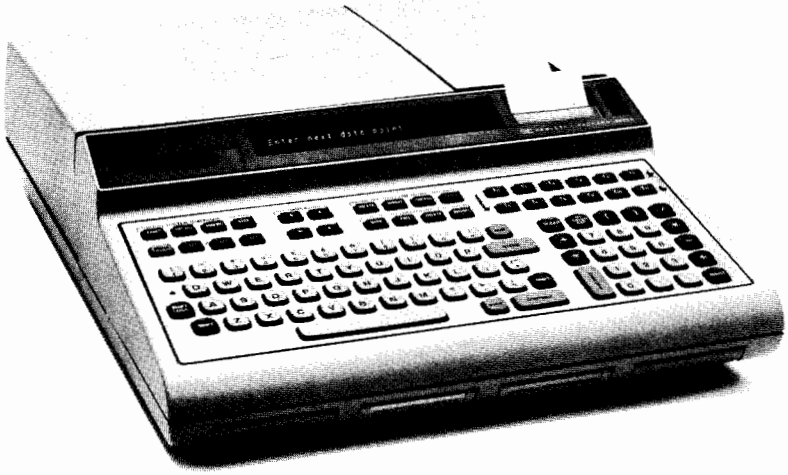
Hewlett-Packard 9825A Calculator
Extended I/O Programming

Extended I/O Programming



The HP 9825A Calculator

Hewlett-Packard Calculator Products Division
P.O. Box 301, Loveland, Colorado 80537, Tel. (303) 667-5000
(For World-wide Sales and Service Offices see back of manual.)

Manual Changes

HEWLETT·PACKARD

HP 9825A Calculator
Extended I/O Programming Manual

(For Manual P/N 09825-90025, Dated October 1, 1976)

Page 8:

Replace the last paragraph with this:

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.

Page 16:

Change the last line of the address sequence from:

- Listen address 1 (on bus 7) *to*
- Listen address 11 (on bus 7)

Page 20:

Change the fourth line to read:

- trfr source ; destination[; character count[; last character]](see page 72)

Delete the sixth line.

Page 22:

Replace the first printout with this:

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

Replace the third printout with this:

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

Page 30:

Add to The Poll Configure Statement:

NOTE

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

Page 37:

Change the example program lines –

```
9: cmd"timer","time";cmd"clock","C"
18: cmd"dmm","100kohm";red"dmm",D
```

Page 43:

Add the following note to the bottom of the page:

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
```

Page 48:

Delete the paragraph following the conversion protocol diagram.

Page 55:

Change paragraph 3, last sentence, from:

ldk to lkd

Page 57:

Replace line 10 of the example program to:

```
10: if I>1000;gto 21
```

Page 60

Change program lines 55 and 56 to:

