between lines of the main program. This is called End of Line (EOL) branching, and is described in the following sections. For extreme cases, an "abortive interrupt" routine can be initiated, which causes the calculator to immediately branch to the service routine. This is explained under "Abortive Interrupts".

Vectored Interrupt

The calculator I/O structure provides for two levels of EOL interrupt priority, based on interface select codes: select codes 2 thru 7 have low-level priority, while select codes 8 thru 15 have high-level priority. Automatic interrupts from the keyboard and the I/O Buffer feature (see Chapter 6) are given priority over these high/low levels.

As the calculator is executing each program line, it logs in each interrupt request and assigns it a priority. If more than one interrupt within a priority level is received, the one with the highest select code is given highest priority. Then, at the end of the current program line, the calculator compares any interrupts logged in with the current interrupt routines (if any) being executed: if a new interrupt has a higher priority than the current routine, the calculator branches to the new routine. But if the new interrupt is of equal or lower priority, the calculator continues with the next program line of the current service routine.



For example, if a low-level interrupt routine is being serviced and a high-level interrupt comes in, it does not need to wait until the low-level routine is finished. Rather, at the end of the current line of the low-level service routine, control passes to the high-level routine. When the high-level routine finishes (by executing its `iret` statement), control passes back to the low-level routine to finish its service. The `iret` from that routine then returns control back to the main program. Had another high-level interrupt logged in while the first high-level routine was in progress, its priority would not be sufficient to interrupt that routine. When the first high-level routine finished, however, the second high-level routine would have been executed entirely before the calculator returned to finish the low-level routine.

Notice that within this EOL branching scheme, any interrupts logged in within a program line are considered simultaneous interrupts. So if (within one line of the program) select code 4 interrupts and logs in, followed immediately by select code 6, they would both be logged in by the end of the line, and select code 6 would be granted service first, even though it interrupted slightly after select code 4. Once the service routine for select code 6 has started, however, a new interrupt from, say select code 7, would have to wait for the select code 6 service routine to be completed before being granted a branch to its service routine.

A line executed under the live keyboard mode takes priority over all service routines. It will be executed at the end of the current program line, regardless of the current interrupt-level being serviced. So the operator is never "locked out" by the EOL branching scheme, unless the live keyboard has been previously disabled using the `lkd` statement.

The On Interrupt Statement

```
oni select code # "label " or string variable [ # abort byte]
```

This statement establishes linkages between each select code that will interrupt and the location of a service routine in the program.

Only interfaces can interrupt, not internal devices. The select code must be a value from 2 thru 15. Since the keyboard (select code 0) and the tape drive (select code 1) are handled automatically by the calculator, they are not available as interrupting devices.

The label or string variable specifies the first line of a service routine beginning with the matching label. A line number cannot be used to specify the location of a service routine.

The optional abort byte is described later, under "Abortive Interrupts".

For example, this statement specifies to branch to the service routine labelled "plot" when the interface at select code 5 interrupts.

```
0: oni 5,"plot"
```


This sequence sets up the same branch, but uses a string variable to specify the label name:

```
0: dim A$[5];"plot"->A$
1: oni 5,A$
```

At any time (including within the service routine itself) another oni statement for the same select code may be executed, either to re-establish a new location for interrupts from that device or to modify the abort byte. Each oni for a given select code cancels any previous oni for the same select code.

Enable Interrupt Statement

```
eir select code [ : interrupt enable byte]
```

Once a service routine and an interface have been specified via the oni statement, the eir statement actually enables the interface to interrupt. This is accomplished by writing the interrupt enable byte to the interface control register R5 out. When the computer is switched on or  is pressed, all interfaces are disabled from interrupting the computer.

When the enabled interface interrupts, the calculator logs in the fact that the interface would like service, and then disables the interface from further interrupts. This is to prevent the interface from continually interrupting until its service routine has been executed. If you wish to enable the interface for further interrupts after the service routine is complete, another eir statement with the desired interrupting conditions specified should be executed before exiting the service routine (i.e., before the iret statement). This provides for repeated calls to the service routine each time one of the specified conditions occurs.

To disable an interrupt, set the interrupt enable byte to zero (e.g., eir 7,0). The conditions for which an interface can interrupt depends on the interrupt enable byte and the type of interface. For example, specifying an interrupt enable byte of 128 (octal 200) sets control bit 7 on the 98032A Interface, causing it to interrupt whenever its peripheral device is ready for more operations (indicated by the peripheral flag line being true). So whenever an eir statement is executed and no interrupt enable byte is given, a byte to set control bit 7 is automatically given. The transfer statement (chapter 6) automatically enables an interrupt.

Since most peripherals connected via the 98032A Interface indicate "ready" most of the time, the programmable interrupt scheme is not suited for data transfer operations with them. (An

exception is the 9862A Plotter, as described later.) The automatic interrupt scheme is specifically designed for data transfer with these devices. Examples of data transfer under automatic interrupt control are in Chapter 6.

Interrupt Return Statement

```
iret
```

This statement is always the last statement executed in an interrupt service routine. It causes program control to return to the program line that would have been executed next if the interrupt had not come in. Although `iret` is the service routine equivalent of the `ret` statement for a subroutine called by a `gsb` statement, the two statements cannot be mixed; so a `gsb` call must end with a `ret` statement, and a branch to an interrupt service routine (initiated by the interface card) must end with an `iret` statement.

Example Application

The 9862A Plotter is a device which requires data in non-ASCII code and in a special format. The 9862A Plotter ROM generates the special code to control the plotter. Although the plotter is a comparatively slow device (due to its mechanical plotting requirements), it is connected via the 98032A Interface, so the programmable interrupt scheme can be used to speed up program execution time by reducing the time spent waiting for the plotter.

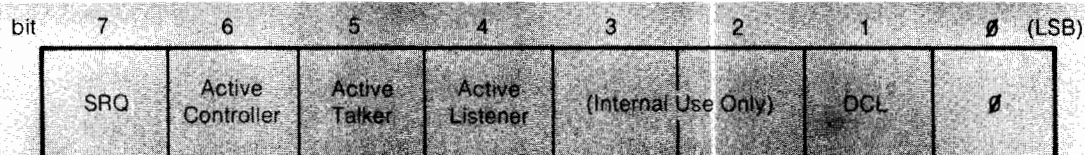
The following sequence shows one method to allow the calculator to rapidly calculate and store data points in an internal array, while a service routine outputs the points to the plotter. This enables the calculator to perform lengthy calculations, or even control other I/O devices, while also driving the plotter at its fastest rate.

```
0: dim A[1000,2]
1: oni 5,"plot"
2: 0→J;1→I
   ●
   ●
10: if I>1000;gto 21
   ●
   ●
18: X→A[I,1];Y→A[I,2]
19: I+1→I
20: eir 5;gto 10
21: if J+1<I;eir 5;jmp 0
22: pen;plt 100,100
23: end
24:
25: "plot":
26: if J+1>=I;jmp 2
27: J+1→J;plt A[J,1],A[J,2]
28: iret
```


- Line 0 – Dimensions an array to hold 1000 sets of plotter coordinates.
- Line 1 – Sets up an interrupt routine such that the program will branch to label “plot” whenever the plotter is ready for another set of coordinates.
- Line 2 – Initialize J, the output pointer, and I, the input pointer.
- Line 10 – Check if computation of plot coordinates is complete.
- Line 11 thru 18 – Compute one set of X-Y coordinates.
- Line 19 – Increment input pointer.
- Line 20 – Enable interrupt 5 and continue computing X-Y coordinates.
- Line 21 – Plot computation is complete, so continue in loop until output pointer plus one equals input pointer.
- Line 22 & 23 – Lift pen & move out of way at the end.
- Lines 25 thru 28 – Interrupt routine to plot each point.

HP-IB Interrupt Control

The 98034A HP-IB Interface can interrupt for a variety of conditions, each of which may be independently enabled (in any combination) by specifying the appropriate interrupt enable byte in an eir statement. The byte is specified as a decimal (or octal when the octal mode is set) equivalent of an 8-bit binary value, where each bit specifies an interrupt condition as shown below.



- Bit 7: Interrupt on Require Service (SRQ) Message.
- Bit 6: Interrupt on becoming Active Controller.
- Bit 5: Interrupt on becoming Active Talker.
- Bit 4: Interrupt on becoming Active Listener.
- Bit 3: Interrupt on input register full.¹
- Bit 2: Interrupt on output register empty.¹
- Bit 1: Interrupt on Clear (DCL) message.
- Bit 0: Always set to 0.¹

¹These bits are for internal use only; see text.

98034A Interrupt-Enable Byte

When the calculator is the active controller on the bus, the Require Service (SRQ) message (bit 7) is the only interrupt condition needed from a device; therefore bit 7 is automatically set when an interrupt byte is not specified. This is equivalent to specifying a byte of decimal 128 or octal 200.

Bits 1, 4, 5, and 6 are useful when the calculator is not the active controller on the bus. This allows the program to go on with other tasks, but to be interrupted when the controller addresses the calculator (as a peripheral device) to talk, listen, respond to a Clear (DCL) message, or take active control. Bits 0, 2, and 3 are used by automatic interrupt control routines only, and should not be set by the user.

When an interrupt enable byte is specified for a 98034A Interface, bits 1, 4, 5, and 6 are not automatically cleared when the calculator logs in an interrupt from the card. Only bit 7 (SRQ) is cleared automatically. So an interrupt is enabled again whenever any of those bits (1, 4, 5, and 6) are set, until either another eir statement changes the byte or the calculator is reset.

For example, suppose that you wish to monitor three devices that can send Require Service messages and respond to a parallel poll via the HP-IB. When parallel polled, device X responds by sending status bit 4, device Y sends bit 2, and device Z sends bit 0. The parallel poll (pol) function returns all bits in one 8-bit byte. Here is a sequence which sets up service routine "SRQ" to parallel poll the bus and then branch to a subroutine to service each device. (Bus polling methods are described in chapter 2.)

```

10: oni 7,"SRQ"
11: eir 7
    ●
    ●
    ●
40: "SRQ":pol(7)+P
41: if bit(4,P)=1;gsb "SVC X"
42: if bit(2,P)=1;gsb "SVC Y"
43: if bit(0,P)=1;gsb "SVC Z"
44: eir 7;iret
45: "SVC X":●●●;ret
46: "SVC Y":●●●;ret
47: "SVC Z":●●●;ret
48: end

```

By using this method, the calculator runs its main program (lines 12 through 39) while waiting for a device to interrupt. When a Require Service message is seen (bit 7 on the 98034A Interface), the calculator automatically branches to the service routine between program lines. The eir statement (line 44) is needed to re-enable interrupt on bit 7 (SRQ) after each pass through the service routine.

5-10 Interrupt Control

Remember that interrupts are not generated by specific devices on the bus, but only by the 98034A Interface itself. So, if more than one device on the bus is able to request service, the only interrupting condition is via the SRQ line. The service routine can determine which devices on the bus are currently requesting service, however, via a serial and/or parallel poll.

Also, it is not possible to establish two different service routines for the same interface: one for active talker and another for active listener, for example. Each oni statement cancels any previous oni statement for the same select code. If both of these conditions are set as interrupting conditions, the service routine must determine which condition caused the interrupt by using the rds function, and then test the appropriate talk/listen bits in the status byte returned.

For example, here is part of an interrupt method used when the calculator controls a subsystem of data measurement devices, and is also controlled by another device on the bus. Line 10 sets up a service routine and enables interrupt from the bus for any of three conditions: active talker, active listener, or active controller. When the calculator is not on the bus, it runs a data reduction and plotting program (lines 11 thru 49). When the active controller interrupts, service routine "BUS" determines which bus function has been addressed (talker, listener, or controller) and branches to an appropriate subroutine. The "TALK" subroutine sends data to the controller. The "LISTEN" subroutine inputs control information for the calculator. The "CONTROL" subroutine programs and takes data from the measurement devices on the bus; then the pass control (pct) statement returns active control to the other controller (address 26).

```
10: oni 7,"BUS";eir 7,112
    •
    •
    •
50: "BUS":rds(7)→A
51: if bit(4,A)=1;gsb "TALK"
52: if bit(5,A)=1;gsb "LISTEN"
53: if bit(6,A)=1;gsb "CONTROL"
54: iret
55: "TALK":wrt 731,A[1];ret
56: "LISTEN":red 731,A$•••;ret
57: "CONTROL":•••;pct 726;ret
58: end
```


Abortive Interrupts

The oni statement described earlier in this chapter allows for an optional abort byte parameter. Normally, interrupts using the end-of-line (EOL) branching scheme described earlier are sufficient, and interrupts are serviced in a reasonable amount of time. Some extraordinary circumstances, however, may require that a critical interrupt (e.g., a warning from a device of a critical or dangerous situation that must be corrected immediately) must be serviced in as short a time as possible. For these rare situations, the abort byte can be used.

NOTE

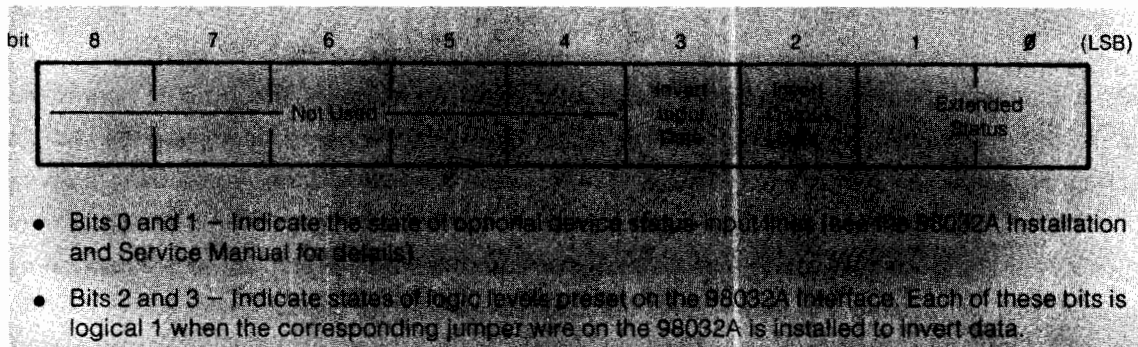
Abortive Interrupts should be used with extreme care.



If an interrupt has been declared abortive, this is detected as soon as the interrupt is received by the calculator and an immediate branch to the service routine is performed. The currently executing line of the main program is aborted, any other pending interrupts are cancelled, and an immediate branch to the service routine is performed, unless a record or load operation is being performed on the tape cartridge. The record or load operation is completed before an interrupt branch takes place. (Interrupts are not recognized during cartridge operations.) As far as the calculator is concerned, an abortive interrupt is nearly the same as pressing  followed by executing `cont "label"`, where "label" is the location of the interrupt service routine specified in the oni statement. This is a drastic action for extreme cases only, since variables that were being modified when the action occurred may be lost. Also, all pending gosubs and for/next loops are lost. The only meaningful action after an abortive interrupt is to perform any I/O operations necessary to quickly correct or halt the critical situation, followed by loading an entirely new program to bring the system back to an operational state.

The abort byte in the oni statement specifies a binary value of which only the four least-significant bits are used for the 98032A Interface. Bits 2 and 3 however, should not be used since they are preset on the interface card. This byte is logically ANDed with the lower four bits of the status byte to determine whether the interrupt is to be abortive. Thus, if any of the bits set in the abort byte are also set in the status byte at the time the interrupt occurs, the interrupt is to be abortive.

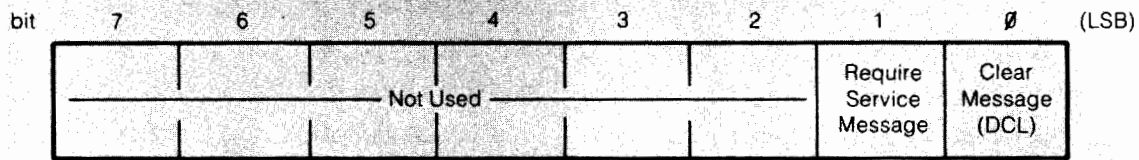
The 98032A status byte is described in chapter 3. Only the four least-significant bits are described here:



98032A Abort Byte

For example, if this oni statement is in effect: `oni 3, "overflow", 2` and the interrupt is enabled: `eir 3` an abortive interrupt will occur to the service routine labelled "overflow" when bit 1 (binary 2) of the status byte is logical 1. But, if this oni statement is used: `oni 3, "overflow", 3` the abortive interrupt occurs when either status bit 0 or 1 or both are logical 1.

The abort byte for the 98034A HP-IB Interface uses only the two least-significant bits. So, the range of a meaningful abort byte is from 1 thru 3. The 98034A abort byte is shown next.




- Bit 0: Execute autostart routine when a Clear message (DCL) is received.
- Bit 1: Abortive interrupt when a Require Service message is received.

98034A HP-IB Interface Abort Byte

If bit 1 is set, a Require Service message on the bus will cause an abortive interrupt. If bit 0 is set, the Clear message from the controller on the bus will initiate the autostart routine (see Chapter 3). Thus, when the calculator is acting as a peripheral on an HP-IB the program may either perform its own definition of the Clear message via a normal programmable interrupt, or it may use that message to cause power-on auto-restart by setting the abort byte. Remember that the 98034A Card is set to interrupt (normal or autostart) on the Clear message by setting bit 1 of the interrupt enable byte in the eir statement.

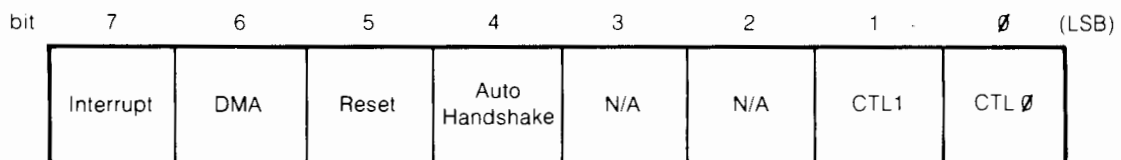
Normal EOL interrupts are serviced only when the program is running. Abortive interrupts and autostart on the Clear message are serviced anytime except during, or immediately after,

program editing (before run is executed). If an abortive interrupt is encountered while a program is being edited, pressing  may be required to return control to the keyboard. In all cases, abortive interrupts should be used with extreme care!

Interface Control Bits

The `eir` statement and the General I/O write control (`wtc`) statement both output to control register (R5) of an interface card. The differences in their actions is described next.

The write control statement provides a means of modifying bit 0, 1, and 5 of the control byte for the 98032A Interface. The format of this control byte is shown below.



98032 Interface Control Bits

Only bits 0, 1, and 5 are settable with the `wtc` statement (bits 2 and 3 are not used by the 98032A). Any of these bits (0 thru 7) may be set or cleared, however, by using the appropriate interrupt enable byte in the `eir` statement. When no byte is specified, bit 7 is automatically set to enable interrupt whenever the peripheral flag line is true. Bits 4 and 6 should not be set via an `eir` statement since, if they are set at the wrong time, they can disrupt normal Extended I/O operations.

You may wish to modify the settings of bits 0 thru 3 of the control byte. If these bits are set via the `wtc` statement, they remain set as specified until the next automatic write to the control register to service an interrupt. If these bits are set via an `eir` statement, however, the setting is saved and the state of bits 0 thru 3 is preserved whenever an output to the control register is required to service interrupts.

For example, executing this statement (in the octal mode): `eir 5,203` enables the interface at select code 5 for interrupt and sets control bits 0 and 1. When the device interrupts, the calculator automatically clears bit 7 to disable interrupt until the service routine is reached. The state of bits 0 thru 3 will be remembered and preserved. Also, `eir 5,3` could be used to set the two control bits without enabling interrupt. A later transfer (`tfr`) statement¹ would automatically enable (and disable when complete) interrupts while maintaining the setting of the four control bits. If these bits had been set via a `wtc`, however, their settings would not be maintained.

¹The transfer statement is used only with an I/O Buffer, as explained in Chapter 6.

Interrupt Lockouts

During certain critical operations within the calculator, all programmable and automatic interrupts may be disabled for short time periods. Usually, these lockout periods are only a few microseconds. An exception to this is during tape drive operations. While a find file (fdf) operation is in progress, for example, the DMA channel is in use and is not available for transfer operations. This simply means that a transfer (tfr) statement that is attempting to set up a DMA transfer will wait for the find file operation to be completed before being granted access to the DMA channel. Similarly, a fdf statement must wait for a tfr to be completed before it can use the DMA channel.

While a tape data transfer is in progress (e.g., load program, record program) the entire interrupt mechanism is turned off. So any devices attempting to interrupt during this time will not be logged in until the tape drive operation is completed. So you should exercise care in writing a program in which critical interrupts and tape drive operations are interleaved. Interrupts are also locked out for the duration of a Fast Read/Write data transfer (see Chapter 6) in order to provide the data transfer rate required. The tfr statement sets up the transfer operation, but the device determines when the transfer begins.

Normal interrupt operation is resumed after each of these interrupt lockout operations is finished.

Variables with Interrupt Service Routines

The programmer should remember when writing interrupt service routines that all variables in the 9825 are "global" variables except p-numbers (see the Advanced Programming chapter of the Operating and Programming Reference). This means that variables are recognized and modifiable in all segments of the program. So care should be taken to ensure that an interrupt service routine (which can be called at any point in the program) does not inadvertently modify program variables used by either the main program or a lower-level service routine.

Also, program modes should be carefully watched. If, for example, a line in the program were of the form:

```
par 3!urt 2,X,Y,Z!par 0
```

parity would be cancelled (par 0) before the calculator could branch to a service routine. If the par 0 statement were on the next line of the program, however, a service routine could interrupt while parity type 3 is still set, which could generate unexpected results within the service routine.

When interrupts are being used, care should be exercised to prevent modes such as par, conv, ctbl, or moct from being active when they are not needed. Similarly, executing format statements from within service routines should be done with care, since they may override formats previously set in the main program.

Notes

Chapter 6 Table of Contents



The Buffered I/O Scheme	6-3
Automatic Interrupt	6-4
Buffer Types	6-4
The Interrupt Buffer	6-5
The Fast Read/Write Buffer	6-5
The DMA Buffer	6-6
Buffer Underflow and Overflow	6-6
Buffer Statement (buf)	6-6
Transfer Statement (tfr)	6-8
Data Output	6-8
Data Input	6-9
I/O Buffer Status	6-10
Buffer Pointers	6-11
String Variables and Buffers	6-13
Inverted Data	6-15
Buffered I/O Example	6-16
Demonstration Programs	6-17
Buffered Benchmarks	6-21

Notes

Chapter 6

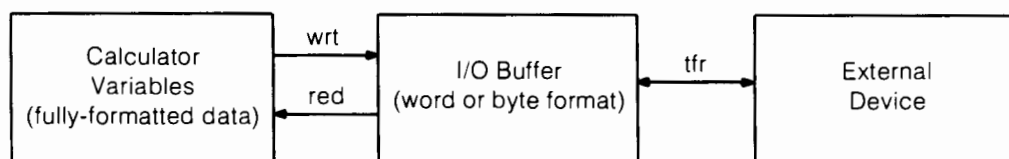
Buffered I/O

The Buffered I/O Scheme

For the majority of I/O operations, the speed of the calculator and the speed of a peripheral are reasonably matched, so the General I/O read and write statements will easily accomplish the data transfer. Very slow and very fast peripherals, however, create speed mismatches which can usually be overcome by using buffered I/O.

The buffered I/O scheme enables the calculator to automatically transfer data to or from external devices using various modes and data formats. Automatic interrupt control is enabled with each transfer operation.

The buffer (buf) statement allocates and names an area of read/write memory as an I/O data buffer. It also specifies whether the buffer is to use a 16-bit binary (word) or 8-bit ASCII (byte) format. The type of data transfer to be performed is also specified. Once the buffer is allocated, General I/O read- and write-type operations are used to exchange data between the buffer and calculator variables, while the transfer (tfr) statement is used to exchange data between the buffer and the external device. In effect, the buffer becomes the peripheral device for read and write operations. The next figure shows this I/O buffer scheme.



The I/O Buffer Scheme

Automatic Interrupt

As described in Chapter 5, programmable interrupts allow you to perform any sequence of operations to service a peripheral interrupt. If the task to be performed is simple data transfer between the calculator and a peripheral, however, the Extended I/O ROM provides an automatic mechanism for handling interrupts with an I/O buffer. This automatic interrupt is set up whenever a transfer statement is executed, and takes priority over programmable interrupts. For high speed data transfers, as explained later, the automatic interrupt even disables keyboard interrupts while the transfer is in progress.

Buffer Types

The buffer statement specifies which of these buffer types is to be set up: interrupt (type 0 or 1), fast read/write (type 2 or 3), or DMA (direct memory access, type 4). The even numbers indicate word (16 bit) format, while the odd numbers indicate byte (8 bit) format. The buffer type specified should match the speed and data format of the external device.

Some devices are extremely slow (such as a 110-baud teleprinter) or totally time-random (like an operator controlled digitizer). With General I/O ROM operations, the calculator simply waits on these devices to complete each I/O operation. If the time spent waiting is significant, and if the program could be performing other calculations while it is waiting for these devices, the interrupt buffer can be used to send or receive each item of data under interrupt while the calculator is performing other useful work.

On the other end of the speed spectrum are very fast devices (such as digital voltmeters and fast analog-to-digital converters) which deliver data at a rate faster than can be read using the read statement. Either a fast read/write or DMA buffer can be used to simply gather the data as fast as possible without spending the time to convert the data to the calculator's internal format (i.e., formatting, converting to floating-point representation, etc.). This work can all be done later, after the data has been input.

The following table summarizes the uses for each buffer type. Notice that I/O operations with medium speed devices (such as a 9866A/B or 9871A Printer, or most HP-IB modules) are not listed in the table, since General I/O read- and write-type operations provide an optimum data transfer rate for most cases. In applications where considerable time is spent on the data transfer, however, use of either the interrupt or DMA buffer may save execution time.

I/O Buffer Applications

Example Application	Buffer Type
Slow Devices:	
• 9863A Tape Reader	Interrupt
• 9869A Card Reader	Interrupt
• 9884A Tape Punch	Interrupt
Random-time Devices:	
• 9864A Digitizer	Interrupt
High-Speed Devices:	
• 9883A Tape Reader	Fast Read/Write or DMA ²
• HP-IB data input	Fast Read/Write ¹
• Burst Read from DVM	Fast Read/Write or DMA ²
Synchronous	DMA

¹The DMA buffer cannot be used with HP-IB; use a fast read/write buffer for the fastest transfer rate.

²For byte transfers, the fast read/write buffer offers the most efficient memory usage.

Use a fast read/write buffer when tape drive (or disk) operations are to be done during data transfer, since a DMA buffer, tape drive and disk require use of the same DMA channel.

The Interrupt Buffer

When a transfer statement using an interrupt buffer is executed, it automatically enables the device to interrupt each time it is ready to output or input another word or byte of data. Then the calculator goes on executing the program statements and lines following the transfer statement. Each time the peripheral is ready, it generates an interrupt, transfers the next word or byte of data, and goes busy again. In the meantime, program execution continues normally, interrupting only long enough to transfer the next data character. This operation continues until the last data character has been transferred, at which time the calculator completes the transfer and disables the peripheral from further interrupts. Note that the entire transfer operation is automatically handled by the calculator and no interrupt service routine is required in the program. Also, each new data request by the peripheral is serviced when received and not at the end of the current line of the user program. End of line (EOL) branching is used only with programmable interrupts, as explained in Chapter 5.

The Fast Read/Write Buffer

Data transfer with a fast read/write buffer is similar to using an interrupt buffer, except that once the data transfer has begun, all interrupts are disabled until the last data item is transferred. None of the main program is executed for the duration of this data exchange. When the transfer is complete, interrupts for other select codes are re-enabled, and the main program continues execution from where it was interrupted. A fast read/write transfer begins when the device interrupts, and continues in a fast I/O exchange until completed.

The DMA Buffer

Using a DMA buffer can achieve even faster data transfer rates through the use of direct memory access. In this mode, data is exchanged between the buffer and the peripheral directly by the calculator processor and independent of the ROM software routines. The DMA transfer occurs on a "cycle stealing" basis, without any disruption of normal program flow. Only the 98032A Interface is capable of running in the DMA mode. For the HP-IB, the Fast Read/Write buffer scheme affords the fastest transfer rate.

All of these transfer operations are completely automatic. All you need do, for say an output operation, is set up the buffer area (buf statement), fill it with data (wrt or wtb statement), and initiate the transfer to the peripheral (tfr statement). The automatic interrupt service and buffer management is taken care of by the calculator.

Buffer Underflow and Overflow

The I/O buffer may be written into and read from using any sequence of read, write, and transfer (tfr) operations, provided that the buffer operation does not cause underflow or overflow (i.e., attempting to read from an empty buffer or write to a full buffer). If a buffer is only partially filled and then emptied, more data may be written into the buffer without erasing the information left in the buffer from the previous write operation. The data is not repacked within the buffer area, however, and any unused space is left in the low end of the buffer. Thus, buffer overflow error E5 may occur even when the buffer contains fewer characters than the size originally specified. When the buffer is emptied (the last character has been output) the buffer may be filled completely again. So partial reads should be done with care. Buffer pointers are explained later.

Buffer Statement

```
buf "name" [ # buffer size or string variable # buffer type]
```

The buffer statement is similar to the dimension (dim) statement in that it allocates and names an area of read/write memory. As with the dim statement, once a buffer has been allocated it cannot be modified (i.e., the name, size, and type cannot be changed) or de-allocated. The buffer can be cleared, however, by executing the syntax:

```
buf "name "
```

The purpose of the buffer for output operations is to prepare and hold data to be transferred to a peripheral by one of the automatic transfer operations described in this chapter. For input operations, it provides a means of buffering data received from a peripheral at its own rate, and reading this data into calculator variables when the program is ready to receive them.

The buffer name can be any string of characters in quotes or a string variable name. The name is then used in place of the select code parameter in I/O operations with the buffer. If a string variable name is used, string operations can be performed on the string buffer.

The buffer size specifies how large an area of memory is to be allocated. The size is specified in either words or bytes, depending upon the buffer type specified. In addition to the specified size, each buffer uses an additional 16 bytes of read/write memory as working storage (overhead). Also string variables can be assigned as buffers, as described later.

The buffer type is a number from 0 thru 4 which specifies one of these types:

I/O Buffer Types

Type	Buffer Type	Data Format
0	Interrupt buffer	words
1	Interrupt buffer	bytes
2	Fast read/write buffer	words
3	Fast read/write buffer	bytes
4	DMA buffer	words

The buffer type specifies whether the buffer is to hold bytes (8-bit characters) or words (16-bit binary data). It also specifies the mode of operation for transfer of data to or from a peripheral device. These buffer types were described earlier.

Once the buffer has been established, General I/O read- and write-type operations are used to exchange data between the buffer and the calculator's internal variables. This is done by simply using the buffer name in place of the select code parameter in read- and write-type statements and functions. The same data that would normally be sent to the peripheral (for write operations) is sent to the specified buffer instead. Within this buffer, the data exists as a simple byte or word sequence, and the General I/O formatting capability may be used to write data to the buffer. Similarly, the byte or word data sequence can be read from the buffer into internal variables, under format control if desired. To specify a format statement in read- and write-type operations, the "name" parameter has the following form:

" name _ format no. "

Since buffer names and device names (see "The Device Statement" in Chapter 2) may be used in place of the select code parameter in read- and write-type operations; a buffer and a device cannot be given the same name. If a buffer statement is executed and the specified name has already been used as either a device name or another buffer name, error E2 will result.

Transfer Statement

```
tfr source, destination[, character count[, last character]]
```

source = buffer "name" or select code

destination = buffer "name" or select code

As mentioned in the previous section, General I/O operations are used to put data into a buffer from calculator variables, or take data from the buffer and put it into calculator variables. The transfer statement is used to exchange data between the buffer and a peripheral device. If the source is a buffer, the destination must be a select code or device name, and vice versa.

Data Output

To transfer data from the buffer to the peripheral, the source parameter is the name of the buffer and the destination parameter is the select code or device name of the peripheral to receive the data. The mode of transfer is determined by the buffer type.

The character-count parameter can be used to terminate the output transfer when the specified number of bytes or words are output. When this parameter is not given, the transfer is terminated after the buffer is emptied. The last-character parameter is ignored during an output transfer.

For example, this program sequence sets up a 300-character interrupt buffer for holding sets of variables to be printed on a teleprinter. Lines 6 thru 20 calculate each set of variables and then write them into the buffer. The transfer statement sets up the automatic output routine between the buffer and the printer on select code 3. After enabling the printer to interrupt when it is ready for each successive character, program execution resumes with the next statement.

```
5: buf "out",300,1
6: for I=1 to 100
  .
  .
  .
20: wtb "out",A,B,C
21: next I
22: tfr "out",3
23: end
```

Notice that the tfr statement is executed only once to set up the automatic transfer operation. In this sequence, the transfer operation is in effect until either the buffer is emptied (underflow) or overfilled via the write statement (overflow). Error E5 indicates underflow or overflow. Since the buffer is large enough to hold many sets of variables, overflow may not occur if the printer is fast enough to keep up with program execution. If the printer is too fast, the buffer will empty and the transfer will have to be re-initialized after new data is written into the buffer.

To avoid error E5 the program can be written to detect the current buffer size, and branch to wait until the buffer is emptied before doing the next write statement (avoid overflow) or to re-execute the tfr statement if the buffer has been emptied already (avoid underflow). See "Buffer Status" later in this chapter for details.

Data Input

For transfer operations into an I/O buffer, the source is specified as a select code or a device name and the destination is specified as a buffer name. Upon execution of the transfer statement, data is taken from the peripheral and placed in the buffer according to the buffer type specified. When the transfer is complete, the data is then taken from the buffer using General I/O read operations, with the buffer name in place of the select code, and using formatting if desired.

During the transfer from the peripheral to the buffer, the calculator must have some way of knowing when the operation is complete, that is, when the last word or byte has been received. You can specify this cutoff condition in the transfer statement through the optional character-count and last-character parameters. The character count is the number of words or bytes to be read in order to complete the transfer operation. If this value is larger than the space available in the buffer, the input transfer is terminated when the buffer is filled.

The last-character parameter is used by byte-type buffers only to terminate when the specified character is input. For example, when decimal 10 (or octal 12) is specified, the input transfer will terminate after an ASCII line feed has been input. If a last character is given, the number of characters must also be specified, although it may be given as zero to indicate that only the terminating character or filling the buffer is to act as the cutoff condition. For example, in this sequence:

```
1: mdec;buf "hold",750,3
2: tfr 3,"hold",500,10
```

data is transferred from the device on select code 3 to the buffer "hold", until either 500 characters are read or an ASCII line feed is seen.

When using the transfer statement with the HP-IB to input into a buffer, the transfer can be terminated as described above or by End Or Identify (EOI). Refer to The HP Interface Bus in chapter 2.

I/O Buffer Status

Since data transfers using a buffer are done automatically while the main program is running, a method is needed for the program to detect when the buffer transfer is finished. There are two methods available for doing this, one uses the read status (rds) function and the other uses a programmable interrupt service routine.

The program can check current buffer status by executing this read status function:

```
rds ( "buffer name " )
```

The function returns -1 whenever a transfer statement is active with the buffer. When the buffer is not busy, the number of words or bytes currently available for output from the buffer is returned as its status. Thus, a buffer that has finished a transfer to a device will show a status of zero and a buffer that has finished a transfer from a device will show a status equal to the character-count parameter (plus any data that was left in the buffer from previous operations).

The second method of detecting the completion of a transfer operation makes use of the programmable interrupt scheme. When a transfer operation has just been completed, and an "oni" location has been previously set up for the same select code, a normal end-of-line service request is logged in. The program then branches to the service routine according to the programmable interrupt scheme explained in Chapter 5.

For example, this sequence specifies that 50 characters (bytes) should be transferred from the device on select code 2, and placed in the buffer called "data". When the transfer is complete, an interrupt is logged in to branch to the service routine labelled "done". Notice that an enable interrupt (eir) statement is not needed (or should not be used!) to enable an interrupt from the same select code; it's done automatically by the transfer operation.

```
0: oni 2,"done"
1: buf "data",50,1
2: tfr 2,"data",50
```

NOTE

If an eir and a tfr statement are in effect for the same select code, the service routine will probably be executed before the transfer operation.

As another example, suppose that we have a calculator-digitizer-printer system and wish to digitize, compute, and print data as fast as possible. The digitizer is connected via a 98032A Interface set to select code 3. Data points are randomly input, since the operator must manually move the digitizer cursor from point to point.

By using the following method, the calculator is free to compute and print data (lines 8 thru 24) while the operator digitizes each new data point. The transfer statement automatically inputs one data point (a 15-character sequence) and then logs in an interrupt causing the calculator to branch to service routine "read". The service routine then empties the current data from the buffer, counts data points, and returns control to the main program. If the main program sequence is finished before the current transfer operation is complete, the calculator displays `Digitize Next Point` and waits (executes line 25 continually) until the buffer has been filled and emptied.

Remember that buffer status is -1 when a transfer is active, and 0 when the buffer is empty.

```

5: oni 3,"read"
6: buf "digitize",15,1
7: tfr 3,"digitize"
  ●
  ●
  ●
25: dsp "Digitize Next Point";jmp rds("digitize")=0
26: gto 7
27: "read":red "digitize",X,Y
28: I+1→I;iret
29: end

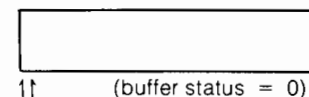
```

Buffer Pointers

Each I/O buffer has two internal pointers which indicate the last word or byte currently input and output. The following diagrams show the position of these pointers after various operations using a 20-byte output buffer. A `↑` indicates an input pointer and a `↓` indicates an output pointer. The read status function `rds ("A")` was executed after each operation to determine the current buffer status.

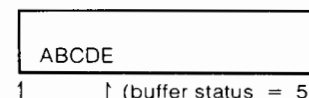
1. Set up a 20-byte buffer:

```
buf "A",20,1
```



2. Write five characters into buffer:

```
wtb "A", "ABCDE"
```

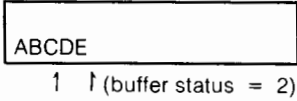


6-12 Buffered I/O

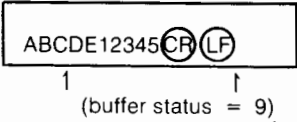
3. Transfer three characters out:

```
tfr "A", 6, 3
```

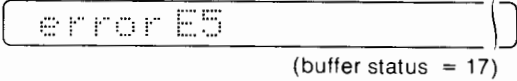
Note that ABC still remains in the buffer.


4. Write more characters in:

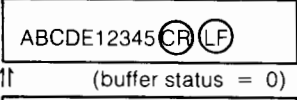
```
wrt "A", "12345"
```


5. Attempt to write too many more characters into buffer: 11 characters

```
wtb "A", "XXXXXXXXXXXXX"
```

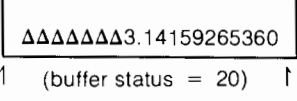

6. Transfer remainder of buffer out:

```
tfr "A", 6
```

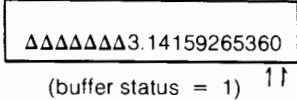

7. Now the buffer can be filled with new data: (Δ = a space):

```
fwt f20.11, z
```

```
wrt "A", f
```

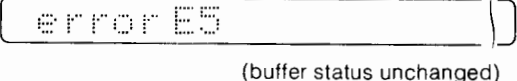

8. Transfer 19 bytes out:

```
tfr "A", 2, 19
```


9. Attempt to write in one byte:

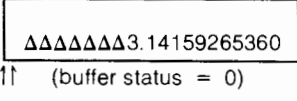
```
wtb "A", "X"
```

Gives error E5 no room to store it.


10. Remove the last byte:

```
tfr "A", 6
```

The buffer can now be refilled.



Notice in each step, that buffer status indicates the number of bytes available for output (the number of bytes between pointers) and not necessarily the total number of bytes in the buffer. As shown by steps 5 and 9, the buffer cannot be refilled after being only partially emptied – the buffer must be emptied completely before it can be filled again. Also remember that data which has been output from the buffer can not be output again, even though it is still in the buffer.

When unwanted data remains in the buffer, as after step 8, it can be removed by executing the `buf` statement with the buffer name as shown.

```
buf "A"
```

In the case where a string variable is used as the buffer (refer to the next section), it can be seen (by printing the string) that the contents of the buffer are not changed, only the input and output pointers are reset (`buffer status = 50`) by the `buf` statement.

String Variables as Buffers

The size parameter in the buffer statement may be replaced by the name of simple string variable. Substrings and strings of a string array are not allowed. This permits the string variable to also serve as an I/O buffer.

For example, these program lines dimension a 100-character string variable and then assign the string as an interrupt type buffer called "fer":

```
0: dim A$(100)
1: buf "fer",A$,1
```

When a string is specified as a buffer, 16 characters are assigned as working storage (overhead). So a string dimensioned at 100 characters will allow a buffer size of only 84 bytes (42 words). If a 100-byte buffer is required, the string should be dimensioned at 116 characters. Remember that non-string buffers automatically allocate the extra area, so the size specified is the size of the buffer. For byte-type buffers, odd buffer sizes are rounded to the next higher even number.

Using string variables as buffers offers additional features. For example, the buffer may be saved on the tape cartridge or a disk for processing at a later time. This allows one segment of a program to be a data acquisition phase and simply fill a buffer from a device, record the buffer, reset the buffer to empty, and continue gathering data. In a later phase the buffers may be loaded from the tape or disk and processed. A second advantage of string buffers is that the string-manipulation functions may be used to "preview" the form of the data received before it's read into computer variables. Or data may be "pre-conditioned" to suit a particular format or data structure. Be aware, however, that these string variable operations are entirely independent of the normal red/wrt/tfr operations.

As an example, suppose that 10 characters (bytes) are transferred into a buffer which uses the area for A\$. Then the statement "ABC"→A\$ is executed. Now executing len(A\$) (to determine the length of the string) will return 3. But the buffer size will still be 10 bytes even though the first three bytes have been changed to ABC and the length of A\$ is 3. Bytes 4 through 10 are unchanged.

It is possible to transfer data into a string buffer and then store the string on disk. However, when the string is retrieved the buffer pointers are not restored, making it impossible to read data from the buffer using red's and rdb's. Although it is possible to use string functions (i.e. "val" and "num") to retrieve the data, this approach will not be as fast. (See benchmarks.)

6-14 Buffered I/O

One way to program around this problem is to store the status of the buffer (i.e., its length) with the string. Then, when the data is to be retrieved, load the string and the buffer status from the disk. The buffer pointers can be restored by writing the string to the buffer.

Here's one way to save the data:

```
dim B$(36)
buf = Buffer, B$, 1
code to fill the buffer
read "Buffer", 1, L
write F, B$, L
end
```

Dimension string 16 characters longer than buffer. Any type is OK.

Save the number of characters in the buffer.

Write the string data and buffer status to disk (a random print could also be used).

Here's how to retrieve the data and restore the buffer.

```
dim B$(36)
buf = "New", B$, 1
sread F, B$, L
wtb "New", B$, 1, L
code to retrieve data from the buffer
end
```

Dimension string 16 characters longer than the buffer. Again, any type will work.

Recall the data from disk. Of course random reads could also be used.

This statement resets the buffer pointer; it works for any length buffer.

If you can guarantee that your buffer will be completely filled, then you do not need to store the status of the buffer on disk and the buffer pointers can be stored with: `wtb "New", B$`

Inverted Data

The 98032A Interface has two jumper wires which may be set to specify inverted (positive true) logic levels for input and/or output data. The card is normally set to handle negative-true logic. During normal read and write operations, the state of these jumpers is checked by the calculator and the data is inverted, if necessary, before writing and after reading. This is also done for data transfers using an interrupt buffer. The two fast-access buffer transfers (fast read/write and DMA), however, do not check these inversion jumpers to maintain maximum transfer rates. So the program must compensate for inverted data when fast read/write or DMA buffers are used with an interface set for inverted logic. The fact that inverted data is received in these modes of transfer should also be considered when specifying the last-character parameter of an input transfer statement.

Currently, the only HP calculator peripheral that uses inverted logic levels is the 9864A Digitizer. Since this is a slow (time random) device, only the interrupt buffer should be used for transfer operations; this also avoids the change for inverted data.



Buffered I/O Example

The following program uses a fast read/write buffer to enter and print measurements from a 5328A Frequency Counter on the HP-IB.

```

0: dim F$(20,17),G$(356);mdec
1: wrt 710,"PF4G2S!T"
2: " Type 3":buf "CBuf",G$,3
3: tfr 710,"CBuf",340
4: rds("CBuf")>B;if B=-1;jmp 0
5: fxd 0;prt "#Bytes=",B;spc ;prt "CBuf=",G$;spc ;prt "Buffer="
6: for J=1 to 20
7: conv 69,101;red "CBuf",F$(J);prt F$(J)
8: next J
9: spc 2
10: "Refmt & Prt":prt "Frequency=";for K=1 to 20
11: fmt 1,f6.2," MHz";wrt 16.1,val(F$(K))/1e6
12: next K
13: spc 2;end

```

- 0: Dimensions string array F\$ to hold 20 frequency readings each 17 characters long, and string G\$ to hold 340 characters of raw data plus 16 extra characters required for housekeeping purposes.
- 1: Programs 5328A Frequency Counter to take multiple measurements and output at end of each measurement.
- 2,3: Sets 9825A for Fast Read/Write (Type 3) buffer. A total of 340 characters are to be accepted.
- 4: Tests status; while buffer is being filled, the status is "-1" indicating "busy"; upon completion, it returns the final character count.
- 5: Prints final character count and raw data. Each reading is 17 character spaces wide including blanks as fillers.
Note that "→" is carriage return and "↓" is line feed. Such raw data printouts are useful for debug purposes.
- 6,8: Since each frequency reading is terminated by the line feed delimiter, a convenient way to separate the raw data string into individual readings is to read it into a string array. At the same time, the exponent prefix is converted to lower-case "e". The string array is printed to illustrate the operation.
- 10,13: The val function transforms the strings of ASCII representations into numeric values so they can be scaled (divided by 10⁶) and printed.

Printout:

#Bytes=	340	Buffer=		Frequency=	
CBuf? =		+	10.870e+6	10.87 MHz	
+ 10.870E+6+		+	10.880e+6	10.88 MHz	
↓ + 10.880E+6		+	10.880e+6	10.88 MHz	
+4 + 10.880E+		+	10.890e+6	10.89 MHz	
6+4 + 10.890E		+	10.890e+6	10.89 MHz	
+6+4 + 10.890		+	10.900e+6	10.90 MHz	
E+6+4 + 10.90		+	10.900e+6	10.90 MHz	
0E+6+4 + 10.9		+	10.910e+6	10.91 MHz	
910E+6+4 + 10.9		+	10.910e+6	10.91 MHz	
00E+6+4 + 10.		+	10.920e+6	10.92 MHz	
910E+6+4 + 10		+	10.920e+6	10.92 MHz	
.910E+6+4 + 1		+	10.920e+6	10.92 MHz	
0.920E+6+4 +		+	10.930e+6	10.93 MHz	
10.920E+6+4 +		+	10.930e+6	10.93 MHz	
10.920E+6+4 +		+	10.940e+6	10.94 MHz	
10.930E+6+4 +		+	10.940e+6	10.94 MHz	
10.930E+6+4 +		+	10.950e+6	10.95 MHz	
10.940E+6+4		+	10.950e+6	10.95 MHz	
+ 10.940E+6+4		+	10.960e+6	10.96 MHz	
+ 10.950E+6+		+	10.960e+6	10.96 MHz	
↓ + 10.950E+6					
+↓ + 10.960E+					
6+4 + 10.960E					
+6+4					

Demonstration Programs

Buffered I/O allows more efficient use of the 9825 Computer as shown in the following programs. The time between samples was increased by use of the 5328A Frequency Counter's sample rate control in order to show that it is possible to do useful work interleaved with data taking where time between samples permits.

The Test Case was run with the sample rate control set fully counter-clockwise to have the counter take readings with the minimum spacing between each one. So little time was left that use of a Type 1 "Interrupt" Buffer was of no avail.

For the two data runs, the time between readings was increased an arbitrary amount by setting the 5328A sample rate control to 1 o'clock. In program line 1, note the inclusion of the code "S7", which permits manual setting of this control.

Program 1 was run without interleaving any computations. Note that lines 5 and 6 test status in a tight loop from which the program exits when the buffer is full.

Program 2 was run with interleaved computations. Note that program line 6 terminates with "jmp -1" to update the index and perform the computation before again testing status to see whether the buffer is full yet.

The results show that more than 1000 computations can be made without taking but 13 milliseconds longer than in the case where the 9825 does no useful work between input samples.

Buffered I/O is a significant contribution to efficient utilization of system resources where the measurement situation is such that samples are spaced in time and there is other useful work the system can perform while the data buffer is being filled.

Test Program – Minimum Time Between Samples:

```

0: dim F$(20,17),G$(356);mdec;0→Y
1: wrt 710,"PF4G3S17T";red 710,A
2: wrt 716,"001E3PR";red 716,C;dsp "Cal=",C;spc
3: " Type 1":buf "CBuf",G$,1
4: wrt 716,"R";tfr 710,"CBuf",340
5: Y+1→Y;Y*ln(Y)→Z
6: if rds("CBuf")=-1;jmp -1
7: red 716,D
8: fxd 0;prt "Time,ms=",D-C;spc ;prt "Work=",Y;spc
9: for J=1 to 20
10: conv 69,101;red "CBuf",F$(J)
11: next J
12: "Refmt & Prt":prt "Frequency=";for K=1 to 20
13: fmt 1,f10.3," MHz";wrt 16.1,val(F$(K))/1e6
14: next K
15: spc 2;end

```



Program 1 – No Interleaved Computations:

```

0: dim F$(20,17),G$(356);mdec;0+Y
1: wrt 710,"PF4G3S17T";red 710,A
2: wrt 716,"001E3PR";red 716,C;dsp "Cal=",C;spc
3: " Type 1";buf "CBuf",G$,1
4: wrt 716,"R";tfr 710,"CBuf",340
5: 0+Y
6: if rds("CBuf")=-1;jmp -1
7: red 716,D
8: fxd 0;prt "Time,ms=",D-C;spc ;prt "Work=",Y;spc
9: for J=1 to 20
10: conv 69,101;red "CBuf",F$(J)
11: next J
12: "Refmt & Prt";prt "frequency=";for K=1 to 20
13: fmt 1,f10.3," MHz";wrt 16.1,val(F$(K))/1e6
14: next K
15: spc 2;end

```

Program 2 – Interleaved Computations:

```

0: dim F$(20,17),G$(356);mdec;0+Y
1: wrt 710,"PF4G3S17T";red 710,A
2: wrt 716,"001E3PR";red 716,C;dsp "Cal=",C;spc
3: " Type 1";buf "CBuf",G$,1
4: wrt 716,"R";tfr 710,"CBuf",340
5: Y+1+Y;Y*ln(Y)+2
6: if rds("CBuf")=-1;jmp -1
7: red 716,D
8: fxd 0;prt "Time,ms=",D-C;spc ;prt "Work=",Y;spc
9: for J=1 to 20
10: conv 69,101;red "CBuf",F$(J)
11: next J
12: "Refmt & Prt";prt "frequency=";for K=1 to 20
13: fmt 1,f10.3," MHz";wrt 16.1,val(F$(K))/1e6
14: next K
15: spc 2;end

```


Buffered I/O Benchmarks

The table below summarizes results of eight benchmark programs run to measure relative speeds of three methods of transferring data into the 9825 while measuring instruments on the HP-IB: Fast Read/Write (type 1), Interrupt Buffer (type 3), and ordinary reads using the red statement.

Buffered I/O Benchmark Times

Average Time/Reading in ms

Frequency Counter	Type 1 "Interrupt"	Type 3 "Fast Read/Write"	Standard Read (for/next loop)	Standard Read Dump into String
5345A	3.79 ms	1.75 ms	4.07 ms	2.75 ms
5328A	5.65 ms	3.00 ms	5.80 ms	4.40 ms

Results show the fast read/write capability to be effective in reducing the time required per reading. This could be significant where data runs are long or where data points must be taken as close together as possible.

The following programs were used to generate the table above.

1. 9825/5345A Counter with buffer type 1:

```

0: dim FS[28,17],GS[353];mdec
1: wrt 710,"12E8:<G<I1";red 710,A
2: wrt 716,"001E3PR";red 716,C;dsp "Cal=",C;spc
3: " Type 1":buf "CBuf",GS,1
4: wrt 716,"R";tfr 710,"CBuf",337
5: if row("CBuf")=-1;jmp 0
6: red 716,D
7: fxd 0;pct "Time,ms=",D;fxd 2;pct "Avg Time/row,ms=",D/28;spc
8: for J=1 to 28
9: conv 69,101;red "CBuf",FS[J]
10: next J
11: "Buf&Prt":prt "Frequency=";for K=1 to 28
12: fnt 1,F10.3," Hz";wrt 16.1,val(FS[K])/1e6
13: next K
14: spc 2;end

```

2. 9825/5345A Counter with buffer type 3:

```

0: dim F$(28,17),G$(353);mdec
1: wrt 710,"I2E8:<G<I1";red 710,A
2: wrt 716,"001E3PR";red 716,C;dsp "Cal=",C;spc
3: " Type 3":buf "CBuf",G$,3
4: wrt 716,"R";tfr 710,"CBuf",337
5: if rds("CBuf")=-1;jmp 0
6: red 716,D
7: fxd 0;prt "Time,ms=",D;fxd 2;prt "Avg Time/rdg,ms=",(D-C)/28;spc
8: for J=1 to 28
9: conv 69,101;red "CBuf",F$(J)
10: next J
11: "Refmt & Prt":prt "Frequency=";for K=1 to 28
12: fmt 1,f10.3," MHz";wrt 16.1,val(F$(K))/1e6
13: next K
14: spc 2;end

```

3. 9825/5345A Counter with read and for... next loop:

```

0: dim F$(28,17);mdec
1: wrt 710,"I2E8:<G<I1";red 710,A
2: wrt 716,"001E3PR";red 716,C;dsp "Cal=",C;spc
3: conv 69,101;wrt 716,"R"
4: for J=1 to 28;red 710,F$(J);next J
5: red 716,D
6: prt "Time,ms=",D;prt "Avg Time/rdg,ms=",(D-C)/28;spc
7: "Refmt & Prt":prt "Frequency="
8: for K=1 to 28
9: fmt 1,f10.3," MHz";wrt 16.1,val(F$(K))/1e6
10: next K
11: spc 2;end

```

4. 9825/5345A Counter with read into a string:

```

0: dim G$(353),A(100);mdec
1: wrt 710,"I2E8:<G<I1";red 710,A
2: wrt 716,"001E3PR";red 716,C;dsp "Cal=",C;spc
3: " Type=Do-It-Yourself":
4: wrt 716,"R";conv 69,101;fmt 2,c337,z;red 710.2,G$
5: red 716,D
6: prt "G$=",G$;spc
7: fxd 0;prt "Time,ms=",D;fxd 2;prt "Avg Time/rdg,ms=",(D-C)/28;spc
8: fxd 0;prt "Frequency="
9: for J=1 to 28
10: val(G$(P))=A(J)
11: pos(G$(P+1),chr(10))+P+1=P
12: fmt 1,f10.3," MHz";A(J)/1e6+A(J);wrt 16.1,A(J)
13: next J
14: spc 2;conv
15: end

```

5. 9825/5328A Counter with buffer type 1:

```

0: dim F$(20,17),G$(356);mdec
1: wrt 710,"PF4G3S1T";red 710,A
2: wrt 716,"001E3PR";red 716,C;dsp "Cal=",C;spc
3: " Type 1":buf "CBuf",G$,1
4: wrt 716,"R";tfr 710,"CBuf",340
5: if rds("CBuf")=-1;jmp 0
6: red 716,D
7: fxd 0;prt "Time,ms=",D;fxd 2;prt "Avg Time/rdg,ms=",(D-C)/20;spc
8: for J=1 to 20
9: conv 69,101;red "CBuf",F$(J)
10: next J
11: "Refmt & Prt":prt "Frequency=";for K=1 to 20
12: fmt 1,f10.3," MHz";wrt 16.1,val(F$(K))/1e6
13: next K
14: spc 2;end

```



6. 9825/5328A Counter with buffer type 3:

```

0: dim F$(20,17),G$(356);mdec
1: wrt 710,"PF4G3S1T";red 710,A
2: wrt 716,"001E3PR";red 716,C;dsp "Cal=",C;spc
3: " Type 3":buf "CBuf",G$,3
4: wrt 716,"R";tfr 710,"CBuf",340
5: if rds("CBuf")=-1;jmp 0
6: red 716,D
7: fxd 0;prt "Time,ms=",D;fxd 2;prt "Avg Time/rdg,ms=",(D-C)/20;spc
8: for J=1 to 20
9: conv 69,101;red "CBuf",F$(J)
10: next J
11: "Refmt & Prt":prt "Frequency=";for K=1 to 20
12: fmt 1,f10.3," MHz";wrt 16.1,val(F$(K))/1e6
13: next K
14: spc 2;end

```

7. 9825/5328A Counter with read and for... next loop:

```

0: dim F$(20,17);mdec
1: wrt 710,"PF4G3S1T";red 710,A
2: wrt 716,"001E3PR";red 716,C;dsp "Cal=",C;spc
3: conv 69,101;wrt 716,"R"
4: for J=1 to 20;red 710,F$(J);next J
5: red 716,D
6: prt "Time,ms=",D;prt "Avg Time/rdg,ms=",(D-C)/20;spc
7: "Refmt & Prt":prt "Frequency="
8: for K=1 to 20
9: fmt 1,f10.3," MHz";wrt 16.1,val(F$(K))/1e6
10: next K
11: spc 2;end

```


8. 9825/5328A Counter with read into a string:

```
0: dim G$(340),A(100);mdec
1: wrt 710,"PF4G3S1T";red 710,A
2: wrt 716,"001E3PR";red 716,C;dsp "Cal=",C;spc
3: " Type=Do-It-Yourself":
4: wrt 716,"R";conv 69,101;fmt 2,c340,z;red 710.2,G$
5: red 716,D
6: prt "G$=",G$;spc
7: fxd 0;prt "Time,ms=",D;fxd 2;prt "Avg Time/rdg,ms=", (D-C)/20;spc
8: l←P;prt "Frequency"
9: for J=1 to 20
10: val(G$(P))→A(J)
11: pos(G$(P+1),char(10))+P+1→P
12: fmt 1,f10.3," MHz";A(J)/1e6→A(J);wrt 16.1,A(J)
13: next J
14: spc 2;conv
15: end
```

Chapter 7

Table of Contents



Summary of Plotter ROM Syntax	7-2
Introduction	7-3
Programming Requirements	7-4
Plotter Select Code Statement (psc)	7-5
Plotting	7-7
Scale Statement (scl)	7-7
Plotter Clear Statement (pclr)	7-10
9872 Axis Statements (xax, yax)	7-11
9862 Axis Statement (axe)	7-18
Pen Statement (pen)	7-22
Plot Statement (plt)	7-22
Pen Select Statement (pen#)	7-25
Offset Statement (ofs)	7-27
Incremental Plot Statement (iplt)	7-29
Line Type Statement (line)	7-32
Limit Statement (lim)	7-34
Lettering	7-36
Label Statement (lbl)	7-36
9872 Charater Sets	7-37
Character Size Statement (csiz)	7-38
Spacing Between Characters	7-40
Character Plot Statement (cplt)	7-41
Plotter Typewriter Statement (ptyp)	7-45
Letter Statement (ltr)	7-47
Digitizing	7-48
Digitizing Statement (dig)	7-48
Additional HP-GL Plotter Control	7-50
9872 Plotter Default Conditions	7-51
Plotter ROM Comparison	7-52
Axe and Ltr Conversions	7-54

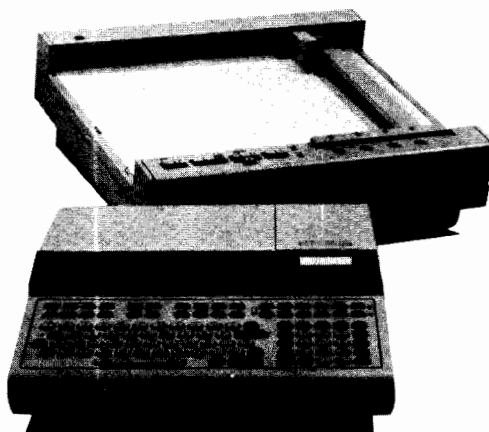
Summary of Plotter ROM Syntax

<code>ssc</code>	Set the select code (and the HP-IB device address) of the plotter to be controlled.
<code>scl</code>	Specify the origin and set the scale units for successive plotting statements.
<code>axe</code>	Draw X and Y axes on the 9862A Plotter.
<code>xax</code> <code>yax</code>	Draw X and Y axes, and optionally label the axes, with the 9872 Plotter ROM.
<code>pcldr</code>	Reset the 9872 Plotter (or other HP-GL device) to its default state. 9872 ROM only.
<code>pen</code>	Raise the pen.
<code>plt</code>	Draw a line to a specified X, Y position.
<code>pen#</code>	Select a plotter pen. 9872 ROM only.
<code>ofs</code>	Move the plotting origin (point 0,0) to a new specified X, Y point.
<code>iplt</code>	Move the pen by an incremental distance, both horizontally and vertically, to a specified point.
<code>line</code>	Select the type of line (dashes, dots, etc.) drawn with <code>plt</code> , <code>iplt</code> , <code>xax</code> , and <code>yax</code> . 9872 ROM only.
<code>lim</code>	Restricts pen motion to a specified area on the platen. 9872 ROM only.
<code>lbl</code>	Print text (labels) on the plotter.
<code>csiz</code>	Specify the size and shape of characters printed on the plotter.
<code>cplt</code>	Move the pen a specified number of character spaces, both horizontally and vertically.
<code>ptyp</code>	Sets a manual lettering mode, allowing the operator or the program to label the plot.
<code>ltr</code>	Set character size and plotting direction. 9862 ROM only.
<code>dis</code>	Allows the device to transmit its current position to the computer. 9872 ROM only.

Chapter 7

Plotter Control

The HP 9862A and 9872A Plotter ROMs provide a common set of statements for controlling a wide variety of plotting devices. Although the 9862 Plotter ROM is designed to control only the HP 9862A Plotter, the 9872 ROM controls plotting devices by transmitting HP-GL (Graphic Language) commands via the HP-IB. This allows you to control the 9872A, 7225 and many other devices which respond to a standard set of HP-GL commands. Each ROM has statements for scaling and plotting, drawing axes, and printing text on the plotter. The 9872 ROM offers many additional statements, as listed on the facing page.



HP 9825A Desktop Computer and 9872A Plotter

The 9862A Plotter ROM uses 70 bytes of read/write memory when installed in an HP 9825A Desktop Computer. The 9872 Plotter ROM uses 104 bytes of 9825A read/write memory. Both plotter ROMs cannot operate in the computer at the same time. Both ROMs are permanently installed in the 9825B Desktop Computer.

NOTE

The 9825B is shipped from the factory set to access the 9872 ROM. To access the 9862 ROM, call your HP Customer Engineer for assistance.

Programming Requirements

Before continuing in this chapter, you should be familiar with programming in HPL. See the Operating & Programming Reference. When controlling an HP-GL plotter via the 9872 ROM, you should also refer to the HP-GL Programming Manual supplied with the plotter. Your plotter may respond to other HP-GL commands not implemented via the 9872 ROM. See the Programming Note furnished with the plotter for details.

Setting up the plotter, from switching it on to loading paper, is covered in the operating manual or beginners guide supplied with your plotter. Be sure to carefully follow the set-up instructions in your plotter's manual before controlling it with the computer.

The following conventions are used in describing plotter syntax in this chapter.

`dot matrix` All items in dot matrix are required exactly as shown.

[] Items within brackets are optional.

Parameters can be numbers, variables or expressions. All plotter statements can be executed from the keyboard, in the live keyboard mode or within a program. See the HPL Syntax appendix for a brief summary of each statement.

Plotter Select Code Statement

PSC select code

A select code is a number that is used to specify which peripheral device is being addressed by the computer. The 9862A plotter interface is preset to select code 5. The HP 98034A HP-IB Interface and the HP-GL plotter, however, can be set to respond to various select codes and addresses. The plotter select code statement specifies the particular interface and plotter to which the computer will transmit the various Plotter ROM statements.

The 98034A HP-IB interface card is preset to select code 7 at the factory. To change the setting, rotate the switch on the card using a small screwdriver.

The select code value consists of either three digits or four digits that combine the two addresses as follows:

cdd	c or cc = one or two-digit interface select code of 0 or within the range of 2 thru 15.
or	
ccdd	dd = two digit plotter address code ranging from 0 thru 30.

The HP-IB interface select code can be one or two digits depending upon the setting (e.g., 7, 10, 14, etc.). The plotter address code, however, must always consist of two digits (e.g., 05, 06, 19, 26, etc.). Note that if the plotter address is a one digit number (e.g., 5), a leading zero must be added to it (e.g., 05). A list of HP-IB address switch settings is in the Reference Table appendix.


For example, to specify an interface set to 7 and a plotter set to 05, execute this statement:

```
PSC705
```

To specify an interface set to 12 and a plotter set to 16, execute this statement:

```
PSC1216
```

The Plotter ROM automatically sets a PSC 705 whenever any of the following conditions occur:

- The calculator is first switched on
-  is pressed or
- ERASE 0 is executed.

Once a PSC statement is executed, all plotter statements that are executed by the calculator will be addressed to the plotter specified by that PSC statement. To operate more than one plotter from the same calculator and interface, you must execute an appropriate PSC statement before each statement or group of statements that are sent to an individual plotter.

For example, you can plot on two different plotters (plotters A and B) connected to the same HP-IB interface, by setting each plotter to a different address code. Plotter A could be left at the factory setting of 05 (00101) and plotter B could be set to 06 (00110). Each statement or group of statements sent to a plotter must be preceded by the appropriate PSC statement. To address plotter A, PSC 705 is executed. To address plotter B, PSC 706 is executed.

If more than one plotter is being used and the different plotters also use different plotting scales, the appropriate scale statement must be executed each time the PSC is changed.

By executing PSC0 with the 9872A ROM, you can test all of the statements in a program except the plotter ROM statements. All plotter ROM statements are completely bypassed when PSC0 is executed. Any error messages that are due to non-plotter program errors will be displayed. None of the plotter ROM errors, however, will appear.

By executing PSC0 with the 9862A ROM, you can test a program without actually connecting the plotter to the computer. This is because all output to the plotter is bypassed when PSC0 is executed, and any error messages that would have appeared if the plotter were connected, are still displayed.

Plotting

Many statements are available for scaling in user units, drawing axes and plotting points. Except where noted, each statement is available with both plotter ROMs.

Scale Statement

```
scI [XP1# XP2# YP1# YP2]
```

The scale (`scI`) statement is used to locate the origin (point 0,0) and to specify the scale units to be used for your plot.

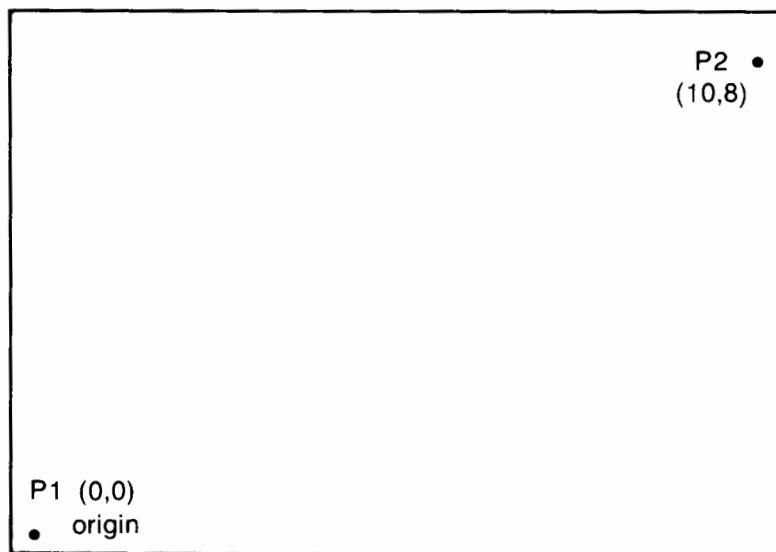
The scale statement uses four parameters which correspond to the x and y coordinate values that you wish to assign to the scaling points P1 and P2.

Once the units for a plot have been established, the physical size of the plot can be changed to fit a larger or smaller plotting area by resetting the scaling points, P1 and P2.

The following examples show various scale statements and the resulting origin locations.

The scale statement below locates the origin at the lower left scaling point, P1.

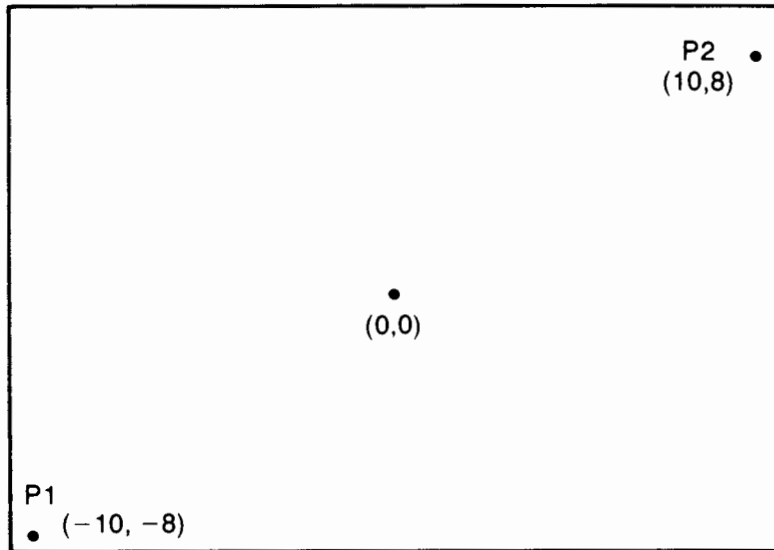
```
scI 0,10,0,8
```



7-8 Plotter Control

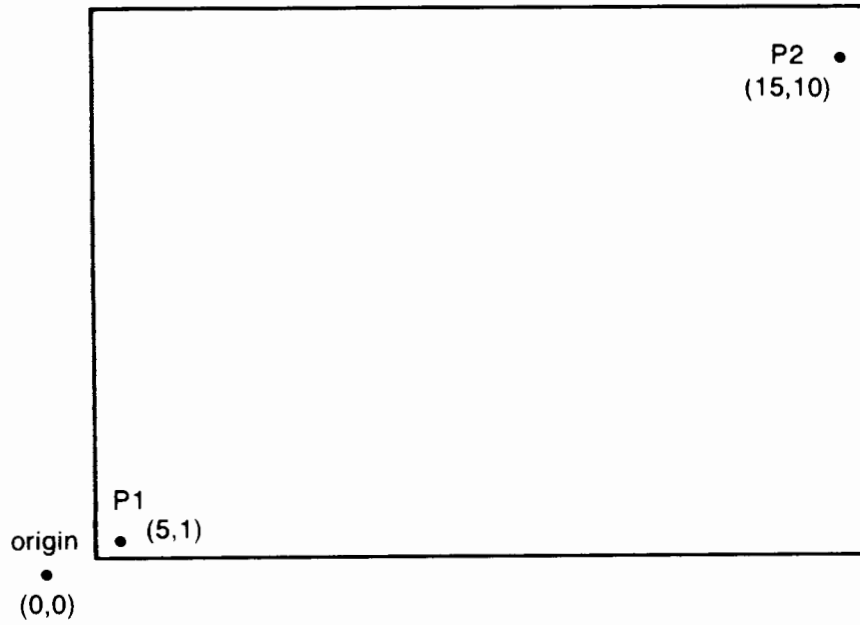
This scale statement locates the origin in the center of the plotting area.

```
sc1 -10,10,-8,8
```



The origin is located outside of the plotting area by this scale statement.

```
sc15,15,1,10
```

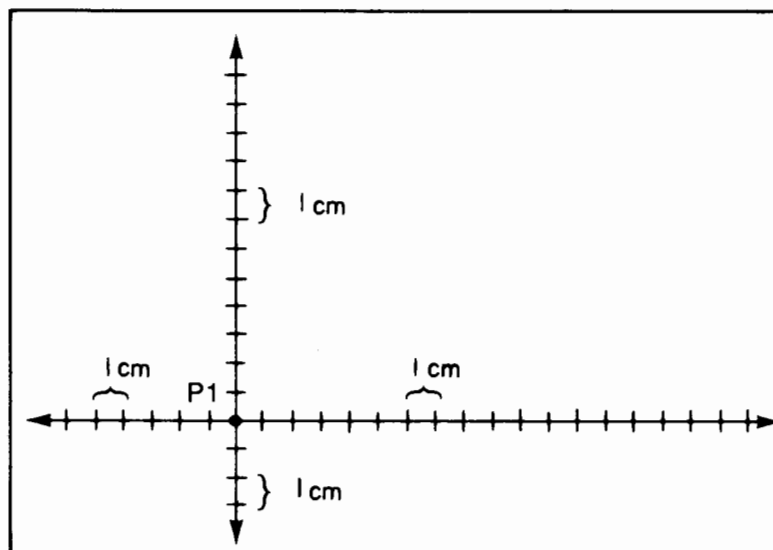


Once a scale statement has been executed, its parameters remain in effect until:

- Another scale statement is executed
- The `RESET` key is pressed
- `ERASE` is executed
- the computer is switched off or
- the HP-GL plotter is initialized.