```
55: "TALK":wrt 731,A[I];ret
56: "LISTEN":red731,A$•••;ret
```

Page 73:

Add this paragraph to the bottom of page 73.

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 Appendix; The HP Interface Bus.

Page 76:

Replace the last paragraph and example with this:

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.

Page 76, Step 5:

Change (buffer status remains unchanged) *to:*

(buffer status = 17)

Page 77:

Change paragraph 3; last sentence should begin:

For byte-type buffers... .

Change the last line of the next to last paragraph to read:

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.

Table of Contents

Chapter 1: General Information

Description 1
Inspection and Installation 2
Syntax 2
Unnecessary Parameters 2
Requirements 3

Chapter 2: Binary Operations

Introduction 5
Binary Representation 5
Decimal/Octal Mode Statements 6
Decimal/Octal Conversion Functions 7
The Binary AND Function 7
The Exclusive OR Function 7
The Inclusive OR Function 8
The Complement Function 8
The Rotate Function 8
The Shift Function 9
The Add Function 10
The Bit Function 10

Chapter 3: HP-IB Operations

Introduction 13
Bus Messages 13
Transfer Parameters 16
Extended Bus Addressing 16
Non-Active Controller Address Parameter 17
The Device Statement 17
Multiple Listeners 18
Data Messages 19
Sending Data Messages 19
Receiving Data Messages 20
Sending the Trigger Message 20
Sending the Clear Message 21
Sending the Remote Message 22
Sending the Local Message 23
Sending the Local Lockout Message 23
Sending the Clear Lockout/Set Local Message 24

Service Requests and Polling 24
Sending the Require Service Message 25
Receiving the Require Service Message 25
Sending the Status Byte Message 26
Serial Polling or Receiving the Status Byte Message 26
Sending the Status Bit Message 28
Parallel Polling or Receiving the Status Bit Message 29
The Poll Configure Statement 30
The Poll Unconfigure Statement 30
Sending the Pass Control Message 30
Sending the Abort Message 31
Sample Application 32
The Command Statement 35
The Equate Statement 37
Extended Read Status 38

Chapter 4: Potpourri

Autostart 41
The Timeout Statement 42
The On Error Statement 42
The Conversion Table Statement 44
Substring Conversion Tables 46
The Parity Statement 47
Conversion Protocol 48
Interface Control Operations 48
The Write Interface Statement 49
The Read Interface Function 50
The I/O Flag Function 50
The I/O Status Function 50
I/O Drivers Example 51

Chapter 5: Interrupt Control

Introduction 53
The Programmable Interrupt Scheme 53
Vectored Interrupt 54
The On Interrupt Statement 55
The Enable Interrupt Statement 56
The Interrupt Return Statement 57
The HP-IB Interrupt Control 58
Abortive Interrupts 61
Interface Control Bits 63

Interrupt Lockouts 64
Variables with Interrupt Service Routines 64

Chapter 6: Buffered I/O

Introduction 67
The Buffered I/O Scheme 67
Automatic Interrupt 68
Buffer Types 68
The Interrupt Buffer 69
The Fast Read/Write Buffer 69
The DMA Buffer 70
Buffer Underflow and Overflow 70
The Buffer Statement 70
The Transfer Statement 72
Data Output 72
Data Input 73
I/O Buffer Status 74
Buffer Pointers 75
String Variables as Buffers 77
Inverted Data 77
Buffered I/O Example 79
Demonstration Programs 80

Appendices

The HP Interface Bus 85
HP-IB Lines and Operations 85
Interface Functions 88
Extended I/O Status Conditions 90
ASCII Character Codes 91
Buffered I/O Benchmarks 92
Extended I/O Syntax Summary 97
Syntax Conventions 97
Binary Statements and Functions 97
HP-IB Statements 98
The Timeout Statement 100
The On Error Statement 100
The Conversion Table Statement 100
The Parity Statement 101
Interface Control Operations 101
Interrupt Control Statements 101
Buffered I/O Statements 102

Extended I/O ROM Error Messages 103
General I/O ROM Error Messages 105
HP Sales and Service Offices 108

Figures

ROM Installation 2
A Typical HP-IB System 32
Conversion Protocol 48
The I/O Buffer Scheme 67
HP-IB Signal Lines 85

Tables

Sample HP-IB Operations with the 9825A Calculator 15
Calculator Response When Not Active Controller 31
I/O Buffer Types 71
HP-IB Command and Address Codes 88
HP-IB Interface Functions 88
Functions Used By Each Bus Message 89
Extended I/O Status Conditions 90
ASCII Character Codes 91
Buffered I/O Benchmark Times 92

Chapter 1
General Information

Description

The Extended I/O ROM expands the capabilities of the General I/O ROM operations and adds 42 new statements and functions. This manual describes Extended I/O operations in this order:

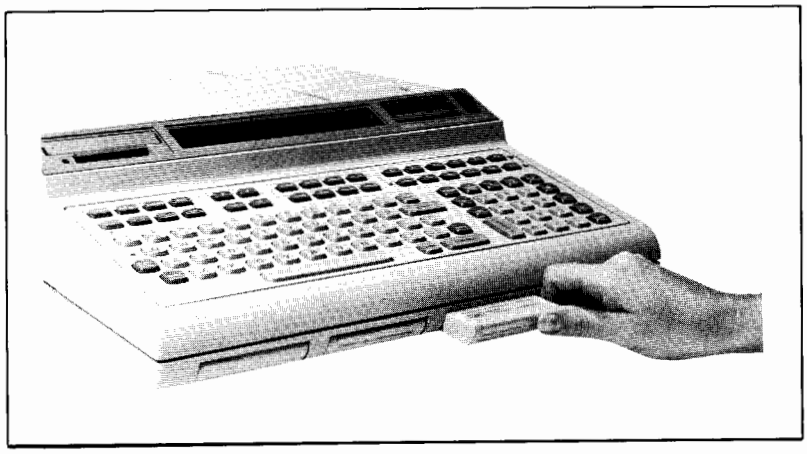
- **Bit Manipulations** – There are 12 instructions for manipulating and testing 16-bit binary values.
- **HP-IB Control** – In addition to extending General I/O ROM operations, there are 14 statements and functions to provide complete control of any current HP Interface Bus system.
- **Potpourri** – 11 additional statements and functions include a 256-character conversion table, parity checking and generation, error recovery, direct interface access, and a timeout routine. An Autostart routine, which automatically loads and runs a program, is also covered.
- **Interrupt Control** – Three statements allow you to program interrupt routines for servicing peripheral devices. An interrupt-priority scheme based on interface select codes is used.
- **Buffered I/O** – Areas of the read/write memory can be allocated as "buffers" for use in transferring data under one of three special schemes: interrupt I/O, fast read/write, or direct memory access (DMA). The transfer (tfr) statement is used to exchange data between the buffer and the peripheral, while General I/O ROM operations are used to exchange data between the buffer and calculator variables.

The Extended I/O ROM uses 94 bytes of calculator read/write memory (RWM) when installed.

The Extended I/O ROM is packaged with one or two other ROMs in a single ROM card. This manual describes Extended I/O operations only. Another manual is furnished with the card to describe operations of each additional ROM.

Inspection and Installation

Refer to the HP 9825A System Test Booklet for the procedure to verify operation of each ROM. The ROM card can be plugged into any one of the four ROM slots located on the bottom front of the calculator, as shown in the next photo.



ROM Installation

To install the card, first turn the calculator off. With the ROM label right side up, slide the card through the ROM slot door; press it in until the front of the card is even with the front of the calculator. Then turn the calculator on.

Syntax

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

`dot matrix` – All items printed in dot matrix are required exactly as shown.

`[ ]` – All items in square brackets are optional, unless the brackets are printed in dot matrix.

`...` – Dots indicate that successive parameters are allowed, when each is separated by a comma.

Unnecessary Parameters

Certain Extended I/O statements will allow the specification of more than the required number of parameters. If any unnecessary parameters are given, they will be ignored at execution time; an error message will not indicate these unnecessary parameters.

See the Appendix for a summary of all Extended I/O statements and functions.

Requirements

The Extended I/O ROM requires that a General I/O ROM also be installed. If not, Extended I/O operations may be keyed in and stored, but an error will occur if an attempt is made to execute them.

Before using this manual, you should be familiar with the 9825A Calculator and the HPL programming language described in the HP 9825A Operating and Programming Manual. Since Extended I/O operations are based on the General I/O ROM operations, you should also have read the General I/O Programming Manual.

Chapter 2
Binary Operations

Introduction

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 complement[1] 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.

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

Decimal/Octal Mode Statements

The Extended I/O ROM allows you to set the calculator in either of two number modes, octal[1] (base 8) or decimal (base 10).

Set Octal Mode Syntax: `moct`
Set Decimal Mode Syntax: `mdec`

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:

```
moct:rdb(3)                135.00
mdec:rdb(3)                 93.00
```

I/O Parameters Affected by the Decimal or Octal Mode

General I/O ROM:

- Values input or output under the b format spec.
- Values output using the write binary (wtb) or write control (wtc) statements.
- Values returned using the read binary (rdb) or read status (rds) functions.
- Conversion codes used in the conversion (conv) statement.

Extended I/O ROM:

- Values used in all binary functions (band, ior, eor, cmp, rot, shf, add, and bit).
- Byte parameters with the require service (rqs), poll configure (polc), and enable interrupt (eir) statements.
- Byte returned with the parallel pol (pol) function.
- Character parameter in transfer (tfr) statement.
- Register number and expression parameters in the write interface (wti) statement.
- Data returned with the read interface (rdi) function.

[1]Octal notation is explained in the Appendix of the General I/O Programming Manual.

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)

The Binary AND Function

Syntax:

band (expression$_A$, expression$_B$)

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 disp A
24→B disp B
band(A,B)
```

```
 20          (0000 0000 0001 0100)
 24          (0000 0000 0001 1000)
 16          (0000 0000 0001 0000)
```

The Exclusive OR Function

Syntax:

eor (expression$_A$, expression$_B$)

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

The Inclusive OR Function

Syntax:

ior (expression$_A$, expression$_B$)