A 9862A scale statement must be executed before plotting is attempted.

The following HP-GL scale is automatically set whenever the plotter is initialized or a scale statement without parameters is executed.



HP-GL Default Scale

This scale corresponds to a centimeter unit of measure of the platen area with the origin (0,0) at the current location of P1. Since the scale corresponds to a unit measure (centimeters), the P2 scaling point is not referenced.

Plotter Clear Statement (9872 ROM only)

`pc1r`

The plotter clear (`pc1r`) statement sets all parameters that have been sent to the plotter to their default values with the following exceptions:

- The current scale (`sc1`) statement parameters remain unchanged.
- the current plotter select code (`psec`) parameter remains unchanged.
- p1 and p2 remain unchanged.
- the current pen location remains unchanged although the pen is raised.
- The pen that is selected remains unchanged.

A complete listing of the default values for each statement's parameters is given in the Appendix.

NOTE

It is a good practice to execute a `pc1r` statement at the beginning of each program to clear any previously set conditions in the plotter.

9872A Axis Statements

`xxx Y-Offset [; Tic Interval[; Start Point[; End Point[; Number-of-Tics/Label]]]]`

The X Axis (`xxx`) statement draws a horizontal axis, with or without tic marks or unit labels.

`yyx X-Offset [; Tic Interval[; Start Point[; End Point[; Number-of-Tics/Label]]]]`

The Y Axis (`yyx`) statement draws a vertical axis, with or without tic marks or unit labels.

The y-offset parameter specifies the y coordinate at which the X axis will cross the Y axis.

The x-offset parameter specifies the x coordinate at which the Y axis will cross the X axis.

The tic interval parameter determines whether or not tic marks are drawn along the axis. If tic marks are to be drawn, the parameter value specifies the spacing, in scale statement units, between tics. A value of 0 results in no tic marks. If a parameter is not specified, a tic mark is drawn at each end of the axis.

The sign of the tic value can result in either normal tic marks being drawn or a tic mark drawn only at the starting point of the axis. This result is determined by the start and end point parameters and is explained next.



The start and end point parameters specify the location of the endpoints of the axis. The axis is always drawn from the start point to the end point. If the end point is not specified, the axis is drawn to the P2 coordinate value in the current scale statement (i.e., X_{P2} for the X axis and Y_{P2} for the Y axis). If both the start point and end point parameters are not specified, the axis is drawn from the P1 coordinate value to the P2 coordinate value specified by the current scale statement (i.e., X_{P1} to X_{P2} for the X axis and Y_{P1} to Y_{P2} for the Y axis).

The following relationship exists between the start and end point parameters and sign of the tic-interval parameter.

A positive tic interval results in:

Normal tic spacing if the start point is less than the end point.

A tic drawn only at the start point if the start point is greater than the end point.

A negative tic interval results in:

Normal tic spacing if the start point is greater than the endpoint.

A tic mark only at the start point if the start point is less than the end point.

The number-of-tics/label parameter determines whether or not the tic marks on an axis will be labeled. Specifying either a 0 or no parameter results in no labels. If labels are desired, the parameter specifies the number of tic marks between labels. A negative parameter will result in only the labels being lettered without the axis or tic marks being drawn. Labels will be lettered on an axis only if a non-zero tic parameter is specified.

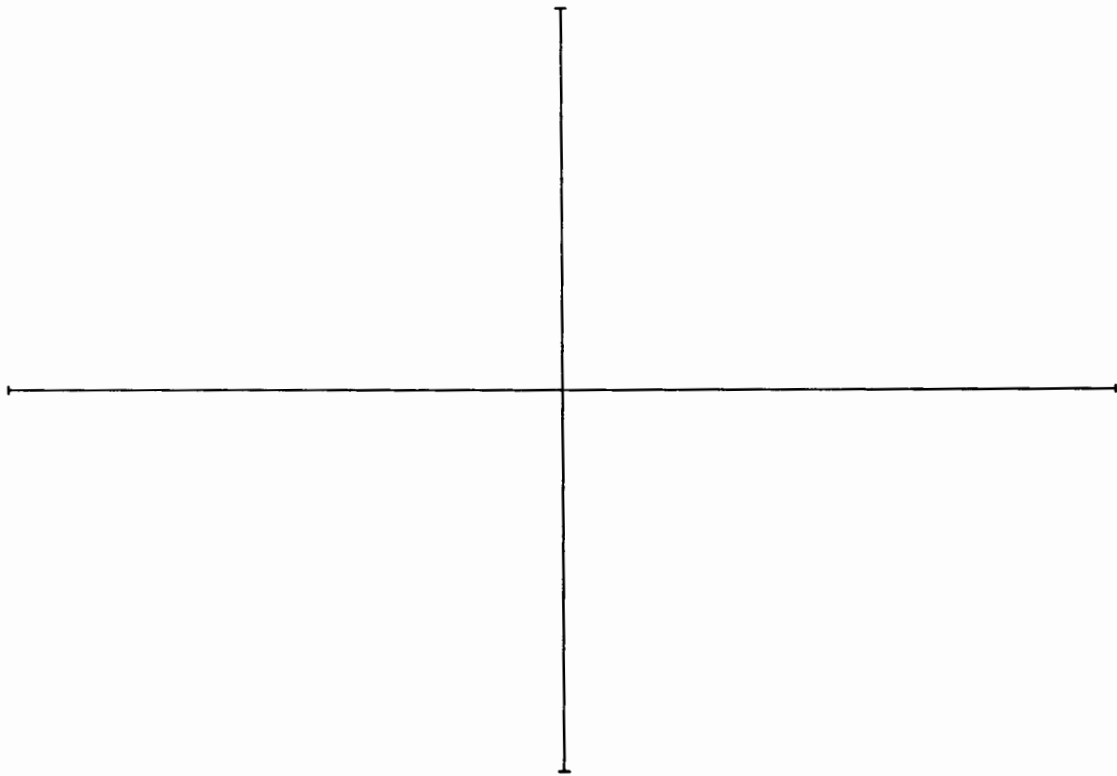
All labels are lettered according to the current character size (`csiz`) statement and in current number format (fixed or float statement). Character size is explained in Chapter 3, 'Lettering'.

All of the following axes drawing examples use this scale statement:

```
scal -10, 10 -8, 8
```

The following axes are drawn when only the offset parameters are specified.

```
xax 0  
yax 0
```

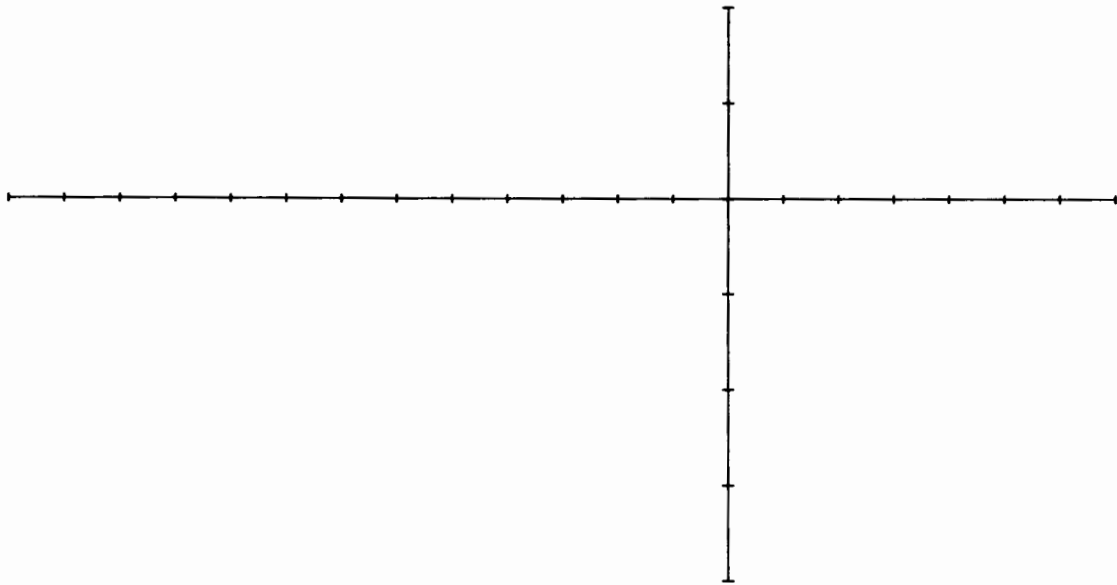


The axes cross at the point 0,0 with tic marks drawn only at the end points. The X axis is drawn from -10 to 10 (X_{P1} to X_{P2}) and the Y axis is drawn from -8 to 8 (Y_{P1} to Y_{P2}).

7-14 Plotter Control

These statements draw the following axes:

```
xax 2,1  
yax 3,2,-6,6
```



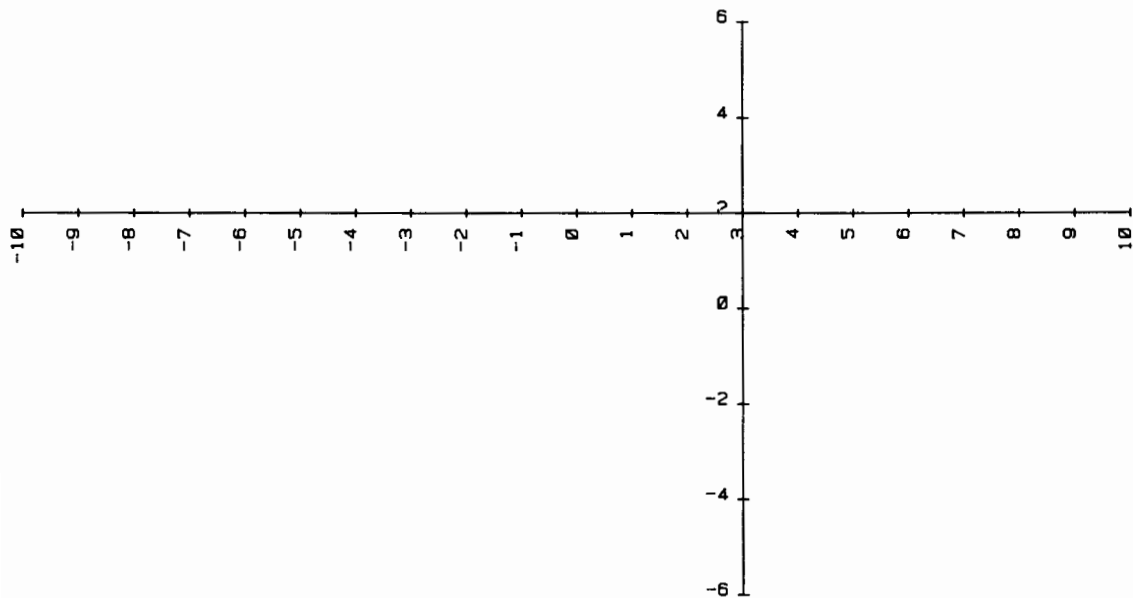
The X axis crosses the Y axis at the point $Y = 2$. Since the start and end point parameters aren't specified, the axis is drawn from -10 to 10 (X_{P1} to X_{P2}) with tic marks drawn at every unit (a tic interval of 1). No labels are specified.

The Y axis crosses the X axis at the point $X = 3$. The axis is drawn from -6 to 6 with tic marks every 2 units. Note that normal tic marks result from using a positive tic parameter with the start point (-6) less than the end point (6). No labels are specified.

These instructions draw the same axes as the previous instructions , but add labels in a fixed 0 number format.

```
fxd 0  
xax 2,1,-10,10,1  
yax 3,2,-6,6,1
```

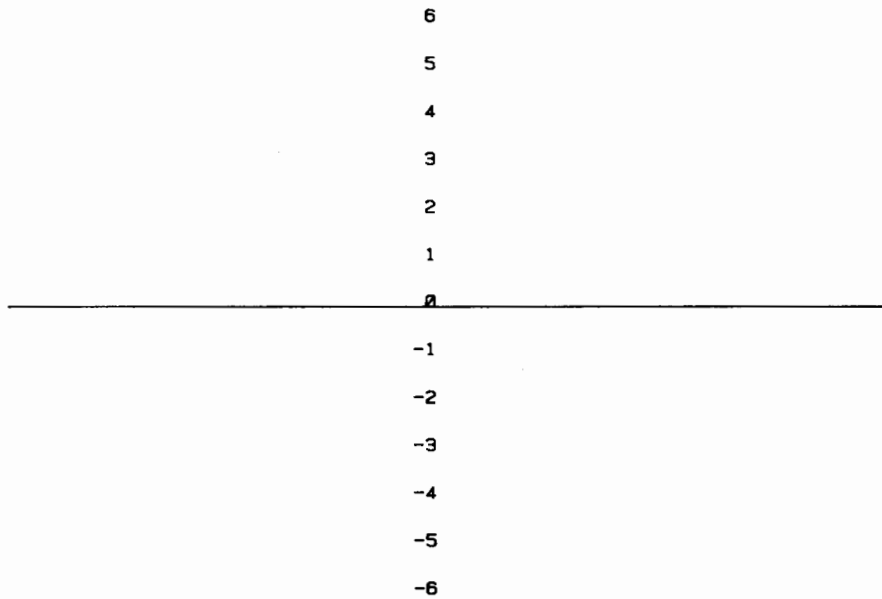
Note that to specify labels in the statement, the start and end point parameters must also be specified.



7-16 Plotter Control

These statements draw the following axes:

```
fxd 0  
xax 0,0,-8,8,1  
yax 0,1,-6,6,-1
```



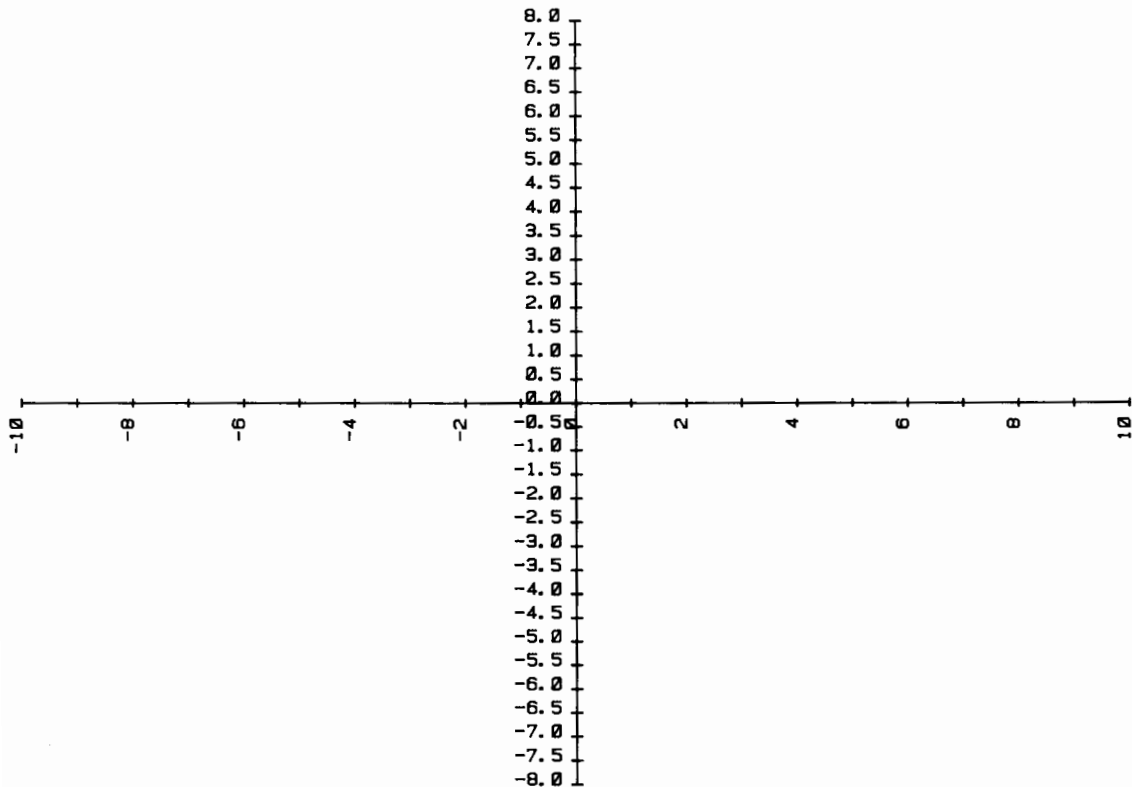
Since the X axis is drawn without tic marks, no labels were drawn even though they were specified. Labels cannot be lettered if tic marks are not specified.

The yax statement specifies a negative number-of-tics/label parameter which results in the labels being lettered without the axis and tic marks being drawn.

The following statements draw, tic mark and label the X and Y axes as shown.

```

fxd 0
xax 0,1,-10,10,2
fxd 1
yax 0,-.5,8,-8,1
    
```



The X axis is drawn from -10 to 10 with 1 unit tic marks and it is labeled every 2 units, beginning with the start point, in a fixed 0 number format.

The Y axis is drawn from 8 to -8 with $.5$ unit tic marks. (Note that normal tic marks are drawn since the tic interval is negative and the start point is greater than the end point). Labels are lettered at every tic mark in a fixed 1 number format.

The axes cross at the point $(0,0)$.

NOTE

Axes are drawn with currently specified line type and symbol mode character. Line type and symbol mode are explained later in this chapter.

9862 Axis Statement

```
AXE Xcoordinate Ycoordinate [ Xtic [ Ytic]]
```

The axis (AXE) statement draws both the X and Y axes over the entire plotting area, crossing at the point specified by the X and Y parameters in the statement. The pen is automatically raised before moving to the starting point and again after the axes are drawn.

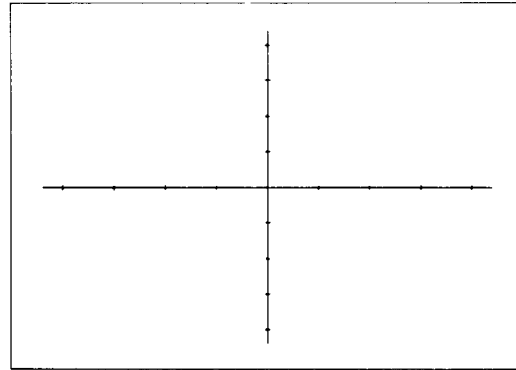
To use the axis statement, two parameters are required - the coordinates of the X and Y axes crossover point. Two other parameters are optional and can be used to specify tic marks along the X and Y axes.

Tic spacing is measured in both directions from the axes' origin so that one tic from each axis falls on the axes crossover point. (Tic spacing is not measured from the scale origin or either edge of the plotting area.)



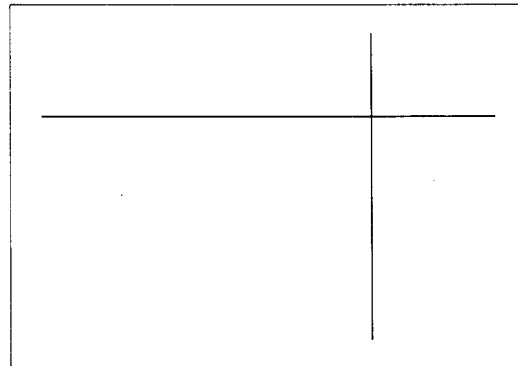
For example, key in and execute this scale statement: `sc1 -2.2, 2.2, -2.2, 2.2`. Then key in and execute -

```
axe 0, 0, .5, .5
```



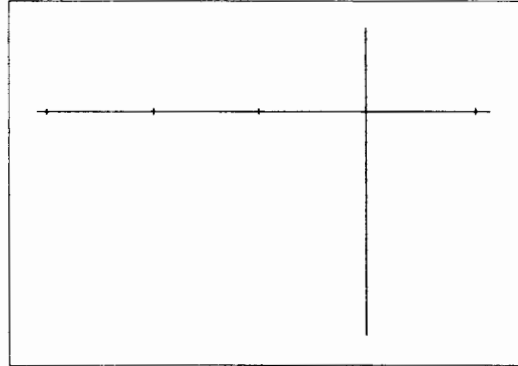
If tic parameters are not given, axes are drawn without tic marks. Replace the plotter paper and execute this axis statement -

```
axe 1, 1
```



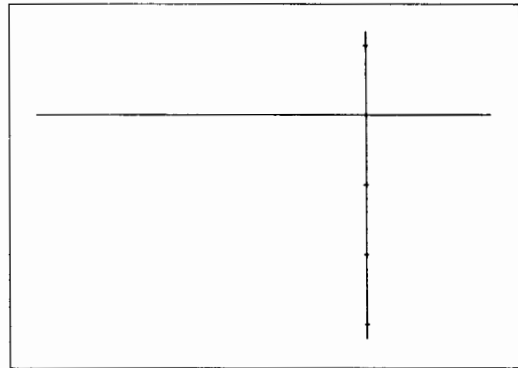
If only three parameters are given in an axis statement, then tic marks are drawn on the X axis only. To draw in tic marks on the X axis of the previous example, execute -

```
axe 1, 1, 1
```



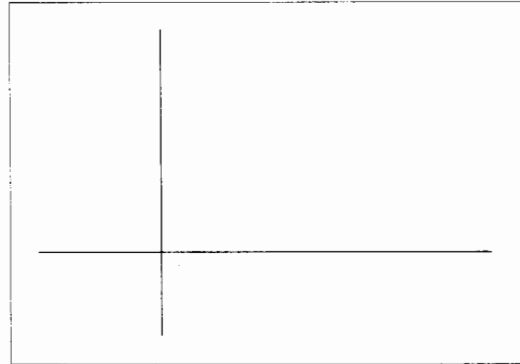
To draw tic marks on the Y axis only, change the plotting paper and use a zero for the X_{tic} parameter.

```
axe 1, 1, 0, 1
```



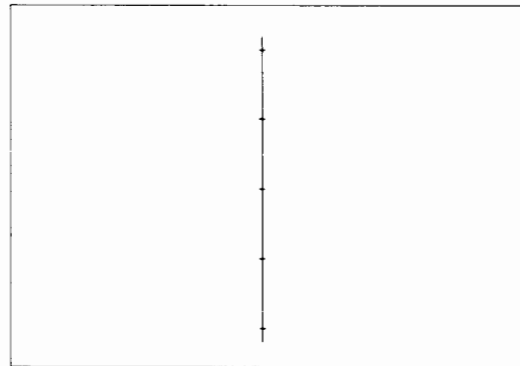
If the tic spacing specified is so large that the tic marks will be off-scale, tic marks are drawn at the crossover point only. Change the plotting paper and execute -

```
axe -1, -1, 5, 5
```



If a tic spacing parameter is negative, neither the axis nor tic marks for that axis is drawn. For example -

```
axe 0, 0, -1, 1
```



Pen Statement

```
Pen
```

The pen (`Pen`) statement raises the pen without moving it to a new location. This statement requires no parameters.

Instructions to raise or lower the pen before or after movement can also be included as a parameter in the plot or iplot statements.

Plot Statement

```
Plot Xcoordinate # Ycoordinate [ # pen control]
```

The plot (`Plot`) statement moves the pen to the point specified by the X and Y coordinate parameters in the statement.

The plot statement requires that both the X and Y coordinates be specified. An optional pen control parameter raises or lowers the pen before or after movement.

If the point specified by a plot statement lies off the platen surface, a line is drawn to the platen limit and then the pen is raised. The pen remains raised until a point on the platen is specified.

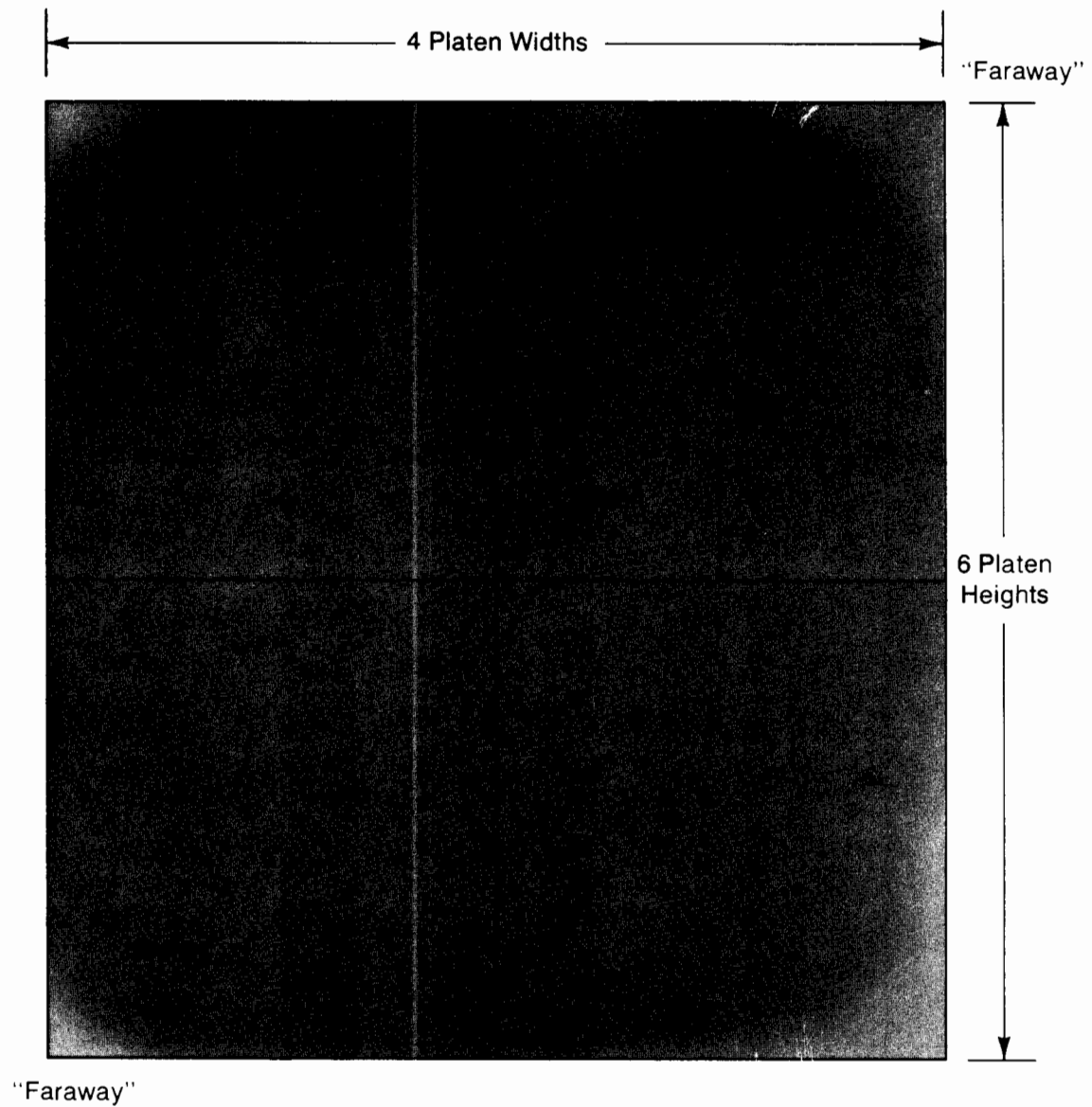
The optional pen control parameter can be an integer in the range –32768 thru 32767.

An odd, positive integer	The pen lifts before moving.
An odd, negative integer	The pen lifts after moving.
An even, positive integer	The pen lowers before moving.
An even, negative integer	The pen lowers after moving.
0	No change.
No parameters	The pen remains in its present position, moves to the point specified, and lowers or remains down.

NOTE

A pen control parameter of 0 or a positive integer should be used with all plot statements executed in interrupt service routines (see chapter 5).

If the point lies off the platen but is within the 'nearby' area (shown below), the out-of-limit light will turn on. If the point lies in the 'faraway' area, the out-of-limit light will blink.



9872 Out-of-limit Areas

The following example programs plot values of the function $Y = X^2$.

9872 ROM

```

0: pclr
1: scl -10,10,-
  10,100
2: fxd 0
3: xax 0,1,-10,
  10,1
4: yax 0,10,-10,
  100,1
5: -10+X
6: plt X,X↑2
7: if (X+1+X)≤1
0:jmp -1
8: pen
9: end

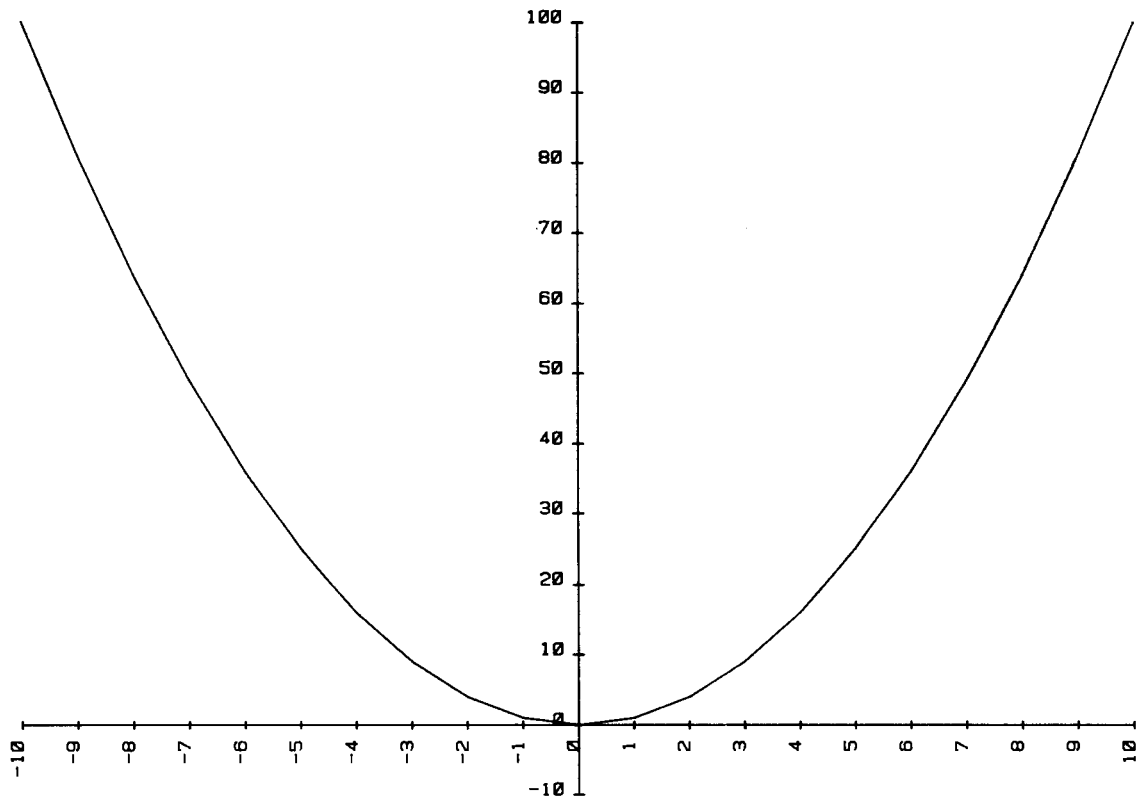
```

9862 ROM

```

0: scl -10,10,-
  10,100
1: fxd 0
2: axe 0,0,1,10
3: -10+X
4: plt X,X↑2
5: if (X+1+X)≤1
0:jmp -1
6: pen
7: end

```



Plot of $y = x^2$ (labels not printed with 9862 ROM)

Pen Select Statement (9872 ROM only)

The pen select (`pen#`) statement provides the capability of programmed pen selection.

`pen#[pen-position number]`

When the `pen#` statement is executed, the pen arm raises the pen it is currently holding (if any) and returns it to an empty pen-storage position. If a valid pen-position number is specified (1 thru 4), the pen in that position is taken and the pen arm returns to its last location on the platen.

A parameter value of 0 or no parameter directs the pen arm to return the pen it is currently using to an empty storage position without taking a new pen.

If the specified pen position is empty or if all of the pen positions are full and there is a pen in the arm, then no operation occurs.

The following example plots three functions on the same plot with each function a different color.

```

0: pclr;fxd 0
1: scl -10,10,-
  100,100
2: pen# 1;xax 0,
  1,-10,10,1
3: yax 0,10,-
  100,100,1
4: pen# 2;-10+X
5: plt X,X+2;if
  (X+1+X)<=10;
  jmp 0
6: pen# 3;-10+X;
  pen
7: plt X,10*X;
  if (X+1+X)<=10;
  jmp 0
8: pen# 4;-10+X;
  pen
9: plt X,-10X;
  if (X+1+X)<=10;
  jmp 0
10: pen# ;plt
  10,100,1
11: end

```

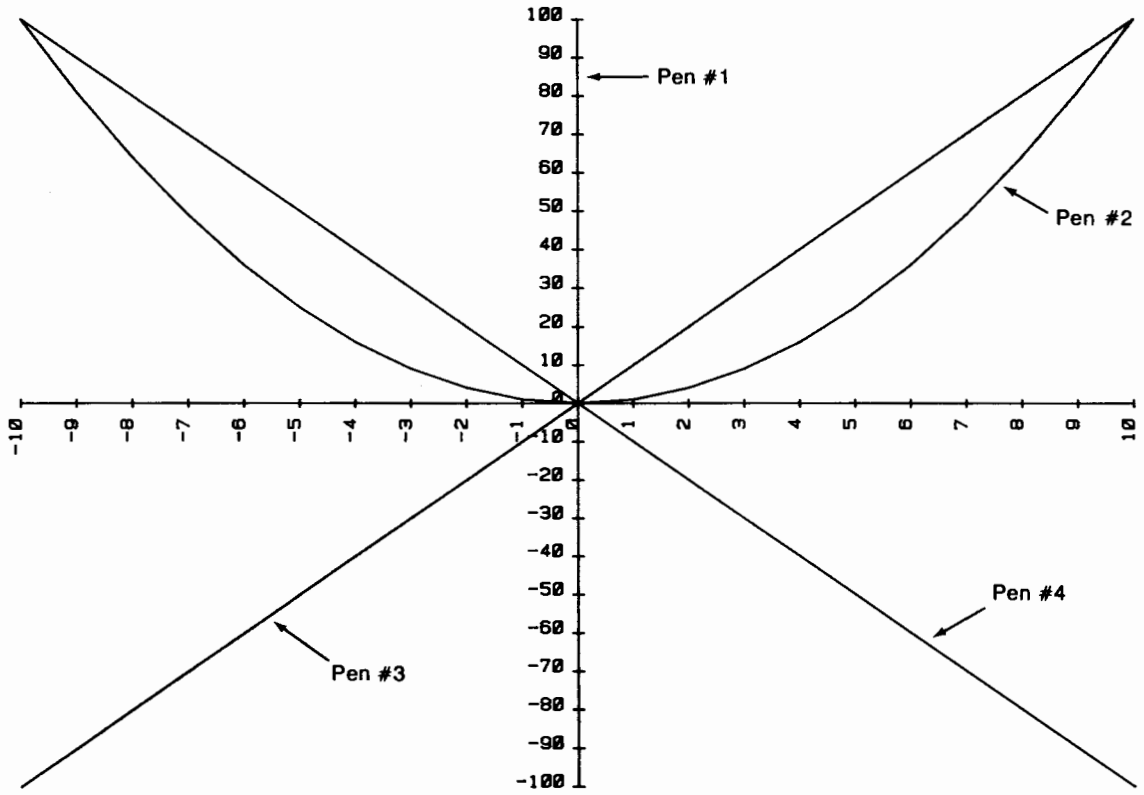
The axes are drawn and labeled
in black (pen#1)

The function ($Y = X^2$) is plotted
in red (pen#2)

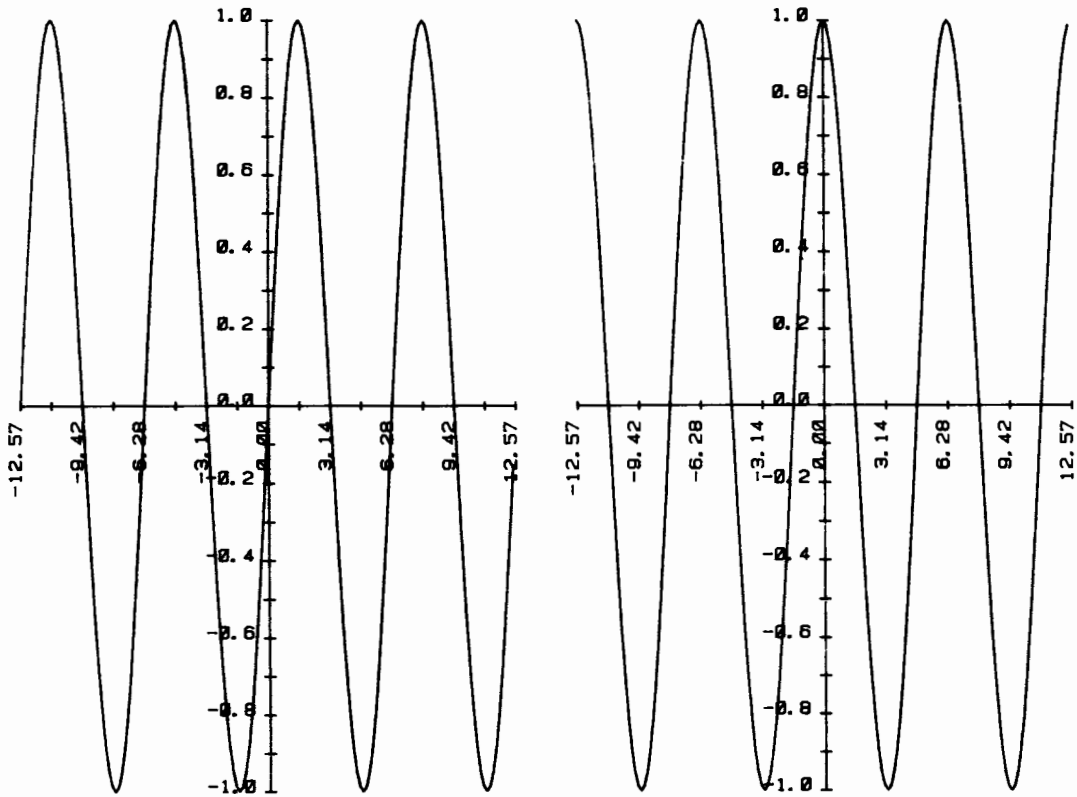
The function $Y = 10X$ is plotted
in green (pen#3)

The function $Y = -10X$ is plotted
in blue (pen#4)

The pen is put away.



Pen Select Example Plot



Offset Plotting with 9872A Plotter

Offset Statement

The offset (`ofs`) statement moves the origin (point 0, 0) from its present location to a new position specified by the X and Y increment values in the statement.

`ofs Xincrement Yincrement`



The X increment specifies the number of horizontal scale-statement units that the origin is to be moved. The Y increment specifies the number of vertical scale-statement units that the origin is to be moved. The signs of the increment-parameters specify the direction that the origin moves as follows:

- A positive parameter moves the origin in a positive direction as defined by the current scale statement.
- A negative parameter moves the origin in a negative direction as defined by the current scale statement.

Here is an example that uses the offset statement to make two separate plots on the same sheet of paper. The 9872 ROM is used here. The plot is on the facing page.

```

0: pclr scl -9π,
  9π, -1, 1
1: radi -4π → X
2: ofs -4.5π, 0
3: pen# 1 fixd 2;
  xax 0, .5π, -4π,
  4π, 2
4: fixd fixax 0;
  .1, -1, 1, 2
5: pen# 2 plot X,
  sin(X) 11 (X+π/
  20 → X) <= 4π; jmp 0
6: ofs 9π, 0
  4π, 2
7: pen# 1 fixd 2;
  xax 0, .5π, -4π,
  4π, 2
8: fixd fixax 0;
  .1, -1, 1, 2
9: pen# 1 plot X,
  cos(X) 11 (X+π/
  20 → X) <= 4π; jmp 0
10: pen# iend

```

Sets the scale large enough for both plots.

The offset statement moves the origin from the center of the page, 4.5π units to the left.

Draws the X axis and the Y axis at this origin.

Plots $\sin X$ from -4π to 4π .

The offset statement moves the origin 9π units horizontally and 0 units vertically.

Draws an X axis and a Y axis at the new origin.

Plots $\cos X$ from -4π to 4π .

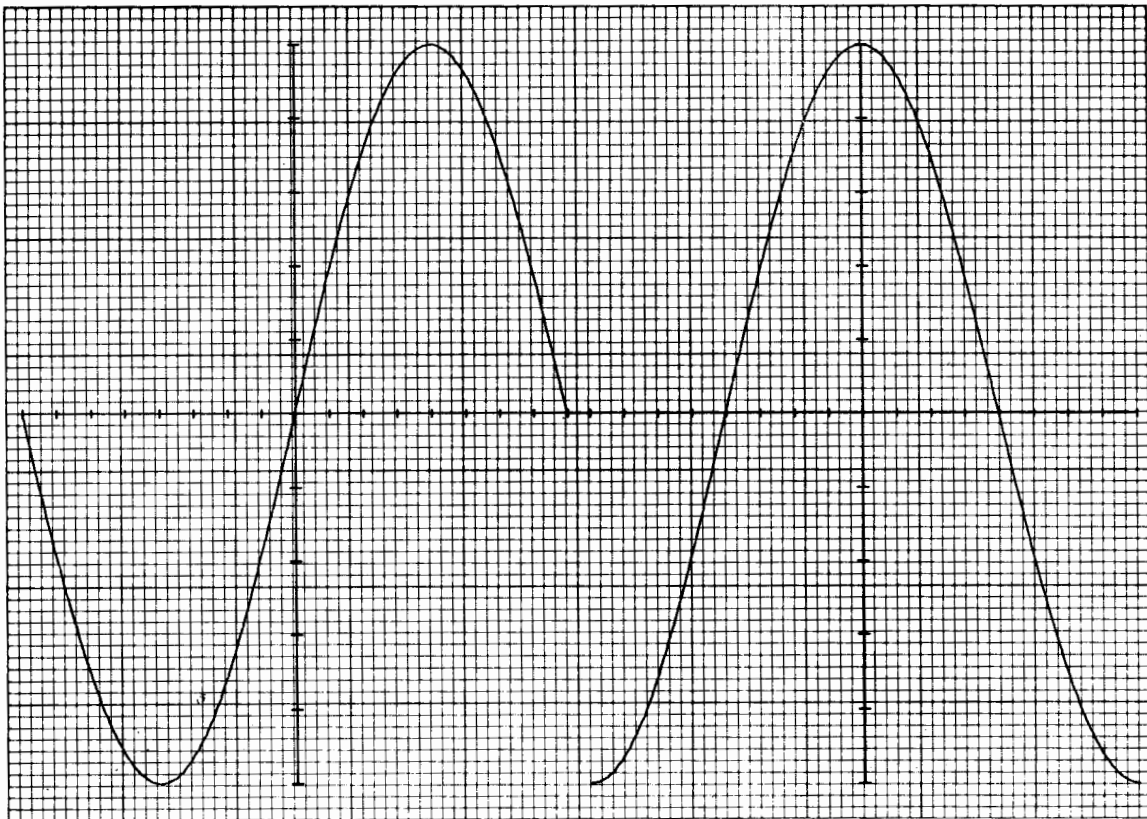
Replaces the pen.

Here's an example that uses the 9862 ROM offset statement to make two separate plots over the same range, on the same sheet of paper. Lines 2 through 7 plot the function on the left and lines 8 through 13 plot the function of the right. (The third parameter in line 9 eliminates redrawing the X axis for the second plot; only the Y axis is drawn.) A plotting area within 7 x 10 inch plotter paper is used.

```

0: scl 0,800,0,
  2.2
1: deg
2: ofs 200,1.1
3: axe 0,0,22.5,
  .2
4: -180*X
5: plt X,sin(X)
6: X+5*X
7: if X<185;eto
  5
8: ofs 400,0
9: axe 0,0,-1,.2
10: -180*Y
11: plt Y,cos(Y)
12: Y+5*Y
13: if Y<185;
  eto 11
14: sen
15: end

```



Offset Plotting with the 9862A Plotter

Incremental Plot Statement

The incremental plot (`iplt`) statement moves the pen from its current location to the new location specified by the X and Y parameters.

```
iplt X_increment : Y_increment [ : pen control]
```

The X increment parameter specifies the number of scale-statement units that the pen is to move horizontally. The Y increment specifies the number of scale-statement units that the pen is to move vertically. The signs of the increment parameters determine the relative direction that the pen moves as follows:

- A positive value moves the pen in a positive direction as defined by the current scale statement.
- A negative value moves the pen in a negative direction.
- The optional pen control parameter is the same as that used with the plot statement.

An odd, positive integer	The pen lifts before moving.
An odd, negative integer	The pen lifts after moving.
An even, positive integer	The pen lowers before moving.
An even, negative integer	The pen lowers after moving.
0	No change.
No parameter	The pen remains in its present position, moves to the point specified, and lowers or remains down.

The following example uses iplot statements to draw crosses at five different locations. The iplot statements are programmed in a subroutine and the origin for each cross is located by an offset statement.

9872A ROM

```

0: pclr: scl 0,
  100,0,70
1: ofs 20,20
  pen# 1: esb 7
2: ofs 0,30: pen#
  2: esb 7
3: ofs 30,-15:
  pen# 3: esb 7
4: ofs 30,15:
  pen# 4: esb 7
5: ofs 0,-30:
  pen# 2: esb 7
6: pen# iend
7: plt -3,3,1
8: iplt 0,7,2
9: iplt 6,0
10: iplt 0,-7
11: iplt 7,0
12: iplt 0,-6
13: iplt -7,0
14: iplt 0,-7
15: iplt -6,0
16: iplt 0,7
17: iplt -7,0
18: iplt 0,6
19: iplt 7,0,-1
20: ret

```

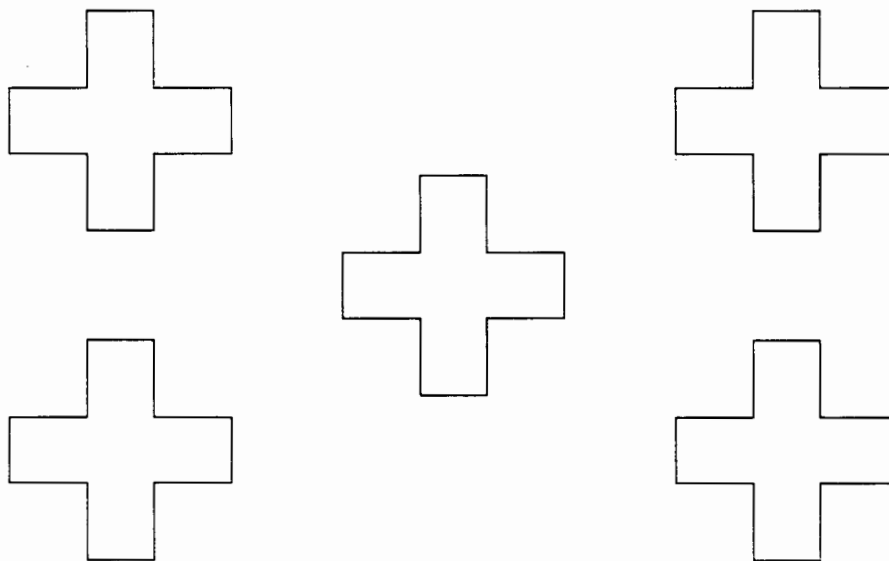
9862A ROM

```

0: scl 0,100,0,
  70
1: ofs 20,20
2: esb 12
3: ofs 0,30
4: esb 12
5: ofs 60,0
6: esb 12
7: ofs 0,-30
8: esb 12
9: ofs -30,15
10: esb 12
11: stop
12: plt -3,3,1
13: iplt 0,7,2
14: iplt 6,0
15: iplt 0,-7
16: iplt 7,0
17: iplt 0,-6
18: iplt -7,0
19: iplt 0,-7
20: iplt -6,0
21: iplt 0,7
22: iplt -7,0
23: iplt 0,6
24: iplt 7,0,-1
25: ret

```

If an iplot statement specifies a point off the platen, the pen draws a line to the limit of the platen and stops. If the point lies off the platen in the "nearby" area (see page 7-23), the out-of-limit light turns on. The plotter recognizes iplot statements in this area. If the point specified lies in the "faraway" area, the out-of-limit light flashes and the plotter does not recognize iplot statements. A regular plot (plt) statement must be used to specify a point that is either on the platen or in the "nearby" area before any further iplot statements are recognized by the plotter.



Incremental Plot Example

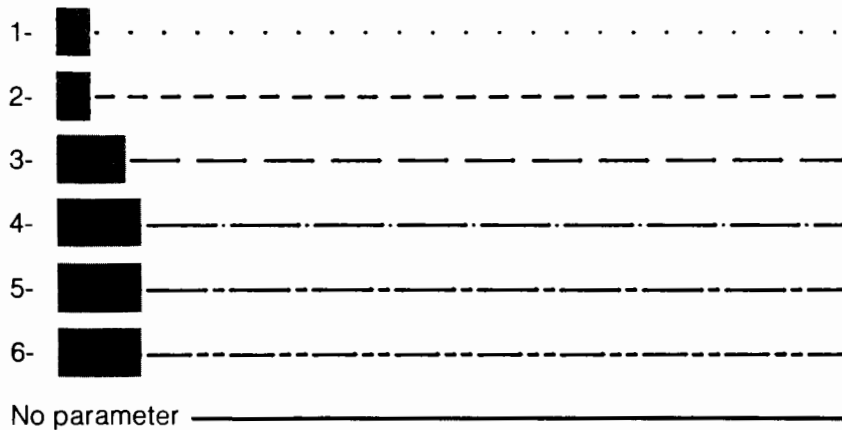
Line Type Statement (9872 ROM only)

The line type (`line`) statement specifies the type of line that will be used with `plt`, `iplt`, `xax` and `yax` statements.

`line [pattern number [, pattern length]]`

Shown below are the line patterns and their pattern numbers.

0-specifies dots only at the points that are plotted.



The shaded portion of each of the line patterns above is one complete segment of the pattern.

The optional pattern length parameter specifies the length of one complete segment of the pattern and is expressed as a percentage of the diagonal distance between the scaling points, P1 and P2. If a pattern length parameter is not specified, a length of 4% is used. The range of the pattern length parameter is from 0 thru 127.99994999.

This example plots 5 crosses with each cross plotted in a different line type.

```

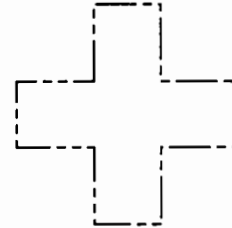
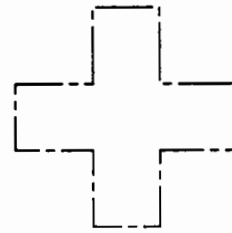
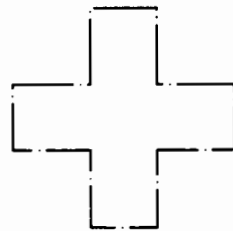
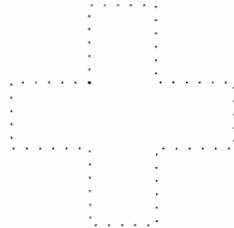
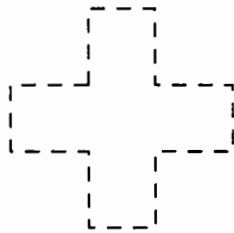
0: pclr:sc1 0,
  100,0,70
1: ofs 20,20;
  line 1,1;asb 7
2: ofs 0,30;line
  2,2;asb 7
3: ofs 30,-15;
  line 4,6;asb 7
4: ofs 30,15;
  line 5;asb 7
5: ofs 0,-30;
  line 6;asb 7
6: pen# ;end

```

```

7: plt -3,3,1
8: iplt 0,7,2
9: iplt 6,0
10: iplt 0,-7
11: iplt 7,0
12: iplt 0,-6
13: iplt -7,0
14: iplt 0,-7
15: iplt -6,0
16: iplt 0,7
17: iplt -7,0
18: iplt 0,6
19: iplt 7,0,-1
20: ret

```



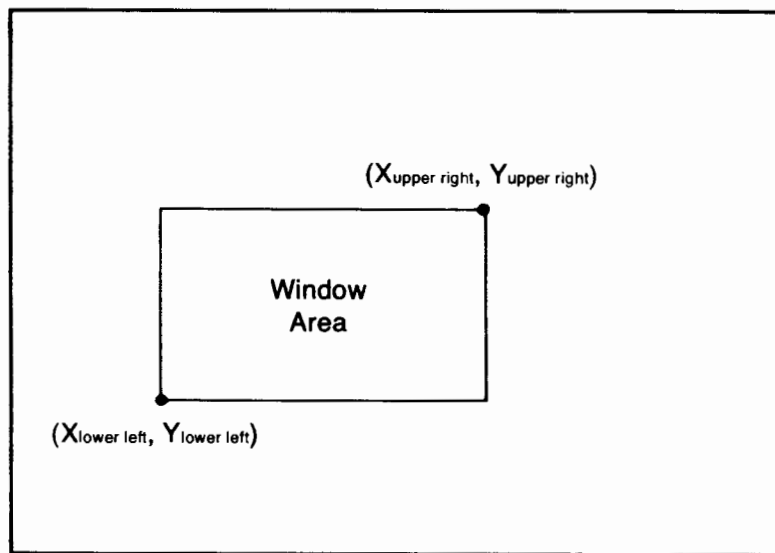
Line Type Example Plot

Limit Statement (9872 ROM only)

The limit (`Lim`) statement restricts programmed pen motion to a specific rectangular area on the platen. This area is called the "window".

$$\text{Lim}[\text{X}_{\text{lower left}} \text{ } \text{X}_{\text{upper right}} \text{ } \text{Y}_{\text{lower left}} \text{ } \text{Y}_{\text{upper right}}]$$

The four parameters specify, in the current scale-statement units, the X and Y coordinates of the lower left and the upper right corners of the window area as shown below.



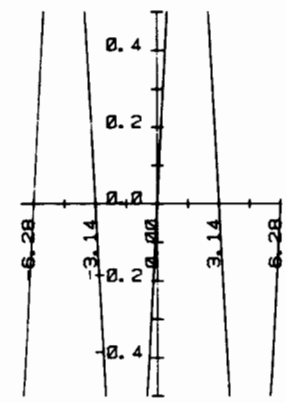
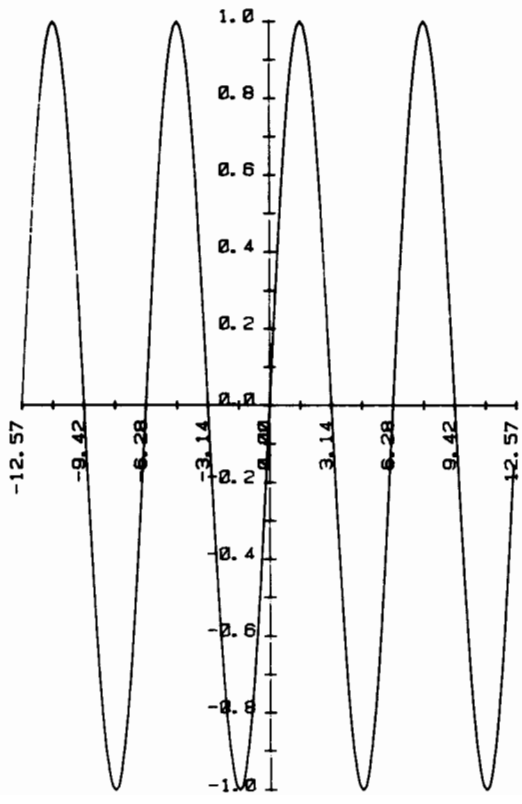
If a limit statement is not executed, or if a limit statement without parameters is executed, the window is automatically set at the mechanical limits of the plotter.

The limit statement can be used to emphasize a specific portion of a plot. The following example plots the function $\sin(X)$ from -4π to 4π and -1 to 1 on the left side of the paper. An offset statement is then used to move the plot origin to the right side of the paper. The plot of $\sin(X)$ is repeated except that a window is specified by a limit statement to restrict the plot so that only the points contained in the rectangle -2.2π to 2π and $-.5$ to $.5$ are plotted.

```

0: polrisc1 -9π,
  9π,-1,1
1: rad;-4π→X
2: ofs -4.5π,0
3: pen# 1;fxd 2;
  xax 0,.5π,-4π,
  4π,2
4: fxd 1;yax 0,
  .1,-1,1,2
5: pen# 2;plt X,
  sin(X);if (X+π/
  20→X)≤4π;jmp 0
6: ofs 9π,0;-
  4π→X
7: lim -2.2π,2π,
  -.5,.5
8: pen# 3;fxd 2;
  xax 0,.5π,-4π,
  4π,2
9: fxd 1;yax 0,
  .1,-1,1,2
10: pen# 4;plt
  X,sin(X);if (X+
  π/20→X)≤4π;
  jmp 0
11: pen# ;end
  
```

lim statement sets the window for the sin (X) plot.



Lettering

This section describes the statements that enable you to letter alphanumeric characters and symbols with the plotter. You can also specify the size and width of the characters (relative to the scaling points), as well as the direction in which the characters are lettered.

Label Statement

The label (`lbl`) statement enables you to letter characters on the plotter. This statement is used like a print (`prt`) statement to letter expressions, text or string variables on the plotter.

```
lbl any combination of text, expressions or string variables
```

Text is specified in a label statement by enclosing it in quotes. Here is an example of text in a label statement.

```
lbl "9872A Plotter"
```

Here is an example that letters expressions.

```
lbl X, X+1, X+2
```

The value assigned to X will be lettered in the current number format (fixed or float). The value resulting from the expression X+1 is lettered next, followed by the value resulting from the expression X+2. The digits in these expressions are lettered as a string of characters. This requires you to add any spaces needed to fit the numbers into the context of the item being lettered.

For example, this statement letters the same expressions above with four spaces between each of the values.

```
l b l X, "      ", X+1, "      ", X+2, "      "
```

This example letters the characters contained in the string variable A\$ (the String Variable ROM is required).

```
l b l A$
```

Before using the label statement, the pen should be moved to the location where labeling is to begin by using one of the plot statements (cplot, iplot or plot) or by using the four direction controls on the plotter front panel. This point will be the lower-left corner of the first character. After lettering a character, the pen stops at the lower-left corner of the next character space.

9872 Character Sets

The 9872 plotter has five character set available. The plotter, when initialized, automatically sets both the "standard" set and "alternate" set to the ANSI-ASCII character set (set 0). The `PCLr` statement, however, designates the 9825 character set (set 1) as the standard set. Refer to the programming manual supplied with the plotter for a complete listing of the characters in each character set and the statements used to change the character set designated as either standard or alternate.

Character Set 1

```
! "#$%&' () *+, -. /0123456789: ; <=>?
@ABCDEFGHIJKLMN O PQRSTU VWXYZ [ \ ] ^ _
` abcdefghijklmnopqrstu vwxyz π τ → ~
```

9872 Character Set (set 1)

Character Size Statement

The `csize` (`csiz`) statement specifies the size and shape of the characters and symbols as well as the direction in which they are to be lettered.

```
csiz[height [ ; aspect ratio [ ; paper ratio [ ; angle of rotation ] ] ] ]
```

The `csize` statement can specify up to four parameters. If any of the parameters are omitted, a specific default value for the parameter is assumed. Note that when a parameter is omitted, the parameter listed to its right must be omitted as well.

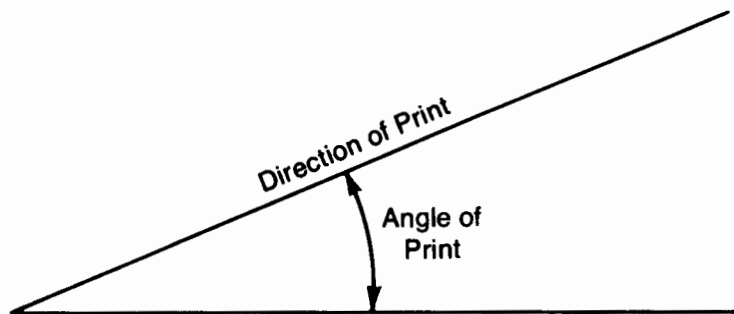
Here is a description of each of the four parameters:

The height parameter specifies the height of the characters as a percent of the scale height defined by the scaling points, P1 and P2. This parameter must be within the range of 0 thru 127.99994999.

The aspect ratio parameter specifies the ratio of the height of a character to its width. For example, an aspect ratio of 2 specifies characters that are twice as high as they are wide. An aspect ratio of 1 specifies square characters.

The paper ratio parameter specifies the ratio of the height of the scaling area to its width. The scaling area is defined by the scaling points, P1 and P2. For example, a 10 inch high by 15 inch wide scaling area has a paper ratio of 10/15 or 2/3.

The angle of rotation parameter specifies the direction in which the characters are printed. The direction is expressed as the angle (measured counter-clockwise) between the line of print and the X axis, as shown below.



This parameter is expressed in the current angular units (degrees, radians or grads).

The default values for the four parameters are as follows:

Height	1.5%
Aspect Ratio	2
Paper Ratio	1
Angle of Rotation	0

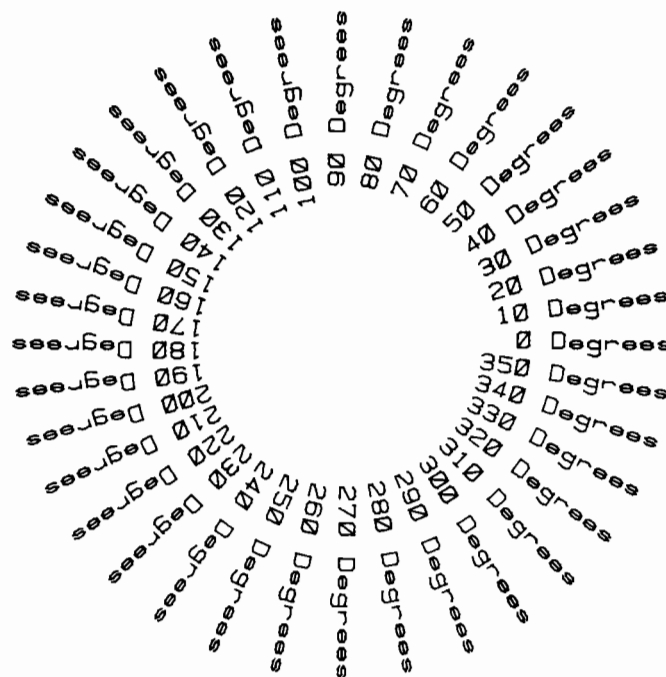
Executing a csize statement without parameters, sets the default values. These values are also set when the plotter is initialized or cleared (pclr).

The following example program uses the csize instruction to specify the character dimensions and shape and to rotate the lettering direction through an entire circle in 10 degree intervals. The plot statement centers the pen for each printing sequence.

The if . . . lines add extra spaces (if needed) to right-justify the values that precede the word "Degrees".

```

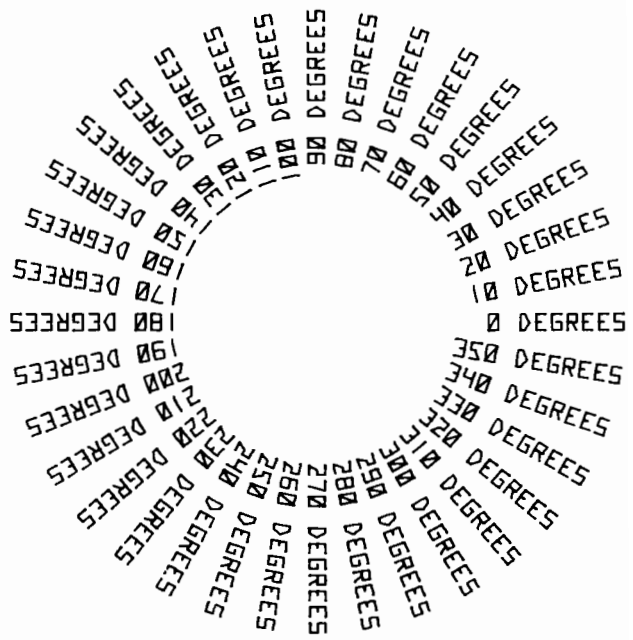
0: pclr;10→R→X;
   deefxd 0
1: ac1 0,30,0,24
2: csize 2,1,5,8/
   10,R
3: plt 15,12,1
4: if R<10;1b1
5: if R<99;1b1
6: 1b1
   ",R," Degrees"
7: if (R+10→R)<3
   60;jmp -5
8: plt 30,20,1
9: end
    
```



Angle of Rotation Plot with the 9872A


```

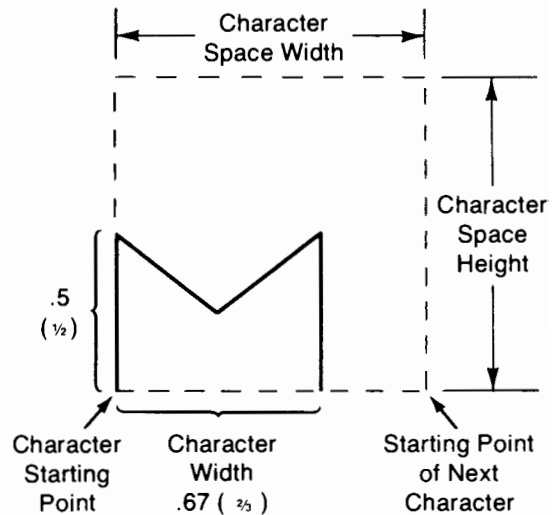
0: scl 0,10,0,7
1: des
2: fxd 0
3: 0→X
4: plt 5,3.5,1
5: csiz 2,1.7,7/
  10,X
6: cplt 8,-.3
7: if X<10;1b1
  " "
8: if X<100;1b1
  " "
9: 1b1 X," DEGRE
  ES"
10: X+10→X
11: if X<360;
  goto 4
12: end
    
```



Angle of Rotation Plot with the 9862A

Spacing Between Characters

In the diagram at the right, you can see the relative position of a character, in this case M, within the character-space field. The character-space field is set indirectly by the csize statement; with the 9872 ROM, the character space height is twice the character's height and the character space width is 1 1/2 times the character width. With the 9862 ROM, however, the height and width of a character is always 6/10 of the height and 6/10 of the width of its character space field.



9872 Character Spacing

When you specify the height of a character in a csize statement, however, you should specify the character height and not the height of the character-space field.



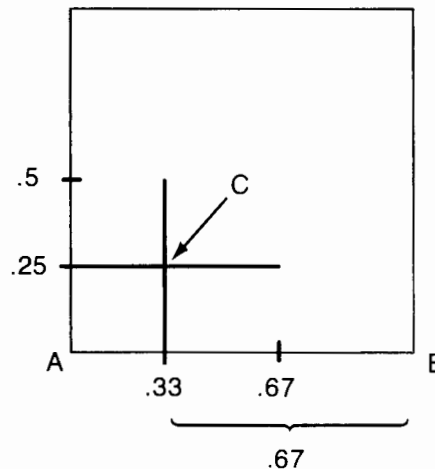
Character Plot Statement

The character plot (`cPlt`) statement moves the pen the specified number of character-space fields. If no parameters are specified (9872 ROM only), a `cPlt` statement performs a carriage return and line feed operation by moving one character-space height down and returning to the margin defined by the last point that the pen was sent to by either a plot statement, `iplot` statement, or the plotter front-panel controls. If a `csize` statement is executed after the pen is positioned by a plot, `iplot` or the front panel controls, the location of the pen when the `csize` statement is executed becomes the margin that the pen returns to when a `cPlt` is executed without parameters.

`cPlt`[character space widths ; character space heights]

When parameters are specified, the `cPlt` statement moves the pen the specified number of character-space widths to the right (a positive value) or to the left (a negative value) and the number of character-space heights up (a positive value) or down (a negative value). The pen's position (raised or lowered) does not change when a `cPlt` statement is executed. The parameters must be within the range of ± 127.9994999 .

The diagram at the right shows the character spacing around the symbol `+`. The pen begins to draw the symbol at point A and ends at point B, ready to draw another character.