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:

```
moct: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)

The Complement Function

Syntax:

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:

```
moct:127→A:dsp A
```
127 (0 000 000 001 010 111)

```
cmpA
```
177650 (1 111 111 110 101 000)

The Rotate Function

Syntax:

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 negative. 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))
```

The Shift Function

Syntax:

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)
```

The Add Function

Syntax:

add (expression$_A$, expression$_B$)

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
```

The 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":

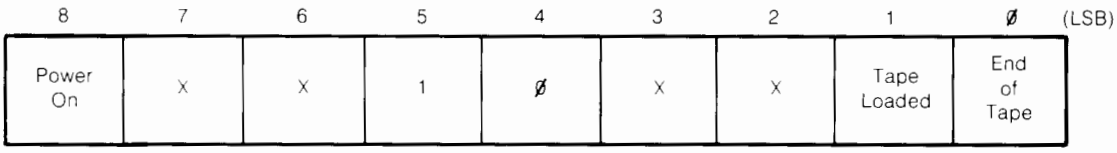
```
mdec:bit("01000001",A)
1 (true)
```

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:



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 A$[10,50]
1: rds(2)→A
2: if bit("1xxxxx1x",A)=0;jmp -1
3: red 2,A$
4: if bit(0,A);jmp 2
5: if (I+1→I)>50;gto 1
6: end
```

More examples using the bit function are in Chapter 3.

Chapter 3
HP-IB Operations

Introduction

The Extended I/O ROM provides the statements and functions for complete control of HP Interface Bus (HP-IB) systems. In addition to using General I/O ROM operations to transmit data and control instructions, the calculator can now transmit all bus control messages (e.g., trigger, clear, local, and remote), conduct serial and parallel polls, and pass bus control to another device on the same bus. The user can assign device names to be used in place of select code parameters. When the calculator is not the active controller, it can transmit the Require Service message and automatically respond to serial and parallel polls. The command (cmd) statement is provided for compatibility with other HP calculator programs.

This chapter describes all Extended I/O bus operations. It assumes that you are familiar with the General I/O ROM operations, as described in the General I/O Programming Manual. You should also know the bus operations for each device in your HP-IB system. Refer to their operating manuals.

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 in the Appendix of this manual.

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 calculator is assumed to be the system controller). The addition of an Extended I/O ROM, however, provides all 12 bus messages to permit complete bus capability. Messages can be transferred among:

- Device and Device(s)
- Controller and Device(s)
- Controller and 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. Refer to the Appendix for more details.

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 31 summarizes calculator response to bus messages when it is not in active control of the bus.

Sample HP-IB Operations with the 9825A Calculator

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.	`cmd 7,"?U-","L10"` `cmd "dvm","L10"`
	Output data to multiple listeners.	`wrt "dvm,printer","L10"` `cmd "?UK","L10"`
	Transfer data from device to device.	`cmd "?K"`
Trigger	Send a Group Execute Trigger to all instruments.	`tra 7`
	Send a GET to selected devices.	`tra 711`
Clear	Clear all devices.	`clr 7`
	Clear selected device.	`clr 711`
Remote	Enable remote mode on all devices. Switching the calculator on also sends a Remote message.	`rem 7`
	Set remote mode on selected device.	`rem 711`
Local	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. (RESET) 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. (RESET) also sends an abort message.	`cli 7`

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

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 General I/O Programming Manual. 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 31[1] (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 calculator's preset talker/listener address and the specified address code in the appropriate order on the bus. When the calculator is the active controller, this address sequence is preceded by the bus Unlisten command to clear all listeners previously set. For example, this statement `wrt 711,3,...` sends this address sequence before sending data:

- Unlisten command
- Calculator talk address
- Listen address 1 (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)

[1]Address 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, the statement `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 Parameter

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.

The Device Statement

Syntax:

`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 (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 21 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:

- wrt select code [, format no.][; expression $_1$[; expression $_2$...]] (see General I/O)
- wtb select code ; expression $_1$[; expression $_2$...] (see General I/O)
- list [#select code [, non-zero digit]][; line nos.] (see General I/O)
- tfr source ; destination [; character count [; last character]] (see page 72)
- cmd select code ; "address characters " [; "data characters "]

  or

  cmd "device name(s) " or select code [; "data characters "]

  (see page 35)

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).

Receiving Data Messages

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

- red select code [, format no.]; variable $_1$ [; variable $_2$...] (see General I/O)
- rdb (select code) (see General I/O)
- rds (select code) (see General I/O and page 38)
- tfr source; destination [; character count [; last character]] (see page ---)
- pol (select code) (see page 29)
- cmd select code; "address characters" [; "data characters"]

  or

  cmd "device name(s)" or select code [; "data characters"] (see page 72).

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.

Sending the Trigger Message

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.

Syntax:

trg select code [device address]

Specifying only the interface card's select code (e.g., trg 7) outputs a Trigger message to all devices currently addressed to listen on the bus. Including a device address (e.g., trg 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","T0F1M3E"
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$[50,20];0→A;1→I→N
1: dev "dmm",722
2: wrt "dmm","F0R6T1M3E"
3: red "dmm",A$[I]
4: wrt "dmm","E"
5: if (I+1→I)<51;jmp -2
*30105
```