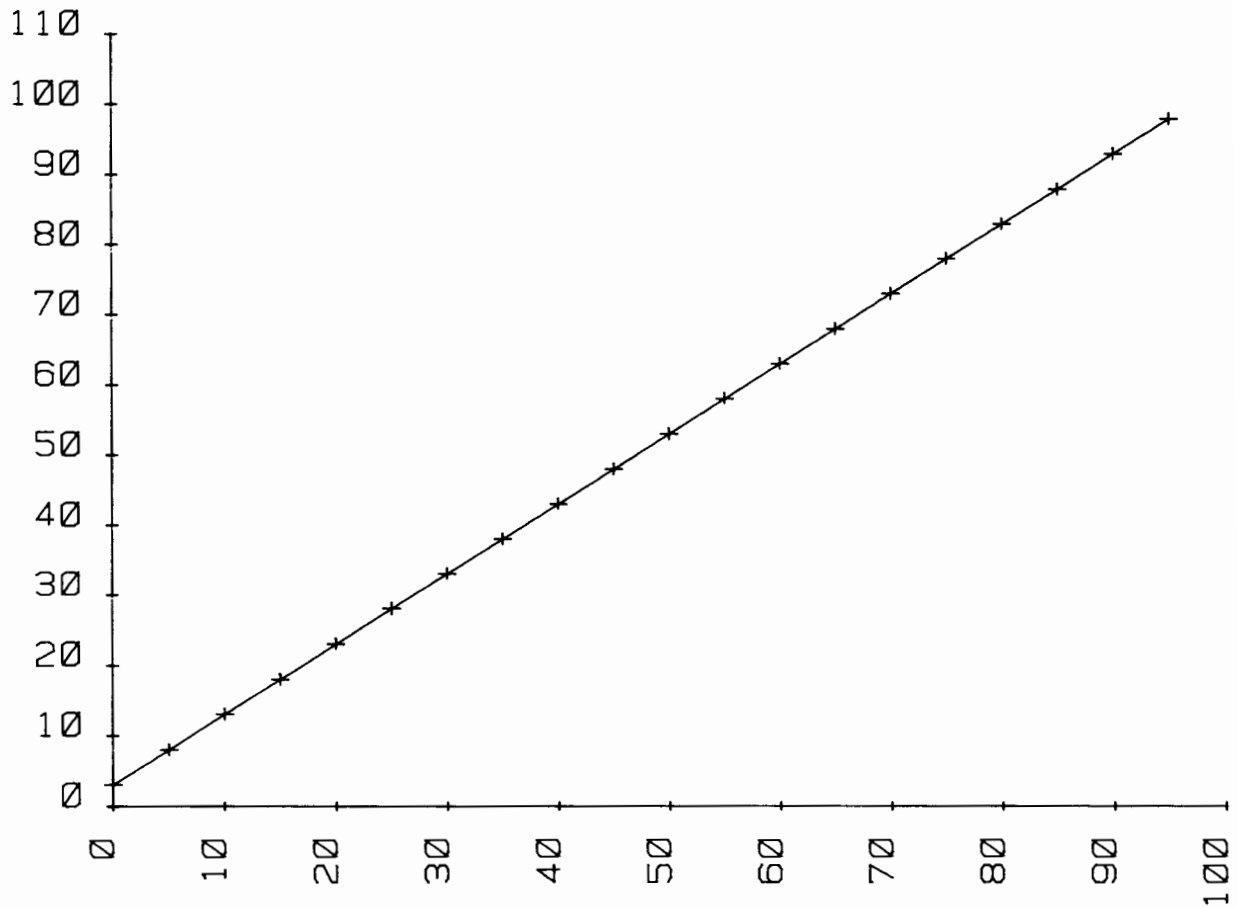


To center the symbol on point C, which represents a plotted point, the pen must be moved to point A. This can be done by executing a `cPlt` statement specifying the parameters `-.33, -.25`. After the symbol is drawn, the pen must be returned from point B to point C to continue plotting the next point. This can be done by executing a `cPlt` statement specifying the parameters `-.67, .25`.

7-42 Plotter Control

The 9872 ROM example below plots the function $Y=X+3$ and labels each point with a +. The value 2 is used in the plot statement in line 5 to lower the pen before continuing the plot. The pen statement raises the pen before the cplt statement in line 6 is executed. Line 6 moves the pen to the lower left corner of the character space where the + is drawn and line 8 moves the pen back to the center point to continue on with the plot.

```
0: pclrisc1 0,
100,0,110
1: fxd 0;csiz 3
2: xax 0,10,0,
100,1
3: yax 0,10,0,
110,1
4: plt 0→X,X+3,1
5: "loop":plt X,
X+3,2;pen
6: cplt -.33,-
.25
7: lbl "+"
8: cplt -.67,.25
9: if (X+5+X)<10
0;goto "loop"
10: end
```



Y = X + 3 Plotted with +'s

Here's an example that uses the label, csize and cplot statements to label a 9862 plot. The csize statement in the first line sets the default values for printing the title of the plot in lines 4, 6, and 8. Lines 5 and 7 are carriage-return line-feeds to position the pen at the beginning of the next line. Lines 10 through 17 label the Y axis and lines 18 through 25 label the X axis. The remainder of the program plots $(\sin X)/X$.

```

0: scl -4.4π,
  4.4π,-.25,1.05;
  csize
1: rad
2: fxd 1
3: plt -4π,.96,
  1
4: lbl "PLOT
  OF"
5: cplt -7,-1
6: lbl "(SIN X)/
  X"
7: cplt -9,-1
8: lbl "[FROM -
  4π TO 4π]"
9: axe 0,0,π/2,
  .1
10: csize 1.5,
  1.7,7/10,0
11: -.2→Y
12: plt 0,Y,1
13: cplt 2,-.3
14: lbl Y
15: Y+.2→Y
16: if Y=0;eto
  15
17: if Y<=1;eto
  12

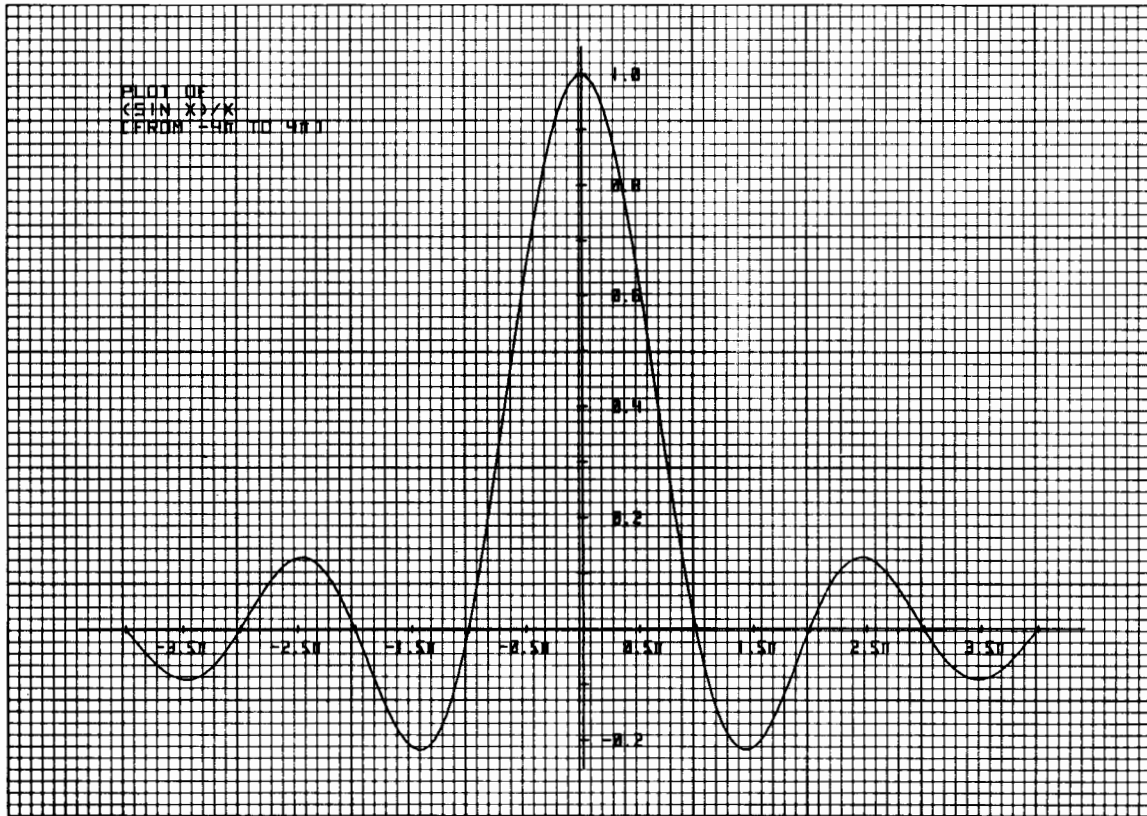
```

```

18: csize 1.5,
  1.7,7/10,0
19: -3.5→X
20: plt Xπ,0,1
21: cplt -2.5,-
  1.3
22: lbl X,"π"
23: X+1→X
24: if X=0;eto
  23
25: if X<=3.5;
  eto 20
26: -4π→Z
27: plt Z,sin(Z)
  /Z
28: Z+π/20→Z
29: if Z<4π+π/
  20;eto 27
30: pen
31: end

```

9862A Cplt Example




9862A Plot of (Sin X)/X using Label, Csize and Cplot

Plotter Typewriter Statement






The plotter typewriter statement sets a manual lettering mode.

ptyp

After the ptyp statement is executed (either in a program, in the live keyboard mode or from the keyboard), you type the desired characters on the calculator keyboard and they are lettered by the plotter. To end the ptyp mode, press the  key once.

The pen can be positioned by plot, iplot or cplot statements before the ptyp mode is established. Once the ptyp mode is established, the four calculator display keys or the four pen movement keys on the plotter's front panel can be used to position the pen for lettering.

The following keys perform these functions while the ptype mode:

Space	
Backspace	
Line Feed	
Inverse Line Feed	
Carriage Return	

When a ptype statement is executed, either within a program, in the live keyboard mode or from the keyboard, the display is not changed or cleared. To indicate that the ptype mode is established, you can display a prompt (a signal within a program) to indicate that characters can be printed in the plotting area. For example -

```
G: "prt "LABEL
  PLOT";ptyp
```

LABEL PLOT

9872 Character Sets

The plotter, when initialized, automatically specifies the ANSI-ASCII character set (Set 0) as both the "standard" set and the "alternate" set. The `PCIR` statement, however, designates the 9825 character set (Set 1) as the standard set and Set 0 as the alternate set. The following keys are used to switch from the standard character set to the alternate set and back.


Select the Standard Set



Select the Alternate Set



Refer to the programming note supplied with the plotter for the description of the available character sets and the statements used to designate which set will be the standard set and which will be the alternate set.

When a statement is executed from a program or from the live keyboard, the program will stop execution as long as the typewriter mode is set. When the mode is ended by pressing , the program will continue from where it was interrupted by the statement. Since the statement leaves the pen raised, it is best not to execute a `ptyp` statement while a plot is in progress.

Letter Statement (9862 ROM only)

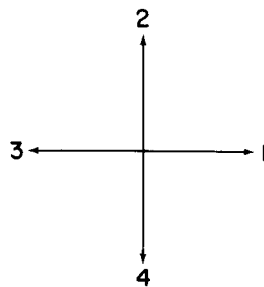
The letter statement is provided for compatibility with the HP 9820/21 Calculators. Unless you're "translating" HP 9820/21 programs for use with the HP 9825A, the csize statement is more versatile for specifying character sizes.

The letter (`ltr`) statement positions the pen for labeling and specifies the size and the direction of the characters to be printed. The pen is raised before movement.

To use the letter statement, two parameters are required - the X and Y positions. A third parameter is optional - HWD indicating Height, Width and Direction of the characters to be printed.

`ltr X coordinate Y coordinate [HWD parameter]`

The HWD parameter is used to set the height, width and direction of the characters. It is a three digit integer - the height and width are each a digit in the range of 1 to 9. The direction is specified by a digit in the range of 1 to 4, each indicating a different direction, as shown below.



Digitizing

The 9872 plotter (or another device responding to HP-GL commands) can be used as a digitizer as well as a plotter since digitizing is basically the inverse of plotting. Instead of sending the coordinates of a point to the plotter and the plotter then moving the pen to that point, you move the pen to a point on the plotter (typically by using the front panel controls) and the plotter sends the coordinates of that point to the computer.

Digitize Statement (9872 ROM only)

```
digitize variable1 [, variable2 [, variable3]
```

The digitize statement enables the digitizer mode. When the digitizer mode is set, the 'ENTER' light on the plotter is lit. You can use the plotter pen-movement controls to position the pen at a point on the platen. Now, pressing ENTER on the plotter front panel sends the coordinates, in scale-statement units, to the calculator where they are assigned to the variables specified by the digitize statement. The coordinate values are assigned to the variables in the following order:

X coordinate value	→	Variable ₁
Y coordinate value	→	Variable ₂
Pen condition (0 = up, 1 = down)	→	Variable ₃ (if specified)

To cancel a digitize statement without entering coordinate values, press the STOP key. If the digitize statement is executed from a program, the STOP key will also stop the program at the end of the line containing the digitize statement.

A special digitizing sight is provided with the plotter which allows you to visually position the pen directly over the point to be digitized. The sight is loaded and stored like a pen.

The following example draws a rectangular figure and then allows you to digitize points around it by moving the pen, via the pen movement controls, to each of the figure's corners. At each corner, you press ENTER to send the coordinates of that corner to the calculator. The program uses the coordinates to calculate the line length or perimeter of the figure.

```

0: pclrisc1 0,
  20,0,16
1: plt 5,4,-1;
  iplt 0,8,2;iplt
  10,0;iplt 0,-
  8;iplt -10,0,-1
2: 0→X→Y→A→B→C→D
  →P
3: dsp "digitize
  the first point,
  please.;"
  dia X,Y
4: X→A→C;Y→B→D
5: dsp "Enter
  the next point,
  please.;"
  dia X,Y
6: P+r((X-A)↑2+
  (Y-B)↑2)→P
7: X→A;Y→B
8: if r((X-C)↑2+
  (Y-D)↑2) < .11
  jmp 2
9: jmp -4
10: prt "Perimet
  er = ";P
11: end

```

This portion of the program scales the plot and uses iplot statements to draw a rectangular figure.

The first digitize statement enables you to enter the coordinates of the first point.

The next digitize statement enables you to enter the rest of the coordinates until the first point is reached again.

The distance formula is used to calculate the length of each digitized line segment and the lengths are summed in P.

Digitizing with the 9872 Plotter

Additional Plotter Control

In addition to the statements described in the previous sections, the plotter has the following additional capabilities accessed via HP-GL commands:

- Pen Control Commands
- Plot Control Commands
- Character Control Commands
- Plotter Configuration and Status Commands

The additional control commands are sent to the plotter as ASCII coded characters (data messages) via write (wrt) statements. Refer to chapter 1 for details concerning write and format statements. The control commands are listed in the HP-GL Programming Manual and operating note supplied with your HP plotter.

9872 Plotter Default Conditions

<code>psc</code>	705 (not changed with <code>pclr</code> or "DF")
<code>scl</code>	Centimeter unit of measure from P1 (not changed with <code>pclr</code> or "DF")
<code>line</code>	Solid line
Line pattern length	4% of the distance from P1 to P2
<code>lim</code>	Total platen area
<code>csiz</code>	1.5, 2, 1, 0
Automatic pen pickup	On
Pen velocity	36 cm/sec
Adaptive pen velocity	Off
Symbol mode	Off
Tic length	.5% of /P1-P2/ length for each half
Standard character set	Set 0 (Set 1 for <code>pclr</code>)
Alternate character set	Set 0
Character slant	0°
Mask value	223,0,0



P1 and P2 are changed only with the initialize command (IN). They are not affected by `pclr` and the default command (DF).

The pen is raised by `pclr`.

The current pen location is moved to the lower right corner with the initialize command (IN) but is unaffected by `pclr` and the default command (DF).

HP 9862A/9872A Plotter ROM Comparison

The following statements have identical parameters and operation with both ROMs.

```
plt
iplt
pen
ofs
lbl
csiz
```

These statements are similar but have minor differences.

9862A ROM	9872A ROM
<p data-bbox="483 905 566 930">ptyp</p> <p data-bbox="285 968 764 1079">Upon exiting the ptype mode, the pen returns to the location that it had when the mode was entered.</p> <p data-bbox="285 1136 764 1251">The shift key reduces the pen movement to 1/10 the increment controlled by the calculator display keys.</p>	<p data-bbox="1040 905 1123 930">ptyp</p> <p data-bbox="842 968 1321 1079">Upon exiting the ptype mode, the pen remains at the location of the next character's origin point.</p> <p data-bbox="842 1136 1321 1251">The shift key does not affect pen movement that is controlled by the calculator display keys.</p>
<p data-bbox="496 1314 553 1339">psc</p> <p data-bbox="285 1371 764 1528">"psc 0" prevents all transmissions to the plotter, but otherwise Plotter ROM statements execute normally. All errors are displayed.</p>	<p data-bbox="1049 1314 1105 1339">psc</p> <p data-bbox="842 1371 1321 1528">"psc 0" prevents execution of Plotter ROM statements thereby preventing error detection for Plotter ROM statements.</p>
<p data-bbox="483 1585 566 1610">cplt</p> <p data-bbox="285 1646 764 1759">Parameters are required by cplot statements. Cplot lifts the pen before moving.</p>	<p data-bbox="1040 1585 1123 1610">cplt</p> <p data-bbox="842 1646 1321 1803">Executing a cplot statement without parameters performs a CR/LF function. Cplot does not lift the pen before moving.</p>

scl

A scale statement must be executed before plotting can be done.

scl

Executing a scale statement without parameters or no scale statement sets a centimeter-unit-of-measure scale with the origin at P1.

These statements are new with the 9872A ROM.

xax	Draws, tic marks and labels a horizontal axis.
yax	Draws, tic marks and labels a vertical axis.
lin	Restricts programmed pen motion to a specific rectangular area.
pen#	Allows programmed selection of the various pens.
line	Specifies one of seven line types for use with plt, ipt, xax and yax. Pattern length can also be selected.
dig	Converts the plotter to a digitizer and converts the coordinates of the pen's location to scale-statement units.
pclr	Resets all programmable functions in the plotter to their default values.

The 9872A Plotter ROM does not use the `axe` or `ltr` statements found in the 9862A Plotter ROM.

Axe and Ltr Conversions

The following provides parameter conversions to replace `axe` with `xax` and `yax` statements and to replace `ltr` with `plt` and `csiz` statements.

```
axe X, Y [, P [, Q]]
```

Two Parameter Conversion

```
xax Y, 0; yax X, 0
```

Three Parameter Conversion

Assume `scl A, M, B, N`.

```
if P = 0; xax Y, 0
```

```
if P > 0; xax Y, Psgn(A - X), X, A; xax Y, Psgn(M - X), X, M; yax X, 0
```

Four Parameter Conversion

Assume `scl A, M, B, N`

```
if P = 0; xax Y, 0
```

```
if P > 0; xax Y, Psgn(A - X), X, A; xax Y, Psgn(M - X), X, M
```

```
if Q = 0; yax X, 0
```

```
if Q > 0; yax X, Qsgn(B - Y), Y, B; yax X, Qsgn(N - Y), Y, N
```

```
ltr X, Y [, Z]
```

Two Parameter Conversion

```
plt X, Y, 1
```

Three Parameter Conversion

Let $D = 90$ for deg

= $\pi/2$ for rad

= 100 for grad

```
plt X, Y, 1
```

```
csiz 0.64int(Z/100), int(Z/100)/(int(Z/10)mod10), 1, (Zmod10 - 1)D
```

This space is for keeping your Interface Operating Notes. Each note is shipped with the appropriately-wired interface card or desktop computer peripheral and describes controlling an HP computer peripheral with your 9825 Desktop Computer. These operating note are currently available:

<u>Operating Note</u>	<u>Part Number</u>	<u>Shipped With</u>
9862A Plotter Op Note	09825-90040	98032A opt 062
9863A Tape Reader Op Note	09825-90041	98032A opt 063
9864A Digitizer Op Note	09825-90042	98032A opt 064
9866A/B Printer Op Note	09825-90043	98032A opt 066
9869A Card Reader Op Note	09825-90044	98032A opt 069
9871A Printer Op Note	09825-90045	98032A opt 071
9883A Tape Reader Op Note	09825-90046	98032A opt 083
9884B Tape Punch Op Note	09825-90047	98032A opt 084
9881A Line Printer Op Note	09825-90048	98032A opt 081
6940A Multiprogrammer Op Note	09825-90049	98032A opt 040
98035A Real Time Clock Op Note	09825-90054	98035A
9875A Tape Memory Op Note	09825-90075	9875A opt 025

This space is for keeping manuals supplied with interface cards and system peripherals. These HP desktop computer interface and peripheral manuals are currently available:

<u>Manual Title</u>	<u>Part Number</u>
Interface Manuals:	
98032A 16-Bit I/O Installation & Service	98032-90000
98033A BCD I/O Installation & Service	98033-90000
98034A HR-IB Installation & Service	98034-90000
98035A Real Time Clock Installation & Service	98035-90000
98036A Serial I/O Installation & Service	98036-90000
98037A A to D Converter Installation & Service	98037-90000
Peripheral Installation and Service Manuals:	
2631A Printer Installation Note	09825-90076
2631A Printer Programming Manual	02631-90903
9862A Plotter Peripheral Manual	09862-90012
9862A Plotter Service Manual	09862-90011
9863A Tape Reader Operating Manual	09863-90000
9864A Diiitizer Peripheral Manual	09864-90000
9866A/B Printer Peripheral Manual	09866-90001
9869A Card Reader Operating & Service	09869-90002
9871A Printer Installation Manual	09871-90035
9871A HP-IB (opt 001) Operating & Service	09871-90001
9872A Plotter Operating & Service	09872-90000
9878A I/O Expander Installation & Service	09878-90000
9881A Printer (2607A) Operators Manual	02607-90005
9883A Tape Reader (2648A) Operating & Service	02748-90032
9884B Tape Punch Operating & Service	09884-90000
9885M/S Disk Drive Installation Manual	09885-90010

This space is for keeping your Interface Operating Notes. Each note is shipped with the appropriately-wired interface card or desktop computer peripheral and describes controlling an HP computer peripheral with your 9825 Desktop Computer. These operating note are currently available:

<u>Operating Note</u>	<u>Part Number</u>	<u>Shipped With</u>
9862A Plotter Op Note	09825-90040	98032A opt 062
9863A Tape Reader Op Note	09825-90041	98032A opt 063
9864A Digitizer Op Note	09825-90042	98032A opt 064
9866A/B Printer Op Note	09825-90043	98032A opt 066
9869A Card Reader Op Note	09825-90044	98032A opt 069
9871A Printer Op Note	09825-90045	98032A opt 071
9883A Tape Reader Op Note	09825-90046	98032A opt 083
9884B Tape Punch Op Note	09825-90047	98032A opt 084
9881A Line Printer Op Note	09825-90048	98032A opt 081
6940A Multiprogrammer Op Note	09825-90049	98032A opt 040
98035A Real Time Clock Op Note	09825-90054	98035A
9875A Tape Memory Op Note	09825-90075	9875A opt 025

Appendix A Table of Contents



Calculator Status Conditions A-3
Extended I/O Status Conditions..... A-4
ASCII Character Codes..... A-5
Octal Keycode Chart..... A-6
ASC Keycode Chart..... A-6
Decimal Keycodes..... A-7
Select Code Settings..... A-8
HP-IB Addresses and Switch Positions A-8

Notes

Appendix **A****Reference Tables****Calculator Status Conditions**

The following table shows the calculator status conditions when the indicated operations are performed. For details about the status condition of modes, variables, etc., see the appropriate section in the manual.

	Erase all or Power on	Operation				Continue after editing	Continue
		Reset	Erase	Run			
Variables	R	X	R	R	X	X	
Flags 0 through 15	R	X	R	R	X	X	
Result	R	X	X	X	X	X	
Binary program	R	X	X	X	X	X	
subroutine return pointers	R	R	R	R	R	X	
Print-all mode	R	R	X	X	X	X	
Verify mode	R	R	X	X	X	X	
Live keyboard mode	R	R	X	X	X	X	
Secure mode	R	X	R	X	X	X	
Cassette select code	R	R	X	X	X	X	
Cassett track	R	R	X	X	X	X	
Angular units for trig functions	R	R	X	X	X	X	
Fixed/Float setting	R	R	X	X	X	X	
Random number seed	R	R	X	X	X	X	
Trace mode	R	R	X	X	X	X	

R = Restored to power-on value

X = Unchanged

Extended I/O Status Conditions

The following table shows conditions for various Extended I/O operations and modes. Notice that the Erase, Erase All-Power on, and Run columns from the previous table are combined into one column here. R = restored to power-on state; X = unchanged.

Extended I/O ROM Operation or Mode	Power On Erase Erase All Run	Calculator Operation		
		Reset	Continue (after edit)	Continue (after Stop)
Conversion and parity tables	R	X	X	X
Binary mode (reset to decimal)	R	X	X	X
I/O buffer area	R	X	X	X
Service name list	R	X	X	X
Equate name list	R	X	X	X
Buffer select code for tfr	R	R	R	X
Interrupt parameters	R	R	R	X
Error recovery routine	R	R	R	X
Timeout routine	R	X	X	X

ASCII Character Codes



ASCII Char.	EQUIVALENT FORMS		
	Binary	Octal	Decimal
NULL	00000000	000	0
SOH	00000001	001	1
STX	00000010	002	2
ETX	00000011	003	3
EOT	00000100	004	4
ENQ	00000101	005	5
ACK	00000110	006	6
BELL	00000111	007	7
BS	00001000	010	8
HT	00001001	011	9
LF	00001010	012	10
VT	00001011	013	11
FF	00001100	014	12
CR	00001101	015	13
SO	00001110	016	14
SI	00001111	017	15
DLE	00010000	020	16
DC1	00010001	021	17
DC2	00010010	022	18
DC3	00010011	023	19
DC4	00010100	024	20
NAK	00010101	025	21
SYNC	00010110	026	22
ETB	00010111	027	23
CAN	00011000	030	24
EM	00011001	031	25
SUB	00011010	032	26
ESC	00011011	033	27
FS	00011100	034	28
GS	00011101	035	29
RS	00011110	036	30
US	00011111	037	31

ASCII Char.	EQUIVALENT FORMS		
	Binary	Octal	Decimal
space	00100000	040	32
!	00100001	041	33
"	00100010	042	34
#	00100011	043	35
\$	00100100	044	36
%	00100101	045	37
&	00100110	046	38
'	00100111	047	39
(00101000	050	40
)	00101001	051	41
*	00101010	052	42
+	00101011	053	43
,	00101100	054	44
-	00101101	055	45
.	00101110	056	46
/	00101111	057	47
0	00110000	060	48
1	00110001	061	49
2	00110010	062	50
3	00110011	063	51
4	00110100	064	52
5	00110101	065	53
6	00110110	066	54
7	00110111	067	55
8	00111000	070	56
9	00111001	071	57
:	00111010	072	58
;	00111011	073	59
<	00111100	074	60
=	00111101	075	61
>	00111110	076	62
?	00111111	077	63

ASCII Char.	EQUIVALENT FORMS		
	Binary	Octal	Decimal
@	01000000	100	64
A	01000001	101	65
B	01000010	102	66
C	01000011	103	67
D	01000100	104	68
E	01000101	105	69
F	01000110	106	70
G	01000111	107	71
H	01001000	110	72
I	01001001	111	73
J	01001010	112	74
K	01001011	113	75
L	01001100	114	76
M	01001101	115	77
N	01001110	116	78
O	01001111	117	79
P	01010000	120	80
Q	01010001	121	81
R	01010010	122	82
S	01010011	123	83
T	01010100	124	84
U	01010101	125	85
V	01010110	126	86
W	01010111	127	87
X	01011000	130	88
Y	01011001	131	89
Z	01011010	132	90
[01011011	133	91
\	01011100	134	92
]	01011101	135	93
^	01011110	136	94
_	01011111	137	95

ASCII Char.	EQUIVALENT FORMS		
	Binary	Octal	Decimal
`	01100000	140	96
a	01100001	141	97
b	01100010	142	98
c	01100011	143	99
d	01100100	144	100
e	01100101	145	101
f	01100110	146	102
g	01100111	147	103
h	01101000	150	104
i	01101001	151	105
j	01101010	152	106
k	01101011	153	107
l	01101100	154	108
m	01101101	155	109
n	01101110	156	110
o	01101111	157	111
p	01110000	160	112
q	01110001	161	113
r	01110010	162	114
s	01110011	163	115
t	01110100	164	116
u	01110101	165	117
v	01110110	166	118
w	01110111	167	119
x	01111000	170	120
y	01111001	171	121
z	01111010	172	122
{	01111011	173	123
	01111100	174	124
}	01111101	175	125
~	01111110	176	126
DEL	01111111	177	127

Octal Keycode Chart*

SYSTEM COMMANDS				DISPLAY		LINE				301	302	303	304	305	306		
223	202	230		220	221	211	210	213	234	f ₀	f ₁	f ₂	f ₃	f ₄	f ₅		
RESET	PRT ALL	REWIND	STEP	←	→	DELETE	INSERT	RECALL	FETCH	101	102	103	104	105	106		
23	2	30		20	21	11	10	13	34	307	310	311	312	313	314		
ERASE	LOAD	RECORD	LIST	←	→	CHARACTER				f ₆	f ₇	f ₈	f ₉	f ₁₀	f ₁₁		
235	237	236	233	216	217	224	225	227	228	107	110	111	112	113	114		
35	37	36	33	16	17	BACK	FWD	DELETE	INS/RPL	107	110	111	112	113	114		
261	262	263	264	265	266	267	270	271	260	373	214	222	340	250	251	257	
!	"	#	\$	%	&	@			'		π	RUN	CLEAR	ENTER EXP	()	/
61	62	63	64	65	66	67	70	71	60	173	14	22	140	50	51	57	
301	367	345	362	364	371	365	351	357	360	380	14	375	325	326	327	252	
Q	W	E	R	T	Y	U	I	O	P	STORE	15	→	7	8	9	*	
181	187	145	182	164	171	185	151	157	180	15	175	125	126	127	52		
SHIFT LOCK	A	S	D	F	G	H	J	K	L	;	↵	=	4	5	6	-	
141	183	144	148	147	150	152	153	154	73	136	275	322	323	324	255		
372	370	343	366	342	356	355	254	256	277	336	75	122	123	124	55		
SHIFT	Z	X	C	V	B	N	M	<	>	:	?	SHIFT	1	2	3	+	
172	170	143	186	142	156	155	54	56	77	136	212	317	320	320	253		
240										201	231	12	118	130	131	7	
										STOP	CONTINUE	12	118	130	131	7	
										1	31	12	118	130	131	7	

ASC Keycode Chart*

SYSTEM COMMANDS				DISPLAY		LINE				140	141	142	143	144	145	
19	2	24		16	17	9	8	11	28	f ₀	f ₁	f ₂	f ₃	f ₄	f ₅	
RESET	PRT ALL	REWIND	STEP	←	→	DELETE	INSERT	RECALL	FETCH	128	129	130	131	132	133	
19	2	24		16	17	9	8	11	28	148	147	148	149	150	151	
ERASE	LOAD	RECORD	LIST	←	→	CHARACTER				f ₆	f ₇	f ₈	f ₉	f ₁₀	f ₁₁	
29	31	30	27	14	15	20	21	23	22	134	135	136	137	136	139	
29	31	30	27	14	15	20	21	23	22	134	135	136	137	136	139	
33	34	35	36	37	38	64	91	93	39	124	12	18	95	40	41	47
!	"	#	\$	%	&	@			'	π	RUN	CLEAR	ENTER EXP	()	/
49	50	51	52	53	54	55	56	57	48	123	12	18	101	40	41	47
81	87	89	82	84	89	85	73	79	80	123	13	125	55	56	57	42
Q	W	E	R	T	Y	U	I	O	P	STORE	13	→	7	8	9	*
113	119	101	114	118	121	117	105	111	112	13	125	55	56	57	42	
SHIFT LOCK	A	S	D	F	G	H	J	K	L	;	↵	=	4	5	6	-
65	63	66	70	71	72	74	75	76	59	92	81	52	53	54	45	
90	86	87	86	86	86	78	77	80	82	56	61	52	53	54	45	
SHIFT	Z	X	C	V	B	N	M	<	>	:	?	SHIFT	1	2	3	+
122	120	99	116	98	110	109	44	46	83	94	10	49	50	51	43	
32										1	25	10	48	48	44	7
										STOP	CONTINUE	10	48	48	44	7
										1	25	10	48	48	44	7

* Unshifted code shown below key; shifted code shown above key.

Decimal Key Codes

KEY	CODE		KEY	CODE		KEY	CODE	
	UNSHIFT	SHIFT		UNSHIFT	SHIFT		UNSHIFT	SHIFT
A	97	225	0	48	176	f10	75	203
B	98	226	π	123	251	f11	76	204
C	99	227	0	78	206	PRT ALL	19	147
D	100	228	1	79	207	REWIND	2	130
E	101	229	2	80	208	STEP	24	152
F	102	230	3	81	209	←	16	144
G	103	231	4	82	210	→	17	145
H	104	232	5	83	211	DELETE	9	137
I	105	233	6	84	212	INSERT	8	136
J	106	234	7	85	213	RECALL	11	139
K	107	235	8	86	214	FETCH	28	156
L	108	236	9	87	215	ERASE	29	157
M	109	237	.	88	216	LOAD	31	159
N	110	238	.	89	217	RECORD	30	158
O	111	239	=	61	189	LIST	27	155
P	112	240	→	125	253	←	14	142
Q	113	241	CLEAR	18	146	→	15	143
R	114	242	ENTER STOP	96	224	BACK	20	148
S	115	243	(40	168	FWO	21	149
T	116	244)	41	169	DELETE	23	151
U	117	245	/	47	175	INS/RPL	22	150
V	118	246	*	42	170	RUN	12	140
W	119	247	-	45	173	STORE	13	141
X	120	248	+	43	171	STOP	1	129
Y	121	249	RESULT	7	135	CONTINUE	25	153
Z	122	250	f0	65	193	↑	10	138
!	49	177	f1	66	194	↓	59	187
"	50	178	f2	67	195	↕	94	222
#	51	179	f3	68	196	<	44	172
\$	52	180	f4	69	197	>	46	174
%	53	181	f5	70	198	.	63	191
&	54	182	f6	71	199	:	63	191
@	55	183	f7	72	200	?	63	191
	56	184	f8	73	201	space bar	32	160
	57	185	f9	74	202			

Factory Set Select Codes

Select Code	Assignment	HP Peripheral Device
0	Calculator Keyboard and Display	
1	Calculator Tape Drive	
2	Paper Tape Punch	9884A*,98032A Interface
3	Paper Tape Reader	9883A*,9863A*,98033A Interface
4	Digitizer	9864A*
5	Plotter	9862A*
6	Printer	9866B*,9871A*
7	HP Interface Bus	98034A Interface
8	Mass Memory	9885A*
9 through 15	Unassigned	Special or Duplicate Peripherals
16	Calculator Printer	

*These peripherals should be ordered with Option 025.

HP-IB Address Switch Positions

Address Characters		Address Switch Settings					Address Codes	
Listen	Talk	(5)	(4)	(3)	(2)	(1)	decimal	octal
SP	@	0	0	0	0	0	0	0
!	A	0	0	0	0	1	1	1
"	B	0	0	0	1	0	2	2
#	C	0	0	0	1	1	3	3
\$	D	0	0	1	0	0	4	4
%	E	0	0	1	0	1	5	5
&	F	0	0	1	1	0	6	6
'	G	0	0	1	1	1	7	7
(H	0	1	0	0	0	8	10
)	I	0	1	0	0	1	9	11
*	J	0	1	0	1	0	10	12
+	K	0	1	0	1	1	11	13
,	L	0	1	1	0	0	12	14
-	M	0	1	1	0	1	13	15
.	N	0	1	1	1	0	14	16
/	O	0	1	1	1	1	15	17
0	P	1	0	0	0	0	16	20
1	Q	1	0	0	0	1	17	21
2	R	1	0	0	1	0	18	22
3	S	1	0	0	1	1	19	23
4	T	1	0	1	0	0	20	24
5	U	1	0	1	0	1	21	25
6	V	1	0	1	1	0	22	26
7	W	1	0	1	1	1	23	27
8	X	1	1	0	0	0	24	30
9	Y	1	1	0	0	1	25	31
:	Z	1	1	0	1	0	26	32
;	[1	1	0	1	1	27	33
<	/	1	1	1	0	0	28	34
=]	1	1	1	0	1	29	35
>	^	1	1	1	1	0	30	36

Appendix B

HPL Syntax



Introduction

The following pages are a compilation of all current 9825 HPL syntax. More information on each operation can be found by referring to the indicated manual and page. The manual titles are abbreviated here:

- D Disk Programming, 09825-90220 (or 09885-90000).
- I/O I/O Control Reference.
- M Matrix Programming.
- O&P Operating & Programming Reference.

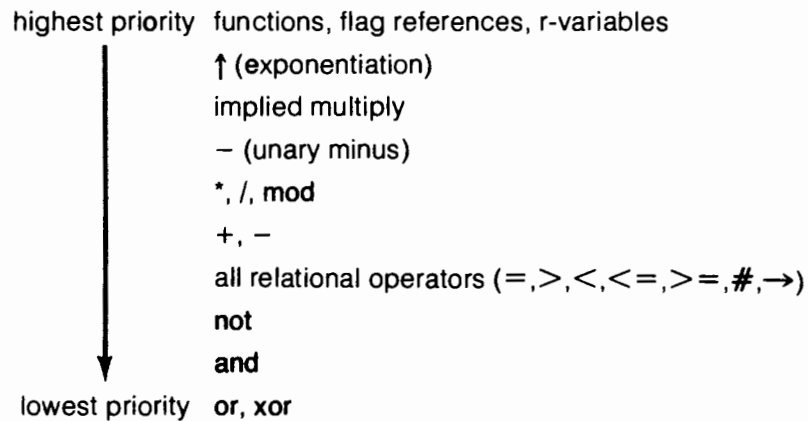
The HPL programming language utilizes four basic types of syntax constructions: **statements**, **functions**, **operators** and **commands**. Operators, such as + and mod, are used with numbers and variable names to construct **expressions** (like A+5). Expressions can be included in many statements and executed from the keyboard. Each statement can also be preceded by a line number and stored as a program line (like 10: prt A). Most functions can include expressions, and can be executed from the keyboard. Functions can also be treated as expressions when constructing a statement (like prt sinA). Commands are operator aids that can only be executed from the keyboard; they're not programmable.

Operators

The available operators are summarized here. For more details see the HPL Programming chapter, page 3-19.

<u>Arithmetic</u>			<u>Relational</u>	
+	Add		→	Assign
-	Subtract, unary -		>	Greater than
*	Multiply		<	Less than
/	Divide		>= or =>	Greater than or equal to
↑	Exponentiate		<= or =<	Less than or equal to
mod	Modulus		# or < > or > <	Not equal to
<u>Logical</u>			<u>String</u>	
and			&	Concatenation
ior	inclusive OR			
xor	exclusive OR			
not				

Math Hierarchy



Operators of the same level in an expression are executed from left to right. Any operations within parentheses, however, are performed first. For more details, see page 3-18 in your Operating and Programming Reference.

Syntax Conventions

These terms and conventions are used in the following listing:

bold type — all key words and characters appearing in bold type must appear exactly as shown. These items are shown in dot matrix in the referenced manuals.

[] — elements enclosed in brackets (not key characters or parentheses) are optional.

... — an ellipsis indicates that the preceding parameter or sequence in the syntax can be repeated.

variable name — a numeric or string variable name (like A or R5 or A\$). Subscripts are allowed (like A [7]).

array name — an array variable name, with or without subscripts.

string variable — a string variable name (like A\$ or B\$ [1,4]).

string — either a string variable or text within quotes ("text").

line number — an expression from 1 through 999 referring to a program line.

line label — a unique name assigned to a program line. It's enclosed in quotes, follows the line number, and is followed by a colon. For example: 5: "print": ...

expression — a logical combination of numeric variable names, constants, operators and functions (including user-defined functions) grouped within parentheses as needed. The evaluated expression yields a numeric result.

constant — a fixed number within the computer's range, like 2.23467.

character — a letter, number or symbol.

item list — a series of constants, expressions and/or strings separated by commas, for example: prt 5,A,"was",A+7

subscripts — numbers within brackets which are attached to variable names to designate a particular variable element or boundary. For example: A [10,5] or B\$ [1,10]

file number — an expression indicating the tape or disk file.

file name — a string indicating the disk file name.

select code — an expression indicating the device's interface select code setting (an integer from 0 through 16). For example: wrt 6

These select codes are assigned to internal devices:

- 0 Keyboard.
- 1 Tape drive.
- 16 Printer.

device address — a two-digit number appended to the select code, indicating a device's HP-IB address. Device address range is from 01 through 31. For example: wrt 711 outputs to device 11 via the HP-IB interface set to select code 7.

format no. — a number from .1 through .9 appended to the select code to reference a corresponding fmt statement. For example: wrt 7.3 references fmt 3.

return variable — a simple numeric variable name (A or R4) where information is stored after the operation.

flag no. — an expression from 1 through 15 indicating a programmable flag.

A

abs expression

Returns the absolute value of the expression. O&P, 3-22.

acs expression

Returns the principal value of the arccosine of the expression in the current angular units. O&P, 3-25.

add (expression , expression)

Returns the sum of the expressions, added in the current numeric mode, decimal (mdec) or octal (moct). I/O, 3-15.

aprt array variable [, array variable [, ...]]

Prints the specified array's elements on the internal printer. M, 8.

ara array variable₁ [$\left\{ \begin{array}{c} + \\ - \\ * \\ / \end{array} \right\}$ array variable₂] → array variable₃

Performs the arithmetic operation, element by element, on arrays 1 and 2. The result is stored in array 3. (Example: ara A+B→C). Arithmetic operations can be performed on arrays in place (ara A+B→A), arrays can be copied (ara A→B) and implied multiplication is allowed (ara AB→C). M, 11.

asc expression

Returns the ASCII equivalent of the specified 9825 keycode. O&P, 7-25.

asgn file name , file number [, drive number [, return variable]]

Assigns a number (1 through 10) to an existing disk file name and indicates optional drive number and a return variable (values below). D, 3-5 (or 36).

0	File available and assigned.	5	Memory file.
1	File doesn't exist.	6	Binary program file.
2	Program file.	7	File type not defined.
3	Special function key file.	8	File number out of range.
4	File not defined by 9825.	9	Data file, but logical records not 256 bytes long (98228A ROM only).

asn expression

Returns the principal value of the arcsine of the expression in the current angular units. O&P, 3-26.

expression → variable name₁ [→ variable name₂ [→ ...]]

Assigns the value of the expression to the variable(s). O&P, 3-19.

atn expression

Returns the principal value of the arctangent of the expression in the current angular units. O&P, 3-26.

avd

Disables automatic tape verification. O&P, 5-24.

ave

Enables automatic tape verification (default setting). O&P, 5-25.

avm

Returns the size (bytes) of unused read/write memory. O&P, 4-27.

axe X coordinate , Y coordinate [, X tic [, Y tic]]

Draws axes through the X,Y point, drawing optional tic marks at X tic and Y tic intervals. 9862 Plotter ROM only. I/O, 7-18.

B

band (expression , expression)

Returns the 16-bit result of ANDing the expressions. I/O, 3-12.

beep

Sounds the computer's beeper. O&P, 3-16.

bit (expression₁ , expression₂)

Returns the binary value of the bit position in expression 2 indicated by expression 1. I/O, 3-15.

boot

Loads 98217A Disk ROM bootstraps from a disk tape to an initialized disk. D, 1-8 (or 67).

bred (buffer name)

Returns the contents of the specified, active, interrupt buffer. O&P, 7-10.

buf "name" [, buffer size or string variable , buffer type]

Sets up and names a data buffer of either type read/write (no type specified) or the specified type (see below). I/O, 6-6.

Buffer Type	Word	Byte
interrupt	0	1
fast read/write	2	3
DMA	4	—

C

cap (string)

Returns an equivalent string of uppercase characters. O&P, 6-24.

cat [select code or buffer name]

Prints a catalog of files on the specified disk or default drive. File types listed below. D, 1-16 (or 20).

B Binary program file.	M Memory file.
D Data file.	O Other file (not created via 9825).
K Special function keys file.	P Program file.

cfg [flag no. [, ...]]

Clears either all 15 program flags or only the specified flags. O&P, 3-29.

chain file name [, 1st line number [, 2nd line number]]

Loads a program from the specified disk file. Same optional line numbers as get. D, 2-7 (or 25).

char (expression)

Returns the ASCII equivalent character. O&P, 6-20.



cli select code

Sends the abort message to all devices on the HP-IB, I/O, 2-27.

cli 'name' [(variable₁ [, variable₂ [, ...]])]]

Calls the subroutine having the specified label, passing the value of any optional variables as pass-parameters. O&P, 4-10.

cln

Returns the current program line number. O&P, 7-28.

clr select code

Sends the clear message, either the all devices or to only a selected device by including the device address in the select code. I/O, 2-17.

cmd select code , "address parameters" [, "string"]

cmd "device name(s)" or select code [, "string"]

Sends the string of data characters to the specified HP-IB device. I/O, 2-31.

cmf [flag no. [, ...]]

Complements either all 15 program flags or only the specified flags. O&P, 3-29.

cmp (expression)

Returns the 16-bit binary one's complement of the expression. I/O, 3-13.

cont [line number or "line label"]

This command continues program execution, either from the current point or from the specified point. O&P, 2-24.

conv [expression₁ , expression₁ [, expression₂ , expression₂] ...]

Sets up a conversion table (up to 10 sets of expressions) referenced by red and wrt statements. Each expression represents an ASCII character. conv (no parameters) cancels any existing table. I/O, 1-23.

copy [source drive number [, select code] ,] "to"

[, destination drive number [, select code]]

Duplicates the contents of the source disk to the destination disk. Disks must be the same type, either single-sided or double-sided. D, 4-7 (or 60).

dirc

Copies the spare 9885 disk directory (default drive) to the main directory. 98217A ROM only. D, 4-16 (or 65).

drive unit no. [, select code]

Sets the default unit (0 through 3) and, optionally, the select code for disk drives. Default is 0,8 for 98217A ROM and 0,707 for 98228A ROM. D, 1-14 (or 17).

drnd (expression , expression)

Returns the value of the first expression, rounded to the number of digits indicated by the second expression. O&P, 3-22.

dsp item list

Displays the items listed. To display quotes use double quotes within the string (e.g., 1: dsp "Display""test""in quotes."). O&P, 3-12.

dto (expression)

Returns the octal equivalent of the decimal value expressed. I/O, 3-12

dtrk

Dumps a bad 9885 track during the disk error recovery routine. 98217A ROM only. D, 4-15 (or 65).

dtype

Returns a code indicating the type of drive, disk and data format at the default disk address. 98228A ROM only. D, 1-15. Return values are:

- 0 Unable to access default disk controller.
- 1 Drive door is open or drive not present.
- 2 Drive door closed, but door was opened since last disk operation. File pointers are cleared.
- 3 9895 drive, single-sided disk, HP format.
- 4 9895 drive, double-sided disk, HP format.
- 5 9895 drive, single-sided disk, unknown format.
- 6 9895 drive, double-sided disk, unknown format.
- 7 9895 drive, single-sided disk, IBM 3740 format.
- 8 9885 drive, single-sided disk.

dump [file name , tape file name] [, expression]

Transfers the contents of the default disk to a tape cartridge. The optional file names indicate to only dump a specified file. The expression can be 1 or 10, indicating the number of disk records to put in each tape file. A positive expression automatically marks the tape. A negative expression suppresses marking the tape. D, 4-12 (or 62).

E

eir select code [, byte]

Enables an interrupt from the specified select code. Specifying byte = 0 disables the interrupt. I/O, 5-6.

end

Halts program execution and sets the program counter to 0. O&P, 3-17.

enp ["prompt",] string variable

Enters and prints data entered from the keyboard. O&P, 3-15.

ent ["prompt",] variable name

Enters data from the keyboard. O&P, 3-13.

eol code [, [, ...]] [, - delay in milliseconds]

Specifies up to seven optional ASCII characters for an end-of-line sequence for wrt operations (replaces CR/LFs). The optional delay occurs after the last eol character in the sequence. O&P, 7-12.

eor (expression , expression)

Returns the 16-bit binary result of the exclusive ORing of the expressions. I/O, 3-13.

equ "name₁" , "string₁" [, "name₂" , "string₂" [, ...]]

Equates the ASCII character string with the name, for use with cmd. I/O, 2-33.

erase [letter or key]

Erases either all programs and variables or the specified areas listed below. O&P, 2-26.

- a Erase entire memory.
- k Erase all special function keys.
- v Erase all variables and flags.
- fn Erase specified key definitions.

ert file number

Erases the current tape track, beginning with the specified file. O&P, 5-15.

exp (expression)

Returns e (2.71828...) raised the expressed power. O&P, 3-24.

F

fdf file number

Positions the tape at the specified file on the current track. O&P, 5-9.

fetch [line number or key]

Displays the specified program line or special function key definition. O&P, 2-27.

files "file name₁" [: unit no.] [, "file name₂" [: unit no.] [, ...]

Assigns names up to 10 disk files. Substituting an * for a file name allows an asgn statement to assign a file name via a string variable. D, 3-3 (or 34).

flg (flag no.)

Returns flag status: 1 = set; 0 = clear. O&P, 3-30.

flt expression

Sets floating point notation; from 0 through 11 places allowed. O&P, 3-10.

fmt [format no. ,] [spec₁ [, spec₂ ...]]

Sets up a list of format specs for red and wrt operations. Format number can be from 0 through 9. Format specs are listed below. Omitting specs cancels specified format. Omitting format no. sets format 0. A repeat factor can precede each spec. I/O, 1-8.

b	Single-character binary output.	x	Blank space.
cw	String character data.	z	Suppresses auto CR/LF.
ew.d	Exponential format.	/	Outputs CR/LF.
fw.d	Fixed-point.	"text"	Outputs text.
fzw.d	Fixed point with leading zeros.		

w = field width.

d = number of digits to right of decimal point.

for simple variable = initial value **to** value [**by** step value]

Defines start of a for-next loop. O&P, 4-3.

frc (expression)

Returns the fractional part of the expression. O&P, 3-22.

fti (expression)

Rounds and changes the expression to integer precision. The result can be stored in a two-character field. O&P, 4-26.

fts (expression)

Changes the expression to split precision for storage in a four-character field. O&P, 4-20.

fxd expression

Sets the fixed-point format; from 0 through 11 places are allowed. O&P, 3-9.

G

get file name [, 1st line no. [, 2nd line no.]]

Loads the program from the specified disk file. The lines are stored, beginning either at line 0 or at the optional 1st line number. The optional 2nd line number indicates where program execution should begin. D, 2-4 (or 23).

getb file name

Loads the specified disk binary program file. D, 2-11 (or 64).

getk file name

Loads the special function keys disk file. D, 2-9 (or 29).

getm file name

Loads the specified disk memory file. D, 2-10 (or 57).

grad

Sets the grads units for angular calculations. O&P, 3-25.

gsb line number or line label

Branches program execution to the specified subroutine. O&P, 3-34.

gsb + or - no. of lines

Branches to the subroutine beginning the number of lines relative to the current line. O&P, 3-34.

gto line number or line label

Sends program execution to the specified line. O&P, 3-31.

gto + or - no. of lines

Sends execution to specified line relative to the current line. O&P, 3-31.

|

idf file number [, file type [, current size [, absolute-size or [, track]]]]]

Returns info on the current tape file. See tlist for file types. O&P, 5-7.

idn array name [, array name [, ...]]

Creates identity (square) matrices. All elements are 0 except major diagonal elements which are 1. M, 22.

if expression₁ = expression₂

If the equation is true, the rest of the line is executed. If false, execution immediately branches to the next line. Any relational operator can be used (< , # , > = , etc.). When both expressions are strings, the characters are compared using ASCII values. O&P, 3-36.

ina array variable [: number or simple variable]

[, array variable [: number or simple variable] ...]

Initializes each element of the array to the specified value (number or variable). Omitting the value initializes each element to 0. M, 8.

init

Runs the 9885 disk initialization routine and loads bootstraps. 98217A ROM only. D, 4-2 (or 90).

init drive number , select code [, interleave factor]

Initializes disks in either 9885 or 9895 drive. 98228A ROM only. The interleave can be an integer from 1 thru 29. D, 4-3.

**int** (expression)

Returns the integer value of the expression. O&P, 3-22.

inv array variable₁ → array variable₂ [, simple variable]

Stores the inverse matrix of array 1 in array 2. If the simple variable is specified, the determinant of array 1 is returned. M, 24.

iof select code

Returns interface flag state: 0 if peripheral busy; 1 if ready. I/O, 4-12.

ior (expression , expression)

Returns the 16-bit result of the inclusive OR operation on the expression. I/O, 3-13.

ios select code

Returns interface status: 0 if in error condition; 1 if operational. I/O, 4-12.

iplt X increment , Y increment [, expression]

Moves the pen the number of X and Y units from its current position. The expression is for pen control; see plt. I/O, 7-29.

iret

Ends an interrupt service routine and returns to main program. I/O, 5-7.

itf (string variable)

Returns a full-precision number from the packed, integer-precision number (a two-character string). O&P, 7-26.

J

jmp expression

Jumps program execution the relative number of lines forward (+ expression) or back (– expression). jmp 0 returns execution to the beginning of the current line. O&P, 3-33.

K

key

Returns the earliest, unprocessed keycode in the keyboard buffer. 0 indicates no keycodes in the buffer. O&P, 7-8.

kill file name

Purges the specified disk file from the default disk. D, 1-18 (or 27).

killall

Purges all disk user files. 98217A ROM only. D, 1-18 (or 67).

killall drive number , select code

Purges all user files from the specified disk. 98228A ROM only. D, 1-18.

kret

Returns execution to the main program after the key buffer is emptied. O&P, 7-9.

L

lbl expression or "string" [, expression or "string" [, ...]]

Prints characters on the plotter. I/O, 7-36.

lcl select code

Sends the local message to all HP-IB devices or, if the select code includes a device address, sends a clear lockout/local message. I/O, 2-20.

ldb file number

Loads a binary program from the specified tape file. O&P, 5-23.

ldf [file number [, line number₁ [, line number₂]]]

Loads the specified tape file into the appropriate area of memory. The optional line numbers indicate where to start loading (line number 1) and continuing (line number 2) a program. Omitting the file number loads file 0. O&P, 5-18.

ldf [file number [, data list]]

Loads data from the specified tape file into the listed variables. O&P, 5-21.

ldk [file number]

Loads the special function key file into memory. Omitting the file number loads tape file 0. O&P, 5-22.

ldp [file number [, line number₁ [, line number₂]]]

Loads a program from either file 0 (file number omitted) or the specified file. The optional line numbers indicate where to start loading (line number 1) and were to start running (line number 2). O&P, 5-18.

len (string variable)

Returns the character length of the string. O&P, 6-14.

lim [X lower left , X upper right , Y lower left , Y upper right]

Restricts plotter pen movement to the stated bounds in user units. If bounds are omitted, movement is limited to the mechanical limits. 9872 Plotter ROM only. I/O, 7-34.


line [pattern number [, pattern length]]

Specifies the type of line plotted with plt, iplt, xax and yax. 9872 patterns are listed below. Pattern length is percentage of the total line length; default is 4%, 9872 Plotter ROM only. I/O, 7-32.

- | | |
|---|--|
| <p>1 —</p> <p>2 — - - - - -</p> <p>3 — _ _ _ _ _</p> <p>4 — — — — . — — . — — . — —</p> | <p>5 — — — — — — — — — —</p> <p>6 — — — — — — — — — —</p> <p>omit number — — — — — — — — — —</p> |
|---|--|

list [# select code] [line number [, line number]]

Lists the entire program on the internal printer (no parameters) or lists the program to the specified select code. The line numbers indicate starting and ending lines for the listing. O&P, 3-39 and I/O, 1-23.

list  or **listk**

Lists the special function key definition (list  or all definitions (list k). O&P, 3-39.

lkd

Disables live keyboard mode. O&P, 2-32.

lke

Enables live keyboard mode. O&P, 2-32.

llo select code

Sends the local lockout message to all HP-IB devices. I/O, 2-19.

ln (expression)

Returns the natural log (\log_e) of the expression. O&P, 3-24.

load [disk file name , tape file number]

Loads files previously dumped to a tape back onto the disk. Omitting all parameters loads the entire dump back onto the disk. Including parameters loads only selected data files back onto the disk. D, 4-13 (or 63).

log (expression)

Returns the common log (\log_{10}) of the expression. O&P, 3-24.

ltr X coordinate , Y coordinate [, HWD]

Moves the 9862 plotter pen to the specified point and specifies dimensions for lettering. H and W can be from 1 through 9. D is lettering direction and can be from 1 through 4. 9862 Plotter ROM only. I/O, 7-47.

ltrk

Returns corrected data to a reinitialized track during disk error-recovery routine. 98217A ROM only. D, 4-15 (or 65).

M

mat array variable₁* array variable₂ → array variable₃

Array multiplication (arrays must have correct dimensions). M, 19.

max (expression [, expression [, ...]])

Returns the largest value in the list. O&P, 3-22.

mdec

Sets the decimal mode (default) for binary operations. I/O, 3-11.

min (expression [, expression [, ...]])

Returns the smallest value in the list. O&P, 3-22.

moct

Sets the octal mode for binary operations. I/O, 3-11.

mrk number of files , file size [, return variable]

Marks the number of files, beginning at the tape's current position. The last file number marked is returned in the optional return variable. O&P, 5-10.

N

nal

Returns the last program line number plus one; used with store to store strings. O&P, 7-24.

next simple variable

Terminates for-next loop and tests for loop completion. O&P, 4-3.

nor [line number [, line number]]

Clears the master program flag, either while executing all lines (omit all parameters) or only for the specified line numbers. O&P, 3-44.

num ("character" or substring)

Returns the ASCII-decimal value of the character. O&P, 6-21.

O

ofs X coordinate , Y coordinate

Offsets the origin (0,0) to point X,Y. I/O, 7-27.

on end file number , line specifier

Enables a branch to the specified line or label when a disk EOF or EOR mark is encountered during read and write operations. D, 3-19 (or 50).

on err "line label"

Enables an error-trapping routine. The program branches to the label and the erl, ern and rom functions are assigned values when an error occurs. I/O, 4-4.

on key ["line label" [, flag no.]]

Enables a keyboard interrupt routine. The program branches to the label and optionally sets the flag when the keyboard buffer overflows. Omitting all parameters disables the keyboard interrupt. O&P, 7-6.

oni select code , "label"

References an interrupt service routine associated with the peripheral's select code. I/O, 5-5.

open file name , number of records

Creates a disk data file of the specified size (256-byte records). D, 3-2 (or 33).

otd (expression)

Returns the decimal equivalent of the octal value expressed. I/O, 3-12.

P

**par** (expression)

Sets the parity type (listed below) used for I/O checking. I/O, 4-9.

0	Parity disabled.	2	Even parity.
1	Parity = 1.	3	Odd parity.

pclr

Sets default plotter values except scale units, select code, P1, P2, pen location and pen#. 9872 Plotter ROM only. I/O, 7-10.

pct select code

Passes active control to the specified HP-IB device. I/O, 2-26.

pen

Raises the plotter pen. I/O, 7-22.

pen# [expression]

Selects the plotter pen (1 through 4). 9872 Plotter ROM only. I/O, 7-22.

% string [:]

The % free-text prefix allows storing text without syntax checking. Free text is terminated with a semicolon or end of line. O&P, 7-25.

plt X coordinate , Y coordinate [, expression]

Move plotter pen to specified X,Y point. Optional expression controls pen (see below). I/O, 7-22.

even	lowers pen.	positive	action before plotting.
odd	raise pen.	negative	action after plotting.

pol select code

Conducts a parallel poll on the HP-IB. I/O, 2-25.

polc select code , byte

Sets parallel poll bits on the specified HP-IB device. I/O, 2-26.

polu select code

Clears parallel poll bits on the specified device. I/O, 2-26.

pos (string₁ , string₂)

Returns the character position of the second string within the first. O&P, 6-16.

prnd (expression , expression)

Returns the first expression rounded to the power of ten indicated by the second expression. O&P, 3-22.

prt expression or string [, expression or string [, ...]]

Prints the list of items on the internal printer. To print quotes use double quotes (e.g., 3: prt "print""text""in quotes."). O&P, 3-12.

psc select code

Sets the select code for all plotter ROM operations. psc 0 causes either the program to ignore all plotter operations (9872 ROM) or the computer to suppress output to plotter (9862 ROM). I/O, 7-5.

ptyp

Sets a plotter lettering mode. Press STOP key to terminate mode. I/O, 7-45.

R

rad

Sets radians units for angular calculations. O&P, 3-25.

$\sqrt{\quad}$ (expression)

Returns the square root of the expression. O&P, 3-22.

rcf [file number [, line number [, line number]] [, "SE" or "DB"]]

Records either all program lines onto the specified tape file (no line numbers) or only the specified block of lines. Including SE prevents the program from being listed or displayed when reloaded. Including DB records all trace and stop flags with the program for debugging. O&P, 5-16.

rcf file number , variable list

Records the listed variables onto the tape file. O&P, 5-16.

rck file number

Records the special function key definitions on the tape file. O&P, 5-22.

rcm file number

Records the entire computer memory on the specified tape file. O&P, 5-22.

rdb (select code)

Returns one 16-bit binary character code from the specified device. I/O, 3-4.

rdi (register number)

Returns a status byte from the interface specified by wti 0. I/O, 4-12.

rdm array variable [, array variable [, ...]]

Redimensions the array(s) to the specified dimensions. M, 16.

rds (select code)

Returns the current status word from the specified interface. I/O, 3-5.

red select code [. format no.] , variable list

Reads and stores data from the specified device. I/O, 1-5.

rem select code

Sends the remote message to either all HP-IB devices or only one device when its address is included in the select code. I/O, 2-18.

renm old file name , new file name

Renames a disk file on the default disk. D, 1-17 (or 28).

repk

Repacks files on the default disk. D, 4-5 (or 58).

res

Returns the result of the last keyboard operation not stored in a variable. O&P, 2-20.

resave file name [, 1st line number [, last line number]]

Stores a program (or only the specified lines) in an existing disk file. D, 2-9 (or 28).

ret

Ends a subroutine and returns program execution to the main program (line after gsb).
O&P, 3-34.

rew

Rewinds the tape. O&P, 5-6.

rkbd select code [, expression]

Enables a remote keyboard to control the computer. The expression indicates the keycode interpretation: 0 = ASCII (default) or 1 = 9825 keycodes. O&P, 7-24.

rnd (expression)

Returns a pseudo-random number from 0 to (less than) 1. A negative expression is used as a new seed. O&P, 3-22.

rot (expression₁ , expression₂)

Returns the result of binary rotation of the 16-bit equivalent of expression 1, rotated the number of bits indicated by expression 2. I/O, 3-13.

rprt file number , record number [, data list] [, "end" or "ens"]

Prints the list of data items on the disk file, starting at the specified record. Including "end" prints an EOF mark after the data. Including "ens" suppresses the automatic EOR mark printed after data. D, 3-12 (or 43).

rqs select code , byte

Requests service from the HP-IB system controller and sends the serial status byte upon response to a serial poll. I/O, 2-21.

rread file number , record number [, variable list]

Reads data from the disk file, starting at the specified record. Omitting the variable list just repositions the file pointer. D, 3-15 (or 46).

rss (select code)

Returns the 98036 Interface status register byte. O&P, 7-16.

run [line number or "label"]

Begins program execution, either at line 0 or at the specified line. O&P, 2-9.

S

save file name [, 1st line number [, last line number]]

Creates a program file and stores either the entire program (no line numbers) or only the specified lines. D, 2-2 (or 18).

savek file name

Creates a key file and stores all special function key definitions. D, 2-9 (or 29).

savem file name

Creates a memory file and stores the computer's read/write memory. D, 2-10 (or 57).

scl Xp1 , Xp2 , Yp1 , Yp2

Locates the origin and specifies user units for plotting operations. I/O, 7-7.

sfg [flag no. [, flag no. [, ...]]]

Sets either all 15 program flags to 1 or only the specified flags. O&P, 3-28.

sgn (expression)

Returns sign of expression: 0 = zero; 1 = positive; -1 = negative. O&P, 3-22.

shf (expression₁ , expression₂)

Returns the result of right-shifting the 16-bit binary equivalent of expression 1, the number of places indicated by expression 2. A negative expression 2 shifts the byte to the left. I/O, 3-14.

sin (expression)

Returns the sine of the expression. O&P, 3-2.

smpy number or simple variable [*] array variable₁ → array variable₂

Multiplies each element of array 1 by the scalar number. The * can be omitted. M, 13.

spc [expression]

Outputs the expressed number of line feeds on the internal printer. O&P, 3-16.

sprt file number , data list [, "end" or "ens"]

Prints the list of data items on the disk file. Including "end" prints an EOF mark after the data. Including "ens" suppresses the automatic EOR mark printed after data. D, 3-7 (or 38).

sread file number , variable list

Reads data from the specified file into the listed variables. D, 3-10 (or 41).

stf (string variable)

Unpacks and returns a split-precision number from its four-character string. O&P, 4-20.

store string name or "string" [, line number]

Stores program lines from an executing program. O&P, 7-21.

stp [line number₁ [, line number₂]]

Stops program execution either immediately or, optionally, at the specified line (line 1). Specifying both line numbers indicates a block of lines to stop at. O&P, 3-17.

str (expression)

Returns the ASCII character equivalent to the expression. O&P, 6-19.

T

tan (expression)

Returns the tangent of the expression. O&P, 3-25.

tfr source name , destination name [, expression [, last character]]

Transfers data between an I/O buffer and a peripheral device. Optional expression indicates the total number of bytes to transfer. Optional last character expression is the decimal value of the character to terminate the transfer. I/O, 6-8.

time (expression)

Causes an I/O operation to wait for a device to become ready for the specified number of milliseconds. I/O, 4-4.

tinit

Reinitializes a bad 9885 track during disk error recovery. 98217A ROM only. D, 4-15 (or 65).

tlist

Catalogs tape files on the internal printer (file types below). O&P, 5-9.

0	Null file.	4	Memory file.
1	Binary program.	5	Special function key file.
2	Numeric data file.	6	Program file.
3	String or string/data.		

tn ↑ (expression)

Returns 10 raised to the specified power. O&P, 3-24.

trc [1st line number [, last line number]]

Sets the master flag and, optionally, trace flags for specified program lines. O&P, 3-44.

trg select code

Sends the trigger message to the specified HP-IB device. I/O, 2-17.

trk expression

Specifies the tape track (0 or 1) for successive operations. O&P, 5-6.

trn array name → array name

Transposes rows and columns between arrays. M, 23.

type ([-] expression)

Returns a value indicating the next data-item type in a disk file. A positive expression causes any encountered EORs to be skipped (like with sread). A negative expression causes any EORs to be identified (like with rread). D, 3-20 (or 51). Return values are:

0	Unidentified type.	Indicates string overlapping record boundaries:	
1	Full-precision number.	2.1	Start of string.
2	String (within record).	2.2	Middle of string.
3	EOF mark.	2.3	End of string.
4	EOR mark.		

U

units

Returns the currently-set angular units. O&P, 3-25.

V

val (string)

Returns the numeric value of the string. O&P, 6-17.

vfy [return variable]

Verifies the contents of a tape file with the original in memory. Return variable: 0 = no error; 1 = error. O&P, 5-25.

vyb

Checks 98217A bootstraps on disk with those on the disk system cartridge. 98217A ROM only. D, 4-14 (or 67).

voff

Disables data-verification with disk print and copy. D, 4-6 (or 58).

von

Enables the disk data verification (default). D, 4-6 (or 59).

W

wait expression

The program waits for the specified time in milliseconds (from 1 to 32767). O&P, 3-16.

wrt select code [. format no.] [, item list]

Outputs the items to the specified device. I/O, 1-3.

wsc select code , expression

Outputs a control word (expression) to the specified interface. O&P, 7-14.

wsm select code , expression [, expression]

Outputs a mode word and, optionally a control word (second expression) to the specified 98036 Interface. O&P, 7-15.

wtb select code , expression [, expression [, ...]]

Outputs the byte representing each number or character to the specified device. I/O, 3-3.

wtc select code , expression

Outputs a control byte to the specified interface. I/O, 3-9.

wti 0, select code

Specifies an interface for successive wti or rdi operations. I/O, 4-11.

wti expression₁ , expression₂

Outputs a control byte (expression 2) to a specified interface register (expression 1). I/O, 4-11.

X

xax Yoffset [, tic interval [, start [, end [, no. of tics/label]]]]

Draws an X axis with optional tic marks and labels. 9872 Plotter ROM only. I/O, 7-11.

xref

Prints a cross reference of program variables and line numbers, using the current program in memory. O&P, 4-32.

Y

yax Xoffset [, tic interval [, start [, end [, no. of tics/label]]]]

Draws a Y axis with optional tic marks and labels. 9872 Plotter ROM only. I/O, 7-11.