Sending the Clear Message

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.

Syntax:

`clr` select code [device address]

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 name "gen" and "clock":

```
0: wtb "gen,clock"
1: clr 7
```

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

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

Sending the Remote Message

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

Syntax:

rem select code [device address]

The Remote message is automatically output whenever the calculator is switched on or (RESET) 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 "gen",706,"clock",720
1: wtb "gen,clock"
2: clr 7
```

All devices on the bus which can respond to remote enable (REN) are set to remote by this line:

```
2: rem 7
```

Sending the Local Message

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.

Syntax:

```
lcl select code with device address
```

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"
*15637
```

Sending the Local Lockout Message

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.

Syntax:

```
llo select code
```

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
```

Sending the Clear Lockout/Set Local Message

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.

Syntax:

lcl select code

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 with the Extended I/O ROM: 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

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 calculator is not the active controller (either it passed control[1] 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.

Syntax:

rqs select code, status byte

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.

[1]See pass control statement on page *30*.

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.

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.

Serial Polling or Receiving the Status Byte Message

The serial poll is so named because the calculator 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 calculator has serviced each device that has been requesting service, the SRQ line is cleared (assuming no new requests are received).

To serial poll for checking the presence of a service request use the read status (rds) function to check interface card status. As shown in the General I/O Programming Manual, 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 2. 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.

[1]For 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:

bit	7	6	5	4	3	2	1	Ø (LSB)
	Ø	Service Request	Cover Off	Data Latch Ready	Printer Not Ready	Buffer Space	Ø	Ø

- Bits 0, 1 and 7 are always logical zero.
- Bit 2 is a logical 1 when the buffer is within 16 characters of being filled. It remains true until the buffer is empty. Bit 2 is a logical 0 when the buffer is empty.
- Bit 3 is a logical 1 when the printer is not ready to accept data (e.g., cover interlock broken, self test) or if the controller is not ready.
- Bit 4 is a logical 1 when the printer's HP-IB interface assembly is ready to accept data.
- Bit 5 is a logical 1 when the printer's front cover is off.
- Bit 6 is a logical 1 when the printer has sent a Require Service message.

9871A Status Byte[1]

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("xxx01xxx",A);dsp "PRINTER NOT READY";wait 100;gto 2
6: if bit(5,A);dsp "PRINTER COVER OFF";stp
```

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).

[1]For 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.

Parallel Polling or Receiving the Status Bit Message

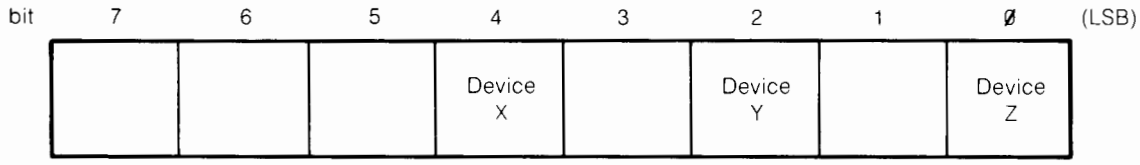
Parallel polling enables the calculator to check the status of up to eight devices at a time. This is possible since each device with parallel poll capability is pre-programmed to output one status bit when parallel polled. The bit is output as the Status Bit message. The status bit for each device indicates 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:

Syntax:

pol (select code)

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.

The Poll Configure Statement

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.

Syntax:

`polc` select code with device address, status byte

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.

The Poll Unconfigure Statement

Syntax:

`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.