Appendix C

Subject Index



This index references subjects in these 9825 manuals:

Title	Part No.	Abbreviation
9825 Operating & Programming Reference	09825-90200	O&P
9825 I/O Control Reference	09825-90210	I/O
9825 Disk Programming Manual	09825-90220	D
Matrix Programming Manual	09825-90022	M

The index does not list subjects in the Interfacing Concepts guide or manuals supplied with computer peripherals or interfaces. Page references for the old 9825/9885 Disk Programming Manual, 09885-90000, are listed in parentheses.

cat (disk catalog) D 1-16 (20)
 cfg (clear flag) O&P 3-29
 c format spec I/O 1-9
 chain (chain disk program files) . D 2-7 (25)
 char (string character) O&P 6-20
 character sets:
 display I/O 1-17
 plotter O&P 7-37
 printer (internal) I/O 1-14
 CLEAR key O&P 2-9
 clear flag (cfg) O&P 3-29
 clear:
 HP-IB interface (abort) I/O 2-26
 message (clr) I/O 2-17
 plotter I/O 7-10
 simple variables O&P 3-39
 cll (call) O&P 4-10
 cln (current line number) O&P 7-28
 clr (clear message) I/O 2-17
 cmd (HP-IB commands) I/O 2-31
 cmf (complement flag) O&P 3-29
 cmp (complement binary) I/O 3-13
 commands O&P 2-24
 common log (log) O&P 3-25
 compatibility (9820/21 & 9825) . . O&P A-10
 computer I/O scheme I/O iv
 concatenation (&) O&P 6-26
 cont (continue) O&P 2-24
 CONTINUE key O&P 2-20
 control bits (98032A) I/O 3-9
 conv (conversion) I/O 1-23
 copy (copy disk file) D 4-7 (60)
 cos (cosine) O&P 3-25
 cplt (character plot) I/O 7-41
 conversion table (ctbl) I/O 4-6
 cross reference (xref) O&P 4-32
 csiz (character size) I/O 7-38
 csv (clear simple variables) O&P 3-39
 ctbl (conversion table) I/O 4-6
 cursor controls (display) O&P 2-16

d

data input operations I/O 1-5,3-4
 data I/O format I/O v
 data output operations I/O 1-3,3-3
 data:
 input operations I/O 2-5,3-4
 I/O format I/O v
 messages (HP-IB) I/O 2-4
 output operations I/O 1-3,3-3

data transfer:
 input I/O 6-9
 output I/O 6-8
 debug ("DB") O&P 5-16
 debug programs O&P 3-41
 default:
 computer conditions O&P A-3
 I/O formats I/O vii
 decimal mode (mdec) I/O 3-11
 default:
 deg (degrees units) O&P 3-25
 del (delete line) O&P 2-25
 DELETE keys O&P 2-17,2-18
 delimiters:
 input (read) I/O 1-6
 buffer transfer (tfr) I/O 6-8
 terminal I/O (eol) O&P 7-12
 write (wrt) I/O 1-3
 determinant, array M 24
 dev (device name) I/O 2-9
 device address (HP-IB) I/O 2-6
 dig (digitize) I/O 7-48
 digit rounding (drnd) O&P 3-22
 dim (dimension variables) O&P 3-37
 dimensioning strings O&P 6-4
 dirc (copy disk directory) D 4-16 (65)
 direct memory access I/O 6-6
 disk drive D 1-1 (2)
 disk operations D 4-1 (15)
 display:
 character set I/O 1-17
 control keys O&P 2-16
 dsp O&P 3-12
 divide (/) O&P 3-19
 division, array M 12
 DMA buffer I/O 6-6
 dot matrix in syntax O&P 3-6
 drive (set disk drive) D 1-13 (17)
 drnd (digit round) O&P 3-22
 dsp (display) O&P 3-12
 dto (decimal to octal) I/O 3-12
 dtrk (dump disk track) D 4-16 (65)
 dtype (disk type) D 1-15
 dump (dump disk to tape) D 4-11 (62)

e

e format spec I/O 1-9
 editing O&P 2-17
 edit specifications I/O 1-9
 eir (enable interrupt) I/O 5-6
 end O&P 3-17

enp (enter print) O&P 3-15
 ent (enter) O&P 3-13
 ENTER EXP key O&P 2-21
 enter exponent (e) O&P 3-10
 EOF mark (disk) D 1-10 (51)
 eol (end-of-line sequence) O&P 7-12
 EOR mark (disk) D 1-10 (51)
 eor (exclusive OR) I/O 3-13
 equ (equate) I/O 2-33
 equal to (=) O&P 3-20
 equipment supplied (see packing lists)
 ERASE key O&P 2-15
 erase (erase memory) O&P 2-26
 ert (error-line variable) I/O 4-4
 ern (error-number variable) I/O 4-4
 error recovery (on err) I/O 4-4
 errors:
 math O&P 3-26
 codes O&P D-1
 tape drive O&P 5-26
 ert (erase tape track) O&P 5-15
 even parity I/O 4-9
 exclusive OR (xor) O&P 3-21
 EXECUTE key O&P 2-9
 execution times O&P 3-46
 exp (exponential) O&P 3-24
 exponential format O&P 3-8
 exponential functions O&P 3-22
 exponentiate (\uparrow) O&P 3-19
 expressions, numeric O&P 3-18
 extended device address I/O 2-8
 Extended I/O ROM O&P 1-9, I/O iii
 extended read status (rds) I/O 2-34

f

fast read/write buffer I/O 6-5
 fdf (find tape file) O&P 5-8
 FETCH key O&P 2-17
 fetch (fetch line) O&P 2-27
 files:
 files statement (disk) D 3-3 (34)
 disk D 1-9 (11)
 string D 3-20 (51)
 tape file size O&P 5-12
 find file (fdf) O&P 5-8
 fixed-point format (fxd) O&P 3-9
 flags:
 debugging O&P 3-43
 flags 13 through 15 O&P 3-28
 programmable flags O&P 3-28
 status flags I/O 4-12

flg (flag) O&P 3-30
 flowcharting, program O&P 3-3
 flt (floating-point format) O&P 3-10
 fmt (I/O format) I/O 1-8
 for (for...next) O&P 4-3
 formats:
 free-field I/O 1-5, 1-6
 I/O I/O 1-3
 numeric (fxd, flt) O&P 3-9
 fmt specifications I/O 1-9
 frc (fraction) O&P 3-24
 free text (%) O&P 7-25
 fti (full to integer) O&P 4-26
 fts (full to short) O&P 4-20
 functions, mathematical O&P 3-22
 fuses, power O&P 1-6
 f format spec I/O 1-9
 FWD key O&P 2-18
 fxd (fixed-point format) O&P 3-9
 fz format spec I/O 1-9

g

General I/O ROM O&P 1-9, I/O iii
 get (get disk program file) D 2-4 (23)
 getb (get binary disk file) D 2-11 (64)
 getk (get disk keys file) D 2-9 (29)
 getm (get disk memory file) D 2-10 (57)
 grad (grads units) O&P 3-25
 graphics language (HP-GL) ... I/O 7-3, 7-50
 greater than (>) O&P 3-20
 greater than or equal to (>= or =>)
 O&P 3-20
 grounding, equipment O&P 1-4
 gsb (go subroutine) O&P 3-34
 gto (go to) O&P 3-31

h

hierarchy, math O&P 3-18
 hints, programming O&P 3-43
 HP-GL (graphics language) ... I/O 7-3, 7-50
 HP-IB:
 functions I/O 2-40
 interrupts I/O 5-8
 lines I/O 2-37
 messages I/O 2-4
 operations I/O 2-1
 HPL programming O&P 3-3
 HPL syntax O&P B-1



i

I/O format I/O iv
 idf (identify tape file) O&P 5-7
 idn (identity matrix) M 22
 if (if...then) O&P 3-36
 immediate execute keys O&P 2-22
 immediate continue keys O&P 2-22
 implied multiplication O&P 3-20
 ina (initialize array) M 8
 inclusive OR (ior) I/O 3-13
 incremental plotting I/O 7-29
 indirect storage O&P 3-8
 init (initialize disk) D 1-6, 4-3 (90)
 inspection, equipment O&P 1-3
 int (integer) O&P 3-22
 integer-precision storage O&P 4-26
 interface:
 overview I/O vi
 registers I/O 4-10
 internal peripherals I/O iv
 internal printer:
 character set I/O 1-14
 loading paper O&P 1-8
 prt O&P 3-12
 interrupt:
 abortive I/O 5-11
 buffer I/O 6-5
 end-of-line (EOL) I/O 5-4
 HP-IB I/O 5-8
 keyboard O&P 7-6
 lockouts I/O 5-14
 peripheral I/O 5-3
 programmable I/O 5-3
 vectored (EOL) I/O 5-4
 interrupt enable (eir) I/O 5-6
 interrupt return (iret) I/O 5-7
 inv (inverse matrix) M 24
 installation, computer O&P 1-3
 I/O bus and format I/O iv
 iof (interface flag) I/O 4-12
 ior (inclusive OR function) I/O 3-13
 ios (interface status) I/O 4-12
 iplt (incremental plot) I/O 7-29
 iret (interrupt return) I/O 5-7
 itf (integer to full) O&P 4-26

j

jmp (jump) O&P 3-33

k

key (key buffer empty) O&P 7-8
 keyboard operations O&P 2-1
 keyboard (ASCII) O&P A-7
 Keyboard magazine O&P 1-12
 keycodes, decimal O&P A-6
 key repetition O&P 2-5
 kill (purge disk file) D 1-18 (27)
 killall (purge all disk files) D 1-18 (67)
 kret (keyboard interrupt return) ... O&P 7-9

l

labeled branching O&P 3-30
 labeling axes I/O 7-36
 labels, line O&P 2-8
 lazy T (+) O&P 2-5
 lbl (plot labels) I/O 7-36
 lcl (local message) I/O 2-19
 ldb (load binary program from tape)
 O&P 5-23
 ldf (load file from tape) O&P 5-18
 ldk (load key file from tape) O&P 5-22
 ldp (load program file from tape) . O&P 5-18
 len (string length) O&P 6-14
 leading spaces, suppressing I/O 1-9
 leading zeros I/O 1-9
 less than (<) O&P 3-20
 less than or equal to (<= or =<) O&P 3-20
 lim (plotter pen limit) I/O 7-34
 line (plotter line type) I/O 7-32
 line length (display) O&P 2-5
 line renumbering O&P 3-30
 linking programs O&P 5-20
 list (list program on display) O&P 3-39
 list# (list to device) I/O 1-23
 list fn (list function) O&P 3-39
 listener (HP-IB) I/O 2-8
 listk (list keys) O&P 3-39
 live keyboard O&P 2-28
 lkd (live keyboard disable) O&P 2-32
 lke (live keyboard enable) O&P 2-32
 llo (local lockout message) I/O 2-19
 ln (natural log) O&P 3-24
 load data:
 from disk (rread, sread) D 3-10 (41, 46)
 from tape (ldf) O&P 5-21
 load keys:
 from disk (getk) D 2-9 (29)
 from tape (ldk) O&P 5-22

load memory;
 from disk (getm) D 2-10 (57)
 from tape (ldm) O&P 5-23
 load program:
 from disk (get) D 2-4 (22)
 from tape (ldp) O&P 5-18
 local parameters O&P 4-12
 logarithms:
 common log (log) O&P 3-24
 natural log (ln) O&P 3-24
 logical operators O&P 3-21
 ltr (letter plot) I/O 7-47
 ltrk (update disk track) D 4-16 (57)

m

mark tape (mrk) O&P 5-10
 mat (matrix multiplication) M 19
 math functions O&P 3-22
 math hierarchy O&P 3-18
 Matrix ROM O&P 1-10, M 1
 max (maximum) O&P 3-22
 mdec (decimal mode) I/O 3-11
 memory:
 organization O&P 2-7
 usage O&P 3-40
 messages, HP-IB control I/O 2-4
 min (minimum) O&P 3-22
 minus sign (-) O&P 3-19
 mod (modulus) O&P 3-19
 mounting, computer O&P 1-12
 moving the origin (plotting) I/O 7-27
 mrk (mark tape) O&P 5-10
 multiple listeners I/O 2-6
 multiply (*) O&P 3-19
 multiplication, implied O&P 3-20

n

N-way branching O&P 3-37
 natural log (ln) O&P 3-22
 nal (last program line) O&P 7-24
 next (for...next) O&P 4-3
 nesting:
 for...next loops O&P 4-6
 subprograms O&P 4-16
 non-active controller I/O 2-26
 nor (normal) O&P 3-44
 not (operator) O&P 3-22
 not equal to (#, > < or > <) O&P 3-20

null field:
 in ent O&P 3-14
 in strings O&P 6-7
 null tape file O&P 5-7
 num (string numeric value) O&P 6-21
 numeric formats O&P 3-8

O

octal mode I/O 3-11
 oct (octal to decimal) I/O 3-12
 odd parity I/O 4-9
 ofs (offset plot) I/O 7-27
 on end D 3-19 (50)
 on err (on error) I/O 4-4
 on key O&P 7-6
 oni (on interrupt) I/O 5-5
 open (open disk file) D 3-2 (33)
 operating system module O&P 1-7
 operators:
 arithmetic O&P 3-19
 assignment O&P 3-19
 logical O&P 3-21
 relational O&P 3-20
 string concatenation O&P 6-26
 OR functions (ior, eor) I/O 3-12
 OR operators (or, xor) O&P 3-21
 otd (octal to decimal) I/O 3-12
 out-of-limits conditions (plotting)
 I/O 7-23
 overflow, buffer I/O 6-6
 overhead, recording strings O&P 6-33

p

% (free text) O&P 7-25
 par (parity check) I/O 4-9
 parallel polling I/O 2-25
 parity checking I/O 4-9
 passing parameters O&P 4-12
 pclr (reset plotter) I/O 7-10
 pct (pass HP-IB control) I/O 2-26
 pen (control pen) I/O 7-22
 pen# (select pen) I/O 7-22
 peripheral:
 control I/O vi
 interrupt I/O 5-3
 status (rds) I/O 3-5
 plotter operations I/O 7-2
 plotter ROMs O&P 1-9, I/O 7-3

plt (plot) I/O 7-22
 plus sign (+) O&P 3-19
 pi O&P 2-21
 p-numbers O&P 4-16
 pol (parallel poll) I/O 2-25
 polc (poll configure) I/O 2-26
 polling:
 parallel I/O 2-25
 serial I/O 2-22
 polu (poll unconfigure) I/O 2-26
 pos (string position) O&P 6-16
 power cords O&P 1-4
 power requirements O&P 1-5
 prerecorded programs O&P 1-11
 print all O&P 2-14
 print arrays (aprt) M 8
 print strings O&P 6-29
 printer paper O&P 1-8
 printer operations I/O 1-14
 printer (internal) status I/O 3-5
 prnd (power-of-ten round) O&P 3-22
 programming O&P 3-3
 prompts:
 in ent O&P 3-13
 in enp O&P 3-15
 prt (print) O&P 3-12
 psc (plotter select code) I/O 7-5
 ptyp (plotter typewriter mode) I/O 7-45

q

quote marks (" "):
 in dsp O&P 3-12
 in prt O&P 3-13
 in strings O&P 6-5

r

r-variables O&P 3-7
 rad (set radians units) O&P 3-25
 radical sign ($\sqrt{\quad}$) O&P 3-22
 random numbers (rnd) O&P 3-23
 range, computing O&P 2-6
 rcf (record file on tape) O&P 5-16
 rck (record keys on tape) O&P 5-22
 rcm record memory on tape) O&P 5-22
 rdb (read binary data) I/O 3-4
 rdi (read interface) I/O 4-12
 rdm (redimensioning arrays) M 16
 rds (read status) I/O 3-5
 read binary (rdb) I/O 3-4
 read interface (rdi) I/O 4-12
 read only memory (ROM) O&P 1-8,2-7
 read only variables (with on err) I/O 4-4
 read/write memory (RWM) O&P 2-6
 RECALL key O&P 2-18
 RECORD key O&P 2-15
 record data:
 on disk (rprr, sprt) D 3-7 (38, 45)
 on tape (rdf) O&P 5-16
 record keys:
 on disk (savek) D 2-9 (29)
 on type (rck) O&P 5-22
 record memory:
 on disk (savem) D 2-10 (57)
 on tape (rcm) O&P 5-22
 record programs:
 on disk (save) D 2-2 (18)
 on tape (rcf) O&P 5-16
 RECORD tab on tape O&P 5-4
 red (read data) I/O 1-5
 redimensioning arrays (rdm) M 16
 relational operators O&P 3-20
 relative branching O&P 3-30
 rem (remote message) I/O 2-18
 remarks (labels) O&P 3-31
 renm (rename disk file) D 1-17 (28)
 repk (repack disk) D 4-5 (58)
 require service message (rqs) I/O 2-21
 res (result) O&P 2-20
 RESET key O&P 2-14
 resave (re-save disk file) D 2-9 (28)
 RESULT key O&P 2-20
 ret (return) O&P 3-34
 rew (rewind tape) O&P 5-6
 REWIND key O&P 2-14
 rkbd (read keyboard) O&P 7-17
 rnd (random number) O&P 3-22
 rom (ROM error variable) I/O 4-4
 ROMs, overview O&P 1-8
 ROM memory usage:
 Advanced Programming O&P 4-3
 Disk D 1-1 (3)
 Extended I/O I/O iii
 General I/O I/O iii
 Matrix M 3
 String Variables O&P 6-3
 Systems Programming O&P 7-3
 rot (rotate) I/O 3-13
 rounding O&P 3-22
 rprr (random disk pring) D 3-12 (43)
 rqs (request service) I/O 2-21
 rread (random disk read) D 3-15 (46)

rss (read serial status) O&P 7-16
 run O&P 2-24
 RUN key O&P 2-9

S

save (save program on disk) D 2-2 (18)
 savek (save keys on disk) D 2-9 (59)
 savem (save memory on disk) D 2-10 (57)
 scientific notation (flt) O&P 3-10
 scalar multiplication (smpy) M 13
 scl (scale plot) I/O 7-7
 secure programs O&P 5-16
 select code:
 recommended settings I/O A-8
 syntax I/O vii
 selecting pens I/O 7-25
 serial polling I/O 2-22
 service contracts O&P 1-11
 service requests I/O 2-20
 sfg (set flag) O&P 3-28
 sgn (sign) O&P 3-22
 shf (shift) I/O 3-14
 SHIFT and SHIFT LOCK keys O&P 2-19
 significant digits O&P 3-11
 sin (sine) O&P 3-25
 single character output I/O 1-17
 smty (scalar multiply) M 13
 spacing O&P 2-5
 spc (space) O&P 3-16
 special function keys:
 defining and using O&P 2-21
 in live keyboard O&P 2-29
 split-precision storage O&P 4-20
 sprt (serial disk print) D 3-7 (38)
 square root O&P 3-22
 sread (serial disk read) D 3-10 (41)
 statements, HPL O&P B-1
 status conditions, computer O&P A-3
 status bits:
 HP-IB interface I/O 2-34
 KDP (internal) I/O 3-5
 98032 interface I/O 3-8
 read status (rds) I/O 3-5
 tape drive (internal) I/O 3-7
 status byte message:
 receiving (serial polling) I/O 2-22
 sending I/O 2-22
 status bytes I/O 2-34
 STEP key O&P 2-15
 STOP key O&P 2-19

stp (split to full) O&P 4-20
 storage range O&P 2-6
 STORE key O&P 2-9
 store (store lins) O&P 7-21
 store programs O&P 5-16
 stp (stop) O&P 3-17
 str (string) O&P 6-19
 string operator (&) O&P 6-26
 String Variables ROM O&P 1-9,6-3
 string variables operations O&P 6-1
 subprograms O&P 4-10
 subroutines:
 go sub O&P 3-34
 from live keyboard O&P 2-29
 subscripts:
 array O&P 3-6
 string O&P 6-6
 substrings O&P 6-6
 subtract (-) O&P 3-19
 suppressing leading spaces I/O 1-9
 syntax:
 brackets [] O&P 3-6
 conventions O&P 3-6
 HPL listing O&P B-1
 Systems Programming ROM O&P 7-19

t

tan (tangent) O&P 3-25
 tape drive, internal O&P 5-1, I/O 3-7
 testing the computer O&P 1-7
 tfr (transfer) I/O 6-8
 tic marks, plotting I/O 7-7
 time (time out) I/O 4-4
 tinit D 4-15 (65)
 tlist (tape list) O&P 5-9
 tn↑ (ten to a power) O&P 3-24
 transfer parameters I/O 2-8
 transfer (tfr) I/O 6-8
 transposition (trn) M 23
 trc (trace) O&P 3-44
 trg (trigger message) I/O 2-16
 trig functions O&P 3-25
 trk (tape track) O&P 5-6
 trn (transpose) M 23
 truth tables:
 binary functions I/O 3-12
 logical operators O&P 3-21
 type (disk data type) D 3-20 (51)
 types of buffers I/O 6-4
 typewriter mode (plotting) I/O 7-45

U

unary - O&P 3-19
 underflow O&P 3-28
 underflow, buffer I/O 6-6
 unlisten command I/O 2-8
 units, scale statement I/O 7-7
 units (trig units) O&P 3-25

V

val (string value) O&P 6-17
 variables:
 allocation O&P 3-8
 array O&P 3-6
 erasing O&P 2-15,2-26,3-39
 loading from tape O&P 5-21
 read only (with on err) I/O 4-4
 recording on tape O&P 5-16
 string O&P 6-3
 vectored interrupt I/O 5-4
 vectors M 3
 vfy (verify data) O&P 5-25
 vfyb (verify disk binary) D 4-4 (67)
 voff (disk auto-verify off) D 4-6 (58)
 voltage setting, computer O&P 1-5
 von (disk auto-verify on) D 4-6 (59)

W

wait O&P 3-16
 word (16 bits) I/O 6-7
 wrt (write) I/O 1-3
 wsc (write serial control) O&P 7-14
 wsm (write serial mode) O&P 7-15
 wtb (write binary) I/O 3-3
 wtc (write control) I/O 3-9
 wti (write interface) I/O 4-11

X

xax (X axis) I/O 7-11
 x format spec I/O 1-11
 xor (exclusive OR) O&P 3-21
 xref (cross reference) O&P 4-32

Y

yax (Y axis) I/O 7-11

Z

z format spec I/O 1-11

Notes

Appendix D Table of Contents



Mainframe Errors (00 thru 77)	D-3
Advanced Programming ROM Errors (A0 thru A9)	D-7
9885 Binary Disk Errors (B0 thru B8)	D-8
Systems Programming ROM Errors (C0 thru C9)	D-8
Disk ROM Errors (D0 thru D9 and d0 thru d9)	D-9
Extended I/O ROM Errors (E0 thru E9)	D-10
9885 Disk Hardware Errors (F0 thru F9)	D-10
General I/O ROM Errors (G0 thru G9)	D-11
Matrix ROM errors (M1 thru M5)	D-11
9862A Plotter ROM Errors (P1 thru P8)	D-12
9872A Plotter (HP-GL) ROM Errors (P1 thru P8 and p0 thru p6)	D-12
String Variable ROM Errors (S0 thru S9)	D-14

Notes

Appendix D

Error Codes

An error in a program sets the program line counter to line 0. Press the continue key to continue the program from line 0. Execute the continue command with a line number to continue at any desired line (such as: cont 50).

00	System error.
01	Unexpected peripheral interrupt.
02*	Unterminated text.
03*	Mnemonic is unknown. Mnemonic not found because disk may be down.
04	System is secured.
05	Operation not allowed; line cannot be stored or executed with line number.
06*	Syntax error in number.
07*	Syntax error in input line.
08	Internal representation of the line is too long (gives cursor sometimes).
09	gto, gsb, or end statement not allowed in present context. Attempt to execute a next statement either from keyboard while for/next loop using same variable is executed in program or from program while for/next loop using same variable is executed from keyboard. Attempt to call function or subroutine from keyboard.
10*	gto or gsb statement requires an integer.
11	Integer out of range or integer required; must be from -32768 thru +32767.
12*	Line cannot be stored; can only be executed.

* Press the **RECALL** key to position the cursor at the location of the error.

- 13 ent statement not allowed in present context.
- 14 Program structure destroyed.
- 15 Printer out of paper or printer failure.
- 16 String Variables ROM not present for the string comparison. Argument in relational comparison not allowed.
- 17 Parameter out of range.
- 18 Incorrect parameter.
- 19 Bad line number.
- 20 Missing ROM or binary program. The second number indicates the missing ROM. In the program mode, the line number is given instead of the ROM number. Displayed number and missing item:
- | | | | |
|---|--------------------------|----|-----------------|
| 1 | Binary Program | 10 | Matrix ROM |
| 4 | Systems Programming ROM | 11 | Plotter ROM |
| 6 | Strings ROM | 12 | General I/O ROM |
| 8 | Extended I/O ROM | 17 | Disk ROM |
| 9 | Advanced Programming ROM | | |
- 21 Line is too long to store.
- 22 Improper dimension specification.
- 23 Simple variable already allocated.
- 24 Array already dimensioned.
- 25 Dimensions of array disagree with number of subscripts.
- 26 Subscript of array element out of bounds.
P-number reference is negative.
- 27 Undefined array.
- 28 ret statement has no matching gsb statement.
- 29 Cannot execute line because a ROM or binary program is missing.
- 30 Special function key not defined.
- 31 Non-existent program line.
- 32 Improper data type.
Non-numeric value in for statement or in fts or fti function.

- 33 Data types do not match in an assignment statement.
- 34 Display overflow due to pressing a special function key.
- 35 Improper flag reference (no such flag).
- 36 Attempt to delete destination of a gto or gsb statement.
- 37 Display buffer overflow caused by dsp statement.
- 38 Insufficient memory for subroutine return pointer. Memory overflow during function or subroutine call.
- 39 Insufficient memory for variable allocation or binary program.
Dimensioned string cannot exceed 32,766 elements.
- 40 Insufficient memory for operation.
Memory overflow while using for statement or while allocating local p-numbers.
- 41 No cartridge in tape transport.
- 42 Tape cartridge is write protected. (Slide record tab to right for recording.)
- 43 Unexpected Beginning-Of-Tape (BOT) or End-Of-Tape (EOT) marker encountered. Tape transport failure.
- 44 Verify has failed.
- 45 Attempted execution of idf statement without parameters or mrk statement when tape position is unknown.
- 46 Read error in file body.
- 47 Read error in file head.
- 48 End-Of-Tape (EOT) encountered before all files were marked.
- 49 File too small.
- 50 Idf statement for a program file must be last statement in the line. get or chain statement should be the last statement in a line.
- 51 or 52 Memory configuration error for attempted Idm statement. For example, a ROM present when memory was recorded is now not present (see error 20), or attempting to load a memory file recorded on a 9825 into a 9825B.

D-6 Error Codes

Memory files are not compatible between the 9825A and 9825B. Only the program portion can be recovered by loading the memory file into the original machine and doing a rcf. This program file can then be loaded into any 9825 with the ldf statement.

- 53 Negative parameter in cartridge statement.
- 54 Binary program to be loaded is larger than present binary program and variables have been allocated.
- 55 Illegal or missing parameter in a cartridge statement.
- 56 Data list is contiguous in memory for a cartridge statement.
- 57 Improper file type.
- 58 Invalid parameter in rcf statement; "SE" or "DB" expected.
- 59 Attempt to record a program or special function keys which do not exist.
- 60 Attempt to load an empty file or the null file (type = 0).
- 61 The line referenced in an ldf or ldp statement does not exist. If the line containing the ldf or ldp statement has been overlaid by the load operation, the line number in the display may be incorrect.
- 62 Specified memory space is smaller than cartridge file size.
- 63 Cartridge load operation would overlay subroutine return address in program; load not executed.

Disk load operation would overlay gsb return address; load not executed.
- 64 Attempt to execute ldk, ldf (program file), or ldp during live keyboard statement.

get, chain or getk not allowed from live keyboard mode or during an ent statement.
- 65 File not found.
File specified in the previous fdf statement does not exist.

Default values associated with errors 66 thru 77 when flag 14 is set are explained in the programming chapter of the operating and programming manual.

- 66 Division by zero.
A mod B, with B equal to zero.

- 67 Square root of negative number.
- 68 Tan ($n * \pi/2$ radians).
Tan ($n * 90$ degrees).
Tan ($n * 100$ grads).
where n is an odd integer.
- 69 In or log of a negative number.
- 70 In or log of zero.
- 71 asn or acs of number less than -1 or greater than $+1$.
- 72 Negative base to non-integer power.
- 73 Zero to the zero power ($0 \uparrow 0$).
- 74 Storage range overflow.
- 75 Storage range underflow.
- 76 Calculation range overflow.
- 77 Calculation range underflow.
- A0 Relational operator in for statement not allowed. No closing apostrophe.
- A1 A for statement has no matching next statement.
- A2 A next statement encountered without a previous for statement.
- A3 Non-numeric parameter passed as a p-number.
- A4 No return parameter for a function call.
- A5 No functions or subroutines running.
Improper p-number.
- A6 Attempt to allocate local p-numbers from the keyboard.
- A7 Wrong number of parameters in fts, stf, fti, or itf function. stf or itf parameter must be a string (not a numeric). stf or itf parameter contains too few characters.
- A8 Overflow or underflow in fts function.
Overflow in fti function.
- A9 String Variables ROM missing for stf or itf functions.



D-8 Error Codes

Errors B0 thru B8 may result during the binary disk initialization and disk error recovery routines.

B0	Wrong syntax, argument out of range or variable not properly dimensioned.
B1	More than six defective tracks on the disk.
B2	Verify error. Boots on the disk not identical to boots on the cartridge.
B3	dtrk or tinit not allowed because error information lost or error not d5, d6, d7 or d9.
B4	Attempt to access record for error correction which isn't part of data file.
B5	Improper string length (inconsistent with length given in header).
B6	Not enough space in computer buffer for data item. Item can't be placed in this part of buffer.
B7	Missing Disk or String ROM.
B8	Track still bad after tinit.
C0	Missing General I/O or Extended I/O ROM.
C1	Incorrect number of parameters.
C2	Improper parameter specified.
C3	Wrong parameter type.
C4	Illegal buffer type for bred statement.
C5	Key buffer overflow.
C6	Too large or wrong sign of parameter.
C7	Improper execution of store statement.
C8	Illegal use of kret.
C9	Missing 98036A Interface card.


D0	Improper argument.
D1	Argument out of range.
D2	Improper file size; must be an integer from 1 thru 32767. No lines to store for save or savek.
D3	Invalid file name.
D4	File not found.
D5	Duplicate file name or attempt to copy non-data file to existing file.
D6	Wrong file type.
D7	Directory overflow.
D8	Insufficient storage space on disk.
D9	Verify error. Disk controller detected no read errors, but the data read back doesn't compare with the original. Reprint data. If the problem persists, service the drive, interface or the computer.

DISK IS DOWN (98217A ROM)**UNABLE TO ACCESS DISC CONTROLLER (98228A ROM)**

Computer cannot access the disk controller. If control is not restored (e.g., power on) press RESET or STOP to cancel operation.

d0	Firmware/driver out of synchronization. Too many defective tracks within it (press RESET).
d1	All drives in system not powered on.
d2	Door opened while disk being accessed or during dump, load or copy.
d3	Disk not in drive or no such drive number. Door open on 9895 drive.
d4	Write not allowed to protected disk.
d5	Record header error (use error recovery routine.)
d6	Track not found (use error recovery routine.)
d7	Data checkword error. (use error recovery routine.)
d8	Hardware failure (Press RESET).
d9	Verify error. Data is readable under normal margins but not under reduced margins. Reprint data. If problem persists, back up disk (new media) or service drive.

D-10 Error Codes

E0	General I/O ROM missing. HP-IB error under interrupt.
E1	Wrong number of parameters.
E2	Improper buffer device or equate table usage. Multiple-listeners error. Buffer busy.
E3	Wrong parameter type.
E4	Timeout error.
E5	Buffer underflow or overflow.
E6	Parameter value out of range.
E7	Parity failure.
E8	Improper use of irect statement. Attempt to DMA with HP-IB. Buffer or select code is busy.
E9	Illegal HP-IB operation.
F0	File overflow when read or print executed.
F1	Bootstraps not found (98217A ROM) or wrong memory configuration for 98228A Disk ROM (9825T required).
F2	String read but wrong data type encountered.
F3	Attempt to read data item but type doesn't match.
F4	Availability table overflow (repack).
F5	Attempt on end branch from other than running program.
F6	Unassigned data file pointer.
F7	Disk is down; line cannot be reconstructed.
F8	Disk is down and  pressed.
F9	System error (save files individually and reinitialize).

G1	Incorrect format numbers.
G2	Referenced format statement has an error.
G3	Incorrect I/O parameters.
G4	Incorrect select code.
G5	Incorrect read parameter.
G6	Improper conv statement parameters.
G7	Unacceptable input data.
G8	Peripheral device down.
G9	Interface hardware problem.
M1*	Syntax error.
M2	Improper dimensions. Array dimensions incompatible with each other or incompatible with the stated operation.
M3	Improper redimension specification. New number of dimensions must equal original number; new size cannot exceed original size.
M4*	Operation not allowed. An array which appears to the left of ' cannot also appear on the right.
M5	Matrix cannot be inverted. Computed determinant = 0.





* Press the **RECALL** key to position the cursor at the location of the error.

9862A Plotter ROM Error Codes

P1	Wrong state. Statements executed out of order.
P2	Wrong number of parameters.
P3	Wrong type of parameters. Parameters for an lbl statement must be expressions, text, or string variables.
P4	Scale out of range. Maximum value is less than or equal to the minimum value.
P5	Integer out of range. Pen control parameter is out of the range —32768 thru +32767 or the select code is not 0 or in the range 2 thru 15.
P6	Character size out of range. Width or height in letter statement is zero or there is an integer overflow in csiz calculations or results.
P7	Not used.
P8	Axes origin off-scale. X, Y specified for axis statement doesn't fall on plotter surface.
PLT DOWN	Check interface connection and select code setting; be sure LINE and CHART HOLD are on.

9872A Plotter ROM (HP-GL) Error Codes

P1	Attempt to store into constant. Occurs when one or more parameters in a dig statement are constants rather than variables.
P2	Wrong number of parameters. Occurs on instructions with numeric-only parameter lists (scl, ofs, plt, iptl, cplt, xax, yax, lim, dig, csiz, line, pen#, and psc). In certain unusual cases where a parameter list contains user-level function calls, an instruction having an incorrect number of parameters may be executed.

P3	Wrong type of parameter or illegal parameter value.
P4	No HP-IB device number specified. Occurs when psc parameter is from 0 thru 14 and an HP-IB card is at the corresponding select code.
P5	Pen control value not from -32768 thru 32767. Hardware transmission error occurs between plotter and computer.
P6	No HP-IB card at specified select code.
P7	axe or ltr statement encountered; 9872 ROM cannot execute them.
P8	Computer  key cancelled operation. Occurs when the plotter fails to respond for three seconds after the  key has been pressed.
p0	Transmission error. The calculator has received an illegal ASCII input from the plotter.
p1	Instruction not recognized. The plotter has received an illegal character sequence.
p2	Wrong number of parameters. Too many or too few parameters have been sent with an instruction.
p3	Bad parameter. The parameters sent to the plotter with an instruction are out of range for that instruction.
p4	Illegal character. The character specified as a parameter is not in the allowable set for that instruction.
p5	Unknown character set. A character set out of the range 0 thru 4 has been designated as either the standard or alternate character set.
p6	Position overflow. An attempt to draw a character or perform a cplot that is located outside of the plotters numeric limit of -32768 thru +32767.

Errors generated by write (wrt) and read (red) statements will typically be displayed in the next executed plotter ROM statement. This can be avoided by using an output error command (wrt select code, "OE"); followed by a read statement (red select code, variable) to check for errors after read or write statements that address the plotter.

D-14 Error Codes

S0	Invalid set of strings in data list of ldf statement.
S1	Improper argument for string function or string variable.
S2	More parameters than expected for string function or string variable.
S3	Accessing or assigning to non-contiguous string, num function of null string.
S4	Trying to find the value of non-numeric string or null string. Exponent too large. Exponent format invalid (e.g., 1e+ +).
S5	Invalid destination type for string assignment.
S6	Parameter is zero, or negative, exceeded dimensioned size. Invalid sequence of parameters for string variable.
S7	String not yet allocated.
S8	String previously allocated.
S9	Maximum string length exceeded; additional string length must be specified in dim statement.
SPARE DIR.	Printed when the spare disk directory (backup track) automatically replaces the main directory.

SALES OFFICES

Arranged alphabetically by country



ANGOLA

Telectra
Empresa Técnica de
Equipamentos
Eléctricos, S.A.R.L.
R. Barbosa Rodrigues,
411-DT *
Caxa Postal, 6487
Luanda
Tel: 35515/6

ARGENTINA

Hewlett-Packard Argentina S.A.
Santa Fe 2035, Martínez
6140 **Buenos Aires**
Tel: 792-1239, 798-6086
Telex: 122443 AR CIGY
Biotron S.A.C.I.Y.M.
Avda. Paseo Colon 221
9 piso
1399 **Buenos Aires**
Tel: 30-4846/185/18384
34-9356/0460/4551
Telex: (33) 17595 BIO AR

AUSTRALIA

AUSTRALIA CAPITAL TERR.
Hewlett-Packard Australia Pty.
Ltd.
121 Wollongong Street
Fyshwick, 2608
Tel: 804244
Telex: 626550

NEW SOUTH WALES

Hewlett-Packard Australia Pty.
Ltd.
31 Bridge Street
Pymble, 2073
Tel: 4496566
Telex: 21561

QUEENSLAND

Hewlett-Packard Australia Pty.
Ltd.
5th Floor
Teachers Union Building
495-499 Boundary Street
Spring Hill, 4000
Tel: 2291544

SOUTH AUSTRALIA

Hewlett-Packard Australia Pty.
Ltd.
153 Greenhill Road
Parkside, 5063
Tel: 2725911
Telex: 82536

VICTORIA

Hewlett-Packard Australia Pty.
Ltd.
31-41 Joseph Street
Blackburn, 3130
Tel: 89-6351
Telex: 31024 MELB

WESTERN AUSTRALIA

Hewlett-Packard Australia Pty.
Ltd.
141 Sirling Highway
Nedlands, 6009
Tel: 3665455
Telex: 93859

AUSTRIA

Hewlett-Packard Ges.m.b.H.
Wehrstrasse 29
P.O. Box 7
A-1205 **Vienna**
Tel: 35-16-21-0
Telex: 13582/135066

Hewlett-Packard Ges.m.b.H.

Wehrstrasse, 29
A-1205 **Wien**
Tel: 35-16-21
Telex: 135066

BAHRAIN

Medical Only
Wael Pharmacy
P.O. Box 648
Bahrain
Tel: 54886, 56123
Telex: 8550 WAEI GJ
Al Hamidiya Trading and
Contracting
P.O. Box 20074
Manama
Tel: 259978, 259958
Telex: 8895 KALDIA GJ

BANGLADESH

The General Electric Co. of
Bangladesh Ltd.
Bangnei House 72
Dikusha Commercial Area
Mottilhal, Dacca 2
Tel: 252415, 252419
Telex: 734

BELGIUM

Hewlett-Packard Benelux
S.A./N.V.
Avenue du Col-Vert, 1,
(Groenkraaglaan)
B-1170 **Brussels**
Tel: (02) 660 50 50
Telex: 23-494 paloben bru

BRAZIL

Hewlett-Packard do Brasil
I.e.C. Ltda.
Alameda Rio Negro, 750
Alphaville
06400 **Barueri** SP
Tel: 429-3222
Hewlett-Packard do Brasil
I.e.C. Ltda.
Rua Padre Chagas, 32
90000 **Pôrto Alegre**-RS
Tel: 22-2998, 22-5621
Hewlett-Packard do Brasil
I.e.C. Ltda.
Av. Epitaco Pessoa, 4664
22471 **Rio de Janeiro**-RJ
Tel: 286-0237
Telex: 021-21905 HPBR-BR

CANADA

ALBERTA
Hewlett-Packard (Canada) Ltd.
11620A - 168th Street
Edmonton T5M 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.
10691 Shelbridge Way
Richmond V6V 2W7
Tel: (604) 270-2277
TWX: 610-925-5059

MANITOBA

Hewlett-Packard (Canada) Ltd.
380-550 Century St.
St. James,
Winnipeg R3H 0Y1
Tel: (204) 786-6701
TWX: 610-671-3531

NOVA SCOTIA

Hewlett-Packard (Canada) Ltd.
P.O. Box 931
800 Windmill Road
Dartmouth B3B 1L1
Tel: (902) 469-7820
TWX: 610-271-4482

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 L4M 1V8
Tel: (416) 678-9430
TWX: 610-492-4246

Hewlett-Packard (Canada) Ltd.

552 Newbold Street
London N6E 2S5
Tel: (519) 686-9181
TWX: 610-352-1201

QUEBEC

Hewlett-Packard (Canada) Ltd.
275 Hymus Blvd.
Pointe Claire H9R 1G7
Tel: (514) 687-4232
TWX: 610-422-3022

FOR CANADIAN AREAS NOT LISTED:

Contact Hewlett-Packard (Canada) Ltd. in Mississauga.
CHILE
Jorge Calcagni y Cia. Ltda.
Aruro Burnie 065
Casilla 16475
Correo 9, **Santiago**
Tel: 220222
Telex: JCALCAGN

COLUMBIA

Instrumentación
Henri A. Langebaek & Kier
S.A.
Carrera 7 No. 48-75
Apartado Aéreo 6287
Bogotá, I.D.E.
Tel: 269-8877
Telex: 44400

Instrumentación

H.A. Langebaek & Kier S.A.
Carrera 63 No. 49-A-31
Apartado 54098
Medellin
Tel: 304475

COSTA RICA

Centifica Costarricense S.A.
Avenida 2, Calle 5
San Pedro de Montes de Oca
Apartado 10159
San José
Tel: 24-38-20, 24-08-0
Telex: 2367 GALTUR CR

CYPRUS

Kypricos
19 Gregorios Xenopoulos
Street
P.O. Box 1152
Nicosia
Tel: 45628/29
Telex: 3018

CZECHOSLOVAKIA

Obchodní zastupitelství v CSSR
Pismeny slyk
Posl. schránka 27
CS 118 01 **Praha** 011
CSSR
Vývojová a Provozní Zakladna
Výzkumných Ústavů v
Bechovicích
CSSR-25097 **Bechovice u**
Prahy
Tel: 89 93 41
Telex: 12133

Institute of Medical Bionics

Vyskumny Ustav Lekarskej
Bioniky
Jedlova 6
CS-88346 **Bratislava-
Kramara**
Tel: 44 55 51
Telex: 93229

DENMARK

Hewlett-Packard A/S
Datovej 52
DK-3460 **Birkeroed**
Tel: (02) 81 66 40
Telex: 37409 hpas dk
Hewlett-Packard A/S
Navervej 1
DK-8600 **Silkeborg**
Tel: (06) 82 71 68
Telex: 37409 hpas dk

ECUADOR

CYEDE Cia. Ltda.
P.O. Box 6423 CCI
Av. Eloy Alfaro 1749
Quito
Tel: 450-975, 243-052
Telex: 2548 CYEDE EO

GERMAN FEDERAL REPUBLIC

Hewlett-Packard GmbH
Vernebstentzente Frankfurt
Berner Strasse 117
Postfach 500 140
0-6000 **Frankfurt** 56
Tel: (06011) 50041
Telex: 04 13249 hpffm d
Hewlett-Packard GmbH
Technisches Büro Böblingen
Herrenberger Strasse 110
0-7030 **Böblingen**,
Württemberg
Tel: (07031) 667-1
Telex: 07265739 bbn
Hewlett-Packard GmbH
Technisches Büro Düsseldorf
Emanuel-Leutze-Str. 1
(Seestern)
D-4000 **Düsseldorf** 01
Tel: (0211) 5971-1
Telex: 085/86 533 hpdd d
Hewlett-Packard GmbH
Technisches Büro Hamburg
Kapselstading 5
D-2000 **Hamburg** 60
Tel: (040) 63804-1
Telex: 21 63 032 hpnd d
Hewlett-Packard GmbH
Technisches Büro Hannover
Am Grossmark 1
0-3000 **Hannover** 91
Tel: (0511) 46 60 01
Telex: 092 3259

Hewlett-Packard GmbH

Technisches Büro München
Eschenstrasse 5
D-8021 **Taufkirchen**
Tel: (089) 6117-1
Telex: 0524985

Hewlett-Packard GmbH

Technisches Büro Berlin
Kahnstrasse 2-4
D-1000 **Berlin** 30
Tel: (030) 24 90 86
Telex: 018 3405 hpbn d

GREECE

Kostas Kirayannis
8 Omirou Street
Athens 133
Tel: 32 30 303/32/731
Telex: 21 59 62 RKAR GR

Hewlett-Packard GmbH

Technisches Büro München
Eschenstrasse 5
D-8021 **Taufkirchen**
Tel: (089) 6117-1
Telex: 0524985

Hewlett-Packard GmbH

Technisches Büro München
Eschenstrasse 5
D-8021 **Taufkirchen**
Tel: (089) 6117-1
Telex: 0524985

Hewlett-Packard GmbH

Technisches Büro München
Eschenstrasse 5
D-8021 **Taufkirchen**
Tel: (089) 6117-1
Telex: 0524985

Hewlett-Packard GmbH

Technisches Büro München
Eschenstrasse 5
D-8021 **Taufkirchen**
Tel: (089) 6117-1
Telex: 0524985

Hewlett-Packard GmbH

Technisches Büro München
Eschenstrasse 5
D-8021 **Taufkirchen**
Tel: (089) 6117-1
Telex: 0524985

Hewlett-Packard GmbH

Technisches Büro München
Eschenstrasse 5
D-8021 **Taufkirchen**
Tel: (089) 6117-1
Telex: 0524985

Hewlett-Packard GmbH

Technisches Büro München
Eschenstrasse 5
D-8021 **Taufkirchen**
Tel: (089) 6117-1
Telex: 0524985

Hewlett-Packard France

Le Logoures
Place Romée de Villeneuve
13100 **Aix-en-Provence**
Tel: (42) 59 41 02
TWX: 410770F
Nicosia
Tel: 45628/29
Telex: 3018

Hewlett-Packard France

2, Allée de la Bourgonette
35100 **Rennes**
Tel: (99) 51 42 44
TWX: 740912F
Hewlett-Packard France
18, rue du Canal de la Marne
67300 **Schiltghelm**
Tel: (88) 83 08 10
TWX: 89041F

Hewlett-Packard France

Immeuble péricentre
rue van Gogh
59650 **Villeneuve D'Ascq**
Tel: (20) 91 41 25
TWX: 160124F
Hewlett-Packard France
Bâtiment Ampère
Rue de la Commune de Paris
B.P. 300
93153 **Le Blanc Mesnil-
Cédex**
Tel: (01) 931 88 50
Telex: 211032F

Hewlett-Packard France

Av. du Pdt. Kennedy
33700 **Mérignac**
Tel: (56) 97 01 81
Hewlett-Packard France
Immeuble Lorraine
Boulevard de France
91035 **Evry-CéDEX**
Tel: 077 96 80
Telex: 692315F

Hewlett-Packard France

23 Rue Lothaire
57000 **Metz**
Tel: (87) 65 53 50

GERMAN FEDERAL REPUBLIC

Hewlett-Packard GmbH
Vernebstentzente Frankfurt
Berner Strasse 117
Postfach 500 140
0-6000 **Frankfurt** 56
Tel: (06011) 50041
Telex: 04 13249 hpffm d
Hewlett-Packard GmbH
Technisches Büro Böblingen
Herrenberger Strasse 110
0-7030 **Böblingen**,
Württemberg
Tel: (07031) 667-1
Telex: 07265739 bbn
Hewlett-Packard GmbH
Technisches Büro Düsseldorf
Emanuel-Leutze-Str. 1
(Seestern)
D-4000 **Düsseldorf** 01
Tel: (0211) 5971-1
Telex: 085/86 533 hpdd d
Hewlett-Packard GmbH
Technisches Büro Hamburg
Kapselstading 5
D-2000 **Hamburg** 60
Tel: (040) 63804-1
Telex: 21 63 032 hpnd d
Hewlett-Packard GmbH
Technisches Büro Hannover
Am Grossmark 1
0-3000 **Hannover** 91
Tel: (0511) 46 60 01
Telex: 092 3259

Hewlett-Packard GmbH

Technisches Büro München
Eschenstrasse 5
D-8021 **Taufkirchen**
Tel: (089) 6117-1
Telex: 0524985

Hewlett-Packard GmbH

Technisches Büro München
Eschenstrasse 5
D-8021 **Taufkirchen**
Tel: (089) 6117-1
Telex: 0524985

Hewlett-Packard GmbH

Technisches Büro München
Eschenstrasse 5
D-8021 **Taufkirchen**
Tel: (089) 6117-1
Telex: 0524985

Hewlett-Packard GmbH

Technisches Büro München
Eschenstrasse 5
D-8021 **Taufkirchen**
Tel: (089) 6117-1
Telex: 0524985

Hewlett-Packard GmbH

Technisches Büro München
Eschenstrasse 5
D-8021 **Taufkirchen**
Tel: (089) 6117-1
Telex: 0524985

Hewlett-Packard GmbH

Technisches Büro München
Eschenstrasse 5
D-8021 **Taufkirchen**
Tel: (089) 6117-1
Telex: 0524985

Hewlett-Packard GmbH

Technisches Büro München
Eschenstrasse 5
D-8021 **Taufkirchen**
Tel: (089) 6117-1
Telex: 0524985

Hewlett-Packard GmbH

Technisches Büro München
Eschenstrasse 5
D-8021 **Taufkirchen**
Tel: (089) 6117-1
Telex: 0524985

Hewlett-Packard GmbH

Technisches Büro München
Eschenstrasse 5
D-8021 **Taufkirchen**
Tel: (089) 6117-1
Telex: 0524985

Hewlett-Packard GmbH

Technisches Büro München
Eschenstrasse 5
D-8021 **Taufkirchen**
Tel: (089) 6117-1
Telex: 0524985

GUAM

Medical Supply, Inc.
Suite C, Airport Plaza
P.O. Box 8947
Surabaya
Tel: 42027
Tel: 42027-5911
Tel: 646-4513

GUATEMALA

Avenida Reforma 3-48
Zona 9
Guatemala City
Tel: 316627, 314786,
66471-5, ext. 9
Telex: 4192 Teletro Gu

HONG KONG

Hewlett-Packard Hong Kong
Ltd.
11th Floor, Four Seas Bldg,
212 Nathan Rd.
Kowloon
Tel: 3-697446 (5 lines)
Telex: 36678 HX
Medical/Analytical Only
Schmidt & Co. (Hong Kong)
Ltd.
Wing On Centre, 28th Floor
Connaught Road, C.
Hong Kong
Tel: 5-455644
Telex: 74766 SCHMX HX

INDIA

Blue Star Ltd.
Sahas
414/2 Vr Savarkar Marg
Prabhadevi
Bombay 400 025
Tel: 45 78 87
Telex: 011-4093
Blue Star Ltd.
Band Box House
Prabhadevi
Bombay 400 025
Tel: 45 73 01
Telex: 011-3751

Blue Star Ltd.

Bhavdeep
Karnata Road
Ahmedabad 380 014
Tel: 43922
Telex: 012-234
Blue Star Ltd.
7 Hare Street
Calcutta 700 001
Tel: 23-0131
Telex: 021-7655

Blue Star Ltd.

Bhandari House
9 Nehru Place
New Delhi 110 024
Tel: 682547
Telex: 031-2463

Blue Star Ltd.

T.C. 7/603 'Poomina'
Maruthankuzh
Trivendrum 695 013
Tel: 65799
Telex: 0884 259

Blue Star Ltd.

11 Magdhit Road
Bangalore 560 025
Tel: 55668
Telex: 0845-430

Blue Star Ltd.

Meachchi Mandiram
XXXXV/1379-2 Mahima
Gandh Rd.
Cochin 682 016
Tel: 32069
Telex: 085-514

Blue Star Ltd.

1-11/17/1 Sarojini Devi Road
Secunderabad 500 033
Tel: 70126
Telex: 0155-459

Blue Star Ltd.

133 Kodambakkam High Road
Madras 600 034
Tel: 82057
Telex: 041-379

ICELAND

Eking Trading Company Inc.
Halnarvoti - Tryggvagötu
P.O. Box 895
IS-Reykjavik
Tel: 1 58 20/1 63 03
Telex: 052 571-5171

INDONESIA

BERCA Indonesia P.T.
P.O. Box 496/K2
Jln. Abdul Muis 62
Jakarta
Tel: 349255, 349886
Telex: 46748 BERSL IA

BERCA Indonesia P.T.

Yokogawa-Hewlett-Packard
Ltd.
29-21, Takaido-Higashi
3-chome
Suginami-ku, **Tokyo** 168
Tel: 03-331-6111
Telex: 232-2024 YHP-Tokyo

Yokogawa-Hewlett-Packard

Yokogawa-Hewlett-Packard
Ltd.
Sunomoto Semei Nagaya Bldg.
11-2 Shimosajima-cho,
Nakamura-ku, **Nagoya**, 450
Tel: 052 571-5171



SALES OFFICES

Arranged alphabetically by country (cont.)

Mushko & Company, Ltd.
10, Bazar Rd.
Sector G-6/4
Islamabad
Tel: 28264

PHILIPPINES
The Online Advanced Systems Corporation
Rico House
Amorsolo cor. Herrera Str.
Laguna Village, Makati
P.O. Box 1510
Manila
Tel: 85-35-81, 85-34-91,
85-32-21
Telex: 3274 ONLINE

RHODESIA
Field Technical Sales
45 Kelvin Road North
P.O. Box 3458
Salisbury
Tel: 705231 (5 lines)
Telex: RH 4122

POLAND
Buro Informacji Technicznej
Hewlett-Packard
Ul. Slawki 2, 6P
PL00-950 **Warszawa**
Tel: 39 59 62, 39 51 87
Telex: 81 24 53

PORTUGAL
Teletra-Empresa Técnica de Equipamentos Eléctricos S.A.I.
Rua Rodrigo de Fonseca 103
P.O. Box 2531
P-Lisbon 1
Tel: (19) 68 60 72
Telex: 12598H

Medical Only
Mundinter
Intercambio Mundial de Comercio S.A.I.
P.O. Box 2761
Avenida Antonio Augusto de Aguiar 138
P-Lisbon
Tel: (19) 53 21 31/7
Telex: 16691 munter p

PUERTO RICO
Hewlett-Packard Inter-Américas
Puerto Rico Branch Office
Calle 272,
#203 Urb. Country Club
Carolina 00630
Tel: (809) 762-7255
Telex: 345 05 14

QATAR
Nasser Trading & Contracting
P.O. Box 1563
Doha
Tel: 22170
Telex: 44339 NASSER

ROMANIA
Hewlett-Packard Reprezentanta
Bd n. Balescu 16
Bucuresti
Tel: 15 80 23/13 88 85
Telex: 10440

SAUDI ARABIA
Modern Electronic Establishment (Head Office)
P.O. Box 1228, Baghdadah Street
Jeddah
Tel: 27 798
Telex: 40035
Cable: ELECTA JEDDAH
Modern Electronic Establishment (Branch)
P.O. Box 2728
Riyadh
Tel: 62596/66232
Telex: 202049

Modern Electronic Establishment (Branch)
P.O. Box 193
Al-Khobar
Tel: 44678-44813
Telex: 670136
Cable: ELECTA AL-KHOBAR

SINGAPORE
Hewlett-Packard Singapore (Pte.) Ltd.
6th Floor, Incharge House
450-452 Alexandra Road
P.O. Box 58
Alexandra Post Office
Singapore 9115
Tel: 631788
Telex: HPSG RS 21486

SOUTH AFRICA
Hewlett-Packard South Africa (Pty.) Ltd.
Private Bag Wendywood,
Sandton, Transvaal, 2144
Hewlett-Packard Centre
Daphne Street, Wendywood,
Sandton, 2144
Tel: 802-5111/25
Telex: 8-4782

Hewlett-Packard South Africa (Pty.) Ltd.
P.O. Box 120
Howard Place,
Cape Province, 7450
Pine Park Centre, Forest Drive,
Pinalands,
Cape Province, 7405
Tel: 53-7955 thru 9
Telex: 57-0006

SPAIN
Hewlett-Packard Española, S.A.
Calle Jerez 3
E-Madrid 16
Tel: (1) 458 26 00 (10 lines)
Telex: 23515 HPE
Hewlett-Packard Española S.A.
Colonia Masarserra
Edificio Juban
c/o Costa Brava, 13
Madrid 34

Hewlett-Packard Española, S.A.
Milesados 21-23
E-Barcelona 17
Tel: (3) 203 6200 (5 lines)
Telex: 52603 hpe e
Hewlett-Packard Española, S.A.
Av Ramón y Cajal, 1
Edificio Sevilla, planta 9ª
E-Barcelona 5
Tel: 84 44 54/58

Hewlett-Packard Española S.A.
Edificio Alba II 7ª B
E-Bilbao 1
Tel: 23 83 06/23 82 06
Hewlett-Packard Española S.A.
C/Ramon Gordo 1
(Entlo.)
E-Valencia 10
Tel: 96-361.13.54/361.13.58

SRI LANKA
Metropolitan Agencies Ltd.
209/9 Union Place
Colombo 2
Tel: 35947
Telex: 1377METRO LTO CE

SUDAN
Radson Trade
P.O. Box 921
Doha
Tel: 22170
Telex: 44339 NASSER

SURINAM
Sunel Radio Holland N.V.
Grote Hofstr. 3-5
P.O. Box 155
Paramaribo
Tel: 72118, 77880

SWEDEN
Hewlett-Packard Sverige AB
Engelstgsvägen 3, Fack
S-161 **Bromma 20**
Tel: (08) 730 05 50
Telex: 10721
Cable: MEASUREMENTS Stockholm
Hewlett-Packard Sverige AB
Fritällsgatan 30
S-421 32 **Västra Frölunda**
Tel: (031) 49 09 50
Telex: 10721 via Bromma office

SWITZERLAND
Hewlett-Packard (Schweiz) AG
Zürcherstrasse 20
P.O. Box 307
CH-8052 **Schlieren-Zürich**
Tel: (01) 7305240
Telex: 53933 hpag ch
Cable: HPAG CH
Hewlett-Packard (Schweiz) AG
Château Bloc 19
CH-1219 **Le Lignon-Geneva**
Tel: (022) 96 03 22
Telex: 27333 hpag ch
Cable: HEWPACKAG Geneva

SYRIA
General Electronic Inc.
Nun Basha-Annal Ebn Kays Street
P.O. Box 5781
Damascus
Tel: 33 24 87
Telex: 11215 ITKAL
Cable: ELECTROBOR DAMASCUS

Medical only
Sawah & Co
Place Azm
B.P. 2308
Damascus
Tel: 16 367-19 697-14 268
Telex: 11304 SATACO SY
Cable: SAWAH DAMASCUS
Suleiman Hlal El Mlawi
P.O. Box 2528
Mamoun Bilar Street, 56-58
Damascus
Tel: 11 46 63
Telex: 11270
Cable: HIAL DAMASCUS

TAIWAN
Hewlett-Packard Far East Ltd.
Taiwan Branch
Bank Tower, 5th Floor
205 Tun Hsu North Road
Taipei
Tel: (02) 751-0404 (15 lines)
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.
20 Yung Sui Road
Taipei
Tel: 3615446-9 (4 lines)
Telex: 22894 SANKWANG

TANZANIA
Medical Only
International Aeradio (E.A.) Ltd.
P.O. Box 861
Dar es Salaam
Tel: 21251 Ext. 265
Telex: 41030

THAILAND
INMESA Co. Ltd.
Ecom Research Building
2538 Sukumvit Ave.
Bangkok, Bangkok
Tel: 39 32-387, 39 30-338

TRINIDAD & TOBAGO
CARTEL
Caribbean Telecoms Ltd.
P.O. Box 732
69 Frederick Street
Port-of-Spain
Tel: 62-53068

TUNISIA
Tunisie Electronique
31 Avenue de la Liberte
Tunis
Tel: 280 144
Corema
1 ter, Av. de Carthage
Tunis
Tel: 253 821
Telex: 12319 CABAM TN

TURKEY
TEKNIM Company Ltd.
Riza Sah Pehevi
Caddesi No. 7
Kavaklidere, **Ankara**
Tel: 275800
Telex: 42155
Teknim Com. Ltd.
Barbaros Bulvari 55/12
Besiktas, **Istanbul**
Tel: 613 546
Telex: 23540

E.M.A.
Mahendisk Kolektif Sirkeli
Medha Eadem Sokak 41/6
Yüksel Caddesi
Ankara
Tel: 17 56 22
Yılmaz Ozyurek
Mali Mudafaa Cad 18/6
Ankara
Tel: 25 03 09 - 17 80 26
Telex: 42576 OZEK TR

UNITED ARAB EMIRATES
Ematic Ltd. (Head Office)
P.O. Box 1841
Sharjah
Tel: 354121/3
Telex: 8136

Ematic Ltd. (Branch Office)
P.O. Box 2711
Abu Dhabi
Tel: 331370/1

UNITED KINGDOM
Hewlett-Packard Ltd.
King Street Lane
Winnereh, Wokingham
Berkshire RG11 5AR
GB-England
Tel: (0734) 784774
Telex: 84 71 78/9
Hewlett-Packard Ltd.
Founer House,
257-263 High Street
London Colney
St. Albans, Herts
GB-England
Tel: (0727) 24400
Telex: 1-89527 16
Hewlett-Packard Ltd.
Trafalgar House
Navigation Road
Aldricham
Cheshire WA14 1NU
GB-England
Tel: (061) 928 6422
Telex: 668068
Hewlett-Packard Ltd.
Lyon Court
Hereford Rise
Dudley Road
Halesowen,
West Midlands, B62 8SD
GB-England
Tel: (021) 501 1221
Telex: 339105
Hewlett-Packard Ltd.
Wedge House
799, London Road
Thornton Heath
Surrey, CR4 6XL
GB-England
Tel: (01) 684-0103/8
Telex: 946225
Hewlett-Packard Ltd.
14 Wesley St
Castleford
Yorks WF10 1AE
Tel: (0977) 510 150
Telex: 5557335
Hewlett-Packard Ltd.
Traxad House
St Mary's Walk
Maidenhead
Berkshire, SL6 1ST
GB-England
Hewlett-Packard Ltd.
Morley Road
Staplehill
Bristol, BS16 40T
GB-England
Hewlett-Packard Ltd.
South Queensferry,
West Lothian, EH30 9TG
GB-Scotland
Tel: (031) 331 1188
Telex: 72682

TAIWAN
Hewlett-Packard Far East Ltd.
Taiwan Branch
Bank Tower, 5th Floor
205 Tun Hsu North Road
Taipei
Tel: (02) 751-0404 (15 lines)
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.
20 Yung Sui Road
Taipei
Tel: 3615446-9 (4 lines)
Telex: 22894 SANKWANG

TANZANIA
Medical Only
International Aeradio (E.A.) Ltd.
P.O. Box 861
Dar es Salaam
Tel: 21251 Ext. 265
Telex: 41030

THAILAND
INMESA Co. Ltd.
Ecom Research Building
2538 Sukumvit Ave.
Bangkok, Bangkok
Tel: 39 32-387, 39 30-338

TRINIDAD & TOBAGO
CARTEL
Caribbean Telecoms Ltd.
P.O. Box 732
69 Frederick Street
Port-of-Spain
Tel: 62-53068

TUNISIA
Tunisie Electronique
31 Avenue de la Liberte
Tunis
Tel: 280 144
Corema
1 ter, Av. de Carthage
Tunis
Tel: 253 821
Telex: 12319 CABAM TN

TURKEY
TEKNIM Company Ltd.
Riza Sah Pehevi
Caddesi No. 7
Kavaklidere, **Ankara**
Tel: 275800
Telex: 42155
Teknim Com. Ltd.
Barbaros Bulvari 55/12
Besiktas, **Istanbul**
Tel: 613 546
Telex: 23540

UNITED STATES
ALABAMA
700 Century Park South,
Suite 128
Birmingham 35226
Tel: (205) 822-6802
P.O. Box 4207
8290 Whitesburg Dr.
Huntsville 35802
Tel: (205) 881-4591

ARIZONA
2336 E. Magnolia St.
Phoenix 85034
Tel: (602) 273-8000
2424 East Aragon Rd.
Tucson 85706
Tel: (602) 273-8000

ARKANSAS
Medical Service Only
P.O. Box 5646
Brady Station
Little Rock 72215
Tel: (501) 376-1844

CALIFORNIA
1579 W. Shaw Ave.
Fresno 93771
Tel: (209) 224-0582
1430 East Orangehurst Ave.
Fullerton 92631
Tel: (714) 870-1000
5400 West Rosecrans Blvd.
P.O. Box 92105
World Way Postal Center
Los Angeles 90009
Tel: (213) 970-7500
Telex: 910-325-6608

INDIANA
7301 North Shadeland Ave.
Indianapolis 46250
Tel: (317) 842-1000
Telex: 810-260-1797

IOWA
2415 Heinz Road
Iowa City 52240
Tel: (319) 351-1020
10 KENTUCKY
10710 Linn Station Road
Suite 525
Louisville 40223
Tel: (502) 426-0100

LOUISIANA
P.O. Box 1449
3229-39 Williams Boulevard
Kenner 70062
Tel: (504) 443-6201

3939 Lankerstern Boulevard
North Hollywood 91604
Tel: (213) 877-1282
Telex: 910-499-2671
3200 Hilview Av
Palo Alto, CA 94304
Tel: (408) 988-7000
646 W. North Market Blvd.
Sacramento 95834
Tel: (916) 929-7222

9606 Aero Drive
P.O. Box 23333
San Diego 92123
Tel: (714) 279-3200
363 Brookhollow Dr.
London Colney
St. Albans, Herts
GB-England
Tel: (0727) 24400
Telex: 1-89527 16

Hewlett-Packard Ltd.
Trafalgar House
Navigation Road
Aldricham
Cheshire WA14 1NU
GB-England
Tel: (061) 928 6422
Telex: 668068
Hewlett-Packard Ltd.
Lyon Court
Hereford Rise
Dudley Road
Halesowen,
West Midlands, B62 8SD
GB-England
Tel: (021) 501 1221
Telex: 339105

FLORIDA
P.O. Box 24210
2727 N.W. 62nd Street
FL. Lauderdale 33309
Tel: (305) 973-2600
4080 Woodcock Drive #132
Broward Building
Jacksonville 32207
Tel: (904) 398-0663
P.O. Box 13910
6177 Lake Elenor Dr.
Orlando 32809
Tel: (305) 859-2900
P.O. Box 12826
Suite 5, Bldg. 1
Office Park North
Panascote 32575
Tel: (904) 476-8422
110 South Hoover Blvd.
Suite 120
Tampa 33609
Tel: (813) 872-0900

GEORGIA
P.O. Box 105005
450 Interstate North Parkway
Atlanta 30348
Tel: (404) 955-1500
Tel: 810-766-4890
Medical Service Only
Augusta 30903
Tel: (404) 736-0592
P.O. Box 2103
1172 N. Davis Drive
Warner Robins 31098
Tel: (912) 922-0449

HAWAII
2875 So. King Street
Honolulu 96826
Tel: (808) 955-4455

ILLINOIS
2111 Prospect Rd.
Bloomington 61701
Tel: (312) 663-0383
5201 Tolview Dr.
Rolling Meadows 60008
Tel: (312) 255-9800
Tel: (910) 487-2260

INDIANA
7301 North Shadeland Ave.
Indianapolis 46250
Tel: (317) 842-1000
Telex: 810-260-1797

IOWA
2415 Heinz Road
Iowa City 52240
Tel: (319) 351-1020
10 KENTUCKY
10710 Linn Station Road
Suite 525
Louisville 40223
Tel: (502) 426-0100

LOUISIANA
P.O. Box 1449
3229-39 Williams Boulevard
Kenner 70062
Tel: (504) 443-6201

MARYLAND
7121 Standard Drive
Parkway Industrial Center
Hanover 21076
Tel: (301) 796-7700
Telex: 710-862-1943
2 Choke Cherry Road
Rockville 20850
Tel: (301) 948-6370
Tel: (301) 948-6370
Telex: 710-828-9684

MASSACHUSETTS
32 Hartwell Ave.
Lexington 02173
Tel: (617) 861-8960
Telex: 710-326-6904

MICHIGAN
23855 Research Drive
Farmington Hills 48024
Tel: (313) 476-6400
724 West Centre Ave.
Kalamazoo 49002
Tel: (616) 323-8362

MINNESOTA
2400 N. Prior Ave.
St. Paul 55113
Tel: (612) 636-0700
111 Zeta Drive
Pittsburgh 15238
Tel: (412) 782-0400
Telex: 510-660-2670

MISSISSIPPI
322 N. Meri Plaza
Jackson 39206
Tel: (601) 982-9363

MISSOURI
1131 Colorado Ave
Kansas City 64137
Tel: (816) 763-8000
Telex: 910-771-2087
1024 Executive Parkway
St. Louis 63141
Tel: (314) 878-0200

NEBRASKA
Medical Only
7101 Mercy Road
Suite 101
Omaha 68106
Tel: (402) 392-0948

NEVADA
Las Vegas
Tel: (702) 736-6610

NEW JERSEY
Crystal Brook Professional Building
Route 35
Eatontown 07724
Tel: (201) 542-1384
W. 120 Century Rd.
Paramus 07652
Tel: (201) 265-5000
Telex: 710-990-4951

NEW MEXICO
11300 Lomas Blvd., N.E.
Albuquerque 87123
Tel: (505) 292-1330
Telex: 910-989-1185
156 Wyatt Drive
Las Cruces 88001
Tel: (505) 526-2484
Telex: 910-9983-0550

NEW YORK
6 Automation Lane
Computer Park
Albany 12205
Tel: (518) 458-1550
Telex: 710-444-4961
650 Perinton Hill Office Park
Fairport 14450
Tel: (716) 223-9950
Telex: 510-253-0092
No. 1 Pennsylvania Plaza
55th Floor
34th Street & 8th Avenue
New York 10001
Tel: (212) 971-0800
5858 East Moley Road
Syracuse 13211
Tel: (315) 455-2486

OHIO
Medical/Computer Only
9920 Carver Road
Cincinnati 45242
Tel: (513) 891-9870
16500 Sprague Road
Cleveland 44130
Tel: (216) 243-7300
Telex: 810-423-9430

OKLAHOMA
P.O. Box 32008
6301 N. Meridan Avenue
Oklahoma City 73112
Tel: (405) 721-0200
9920 E. 42nd Street
Suite 121
Tulsa 74145
Tel: (918) 665-3300

OREGON
17890 S.W. Lower Boones Ferry Road
Tualatin 97062
Tel: (503) 620-3350

PENNSYLVANIA
1021 Bn Avenue
King of Prussia Industrial Park
King of Prussia 19406
Tel: (215) 265-7000
Telex: 510-660-2670

TEXAS
4171 North Mesa
Suite C110
El Paso 79902
Tel: (915) 533-3555
P.O. Box 42816
10535 Harwin St
Houston 77036
Tel: (713) 776-8400
Lubbock
Medical Service Only
Tel: (806) 799-4472
P.O. Box 1270
201 E. Arapaho Rd.
Richardson 75081
Tel: (214) 231-6101
205 Billy Mitchell Road
San Antonio 78226
Tel: (512) 434-8241

UTAH
2160 South 3270 West Street
Salt Lake City 84119
Tel: (801) 972-4711

VIRGINIA
P.O. Box 9669
2914 Hungary Spring Road
Richmond 23228
Tel: (804) 285-3431
Computer Systems/Medical Only
Airport Executive Center
Suite 302
5700 Thurston Avenue
Virginia Beach 23455
Tel: (804) 460-2471

WASHINGTON
Beliefed Office Pk.
1203 - 114th Ave. S.E.
Ballou 98004
Tel: (206) 454-3971
Telex: 910-443-2446
P.O. Box 4010
Spokane 99202
Tel: (509) 535-0864

WEST VIRGINIA
Medical/Analytical Only
4604 Mac Corike Ave., S.E.
Charleston 25304
Tel: (304) 925-0492

WISCONSIN
150 South Sunny Slope Road
Brookfield 53005
Tel: (414) 784-8800

962 Crupper Ave.
Columbus 43229
Tel: (614) 436-1041
330 Progress Rd.
Dayton 45449
Tel: (513) 859-8202

OKLAHOMA
P.O. Box 32008
6301 N. Meridan Avenue
Oklahoma City 73112
Tel: (405) 721-0200
9920 E. 42nd Street
Suite 121
Tulsa 74145
Tel: (918) 665-3300

OREGON
17890 S.W. Lower Boones Ferry Road
Tualatin 97062
Tel: (503) 620-3350

PENNSYLVANIA
1021 Bn Avenue
King of Prussia Industrial Park
King of Prussia 19406
Tel: (215) 265-7000
Telex: 510-660-2670

MISSISSIPPI
322 N. Meri Plaza
Jackson 39206
Tel: (601) 982-9363

MISSOURI
1131 Colorado Ave
Kansas City 64137
Tel: (816) 763-8000
Telex: 910-771-2087
1024 Executive Parkway
St. Louis 63141
Tel: (314) 878-0200

NEBRASKA
Medical Only
7101 Mercy Road
Suite 101
Omaha 68106
Tel: (402) 392-0948

NEVADA
Las Vegas
Tel: (702) 736-6610

NEW JERSEY
Crystal Brook Professional Building
Route 35
Eatontown 07724
Tel: (201) 542-1384
W. 120 Century Rd.
Paramus 07652
Tel: (201) 265-5000
Telex: 710-990-4951

NEW MEXICO
11300 Lomas Blvd., N.E.
Albuquerque 87123
Tel: (505) 292-1330
Telex: 910-989-1185
1