Sending the Pass Control Message

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.

Syntax:

`pct` select code with device address

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 Local Local Lockout Clear Lockout/Set Local Require Service	No Response.
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.

[1]See "Extended Read Status" and Chapter 5.

Sending the Abort Message

Syntax:

```
cli select code
```

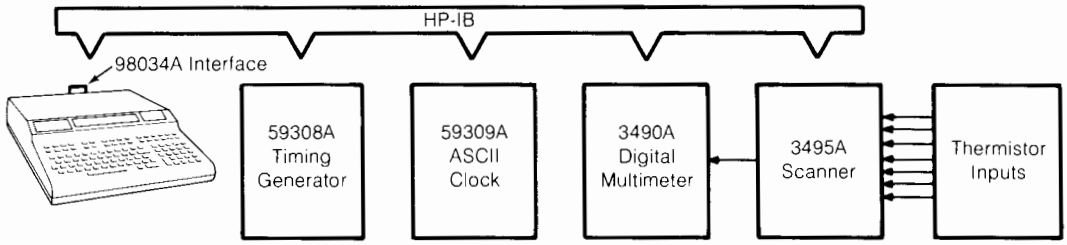
The clear interface statement sends the Abort message to halt all bus operations and return bus control to the calculator. Pressing (RESET) also outputs the Abort message.

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 9825A 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,"R7F1S1T1M3E",z
9: wrt "timer.1",N;wtb "clock","C"
10:
11: red "clock",T;1→I
12: 100frc(T/100)→T[I];int(T/100)→T;jmp (I+1→I)=6
13: wrt 16.2,"Date: ",T[5],"/",T[4],"Time: ",T[3],T[2],T[1]
14: wrt 16.3
15: 1→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+1→C)<8;gto 16
21: if bit(6,rds("timer"))=1;trg "timer,clock";gto 10
22: jmp -1
23: end
*8694
```

- 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.

```
Date: 08/09
Time: 03:25:06

Channel   Temp(C)
-------------------
    1      16.9139
    2      16.9247
    3      16.9233
    4      16.9202
    5      16.9115
    6      16.9200
    7      16.9169
```

- 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.

The 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.[1]

Syntax:

```
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 (calculator)
```

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="
```

[1]ASCII address characters are described in the General I/O Programming Manual and in each device's operating manual.

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

```
0: dim A$[50,20];0→A;1→I→N
1:
2: cmd 7,"?5V","F0R6T1M3E"
3: cmd 7,"?U6";red 7,A$[I]
4: cmd 7,"?5V","E"
5: if (I+1→I)<51;jmp -2
6: A+val(A$[N])→A
7: if (N+1→N)<51;jmp -1
8: prt A/50
9: end
*3790
```

Using Device Names

```
0: dim A$[50,20];0→A;1→I→N
1: dev "calc",721,"dmm",722
2: cmd "dmm","F0R6T1M3E"
3: cmd "calc";red 7,A$[I]
4: cmd "dmm","E"
5: if (I+1→I)<51;jmp -2
6: A+val(A$[N])→A
7: if (N+1→N)<51;jmp -1
8: prt A/50
9: end
*27174
```

- Line 0 – Sets up a 50-element string array and initializes three varaibles.
- 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 in Chapter 4 of the General I/O Programming Manual. For another program using the command statements, see the following pages.

The Equate Statement

Syntax:

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 (String ROM needed). 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 shown on page 33 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 calculator is reset. The rest of the program is as described on page 32.

```
0: dim T[5],A$[9];"T    E6ASK"→A$
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: str(N)→A$[2,4]
4: equ "time",A$,"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: wrt "timer","time";cmd "clock","C"
10:
11: red "clock",T;1→I
12: 100frc(T/100)→T[I];int(T/100)→T;jmp (I+1→I)=6
13: wrt 16.1,"Date: ",T[5],"/",T[4],"Time: ",T[3],T[2],T[1]
14: wrt 16.2
15: 1→C
16:
17: wrt "scan.4",C
18: wrt "dmm","100kohm";red "dmm",D
19: 3807/(log(D)+9.39)-273→D
20: wrt 16.3,C,D;if (C+1→C)<8;gto 16
21: if bit(6,rds("timer"))=1;trg "timer,clock";gto 10
22: jmp -1
23: end
*19634
```

Extended Read Status

Syntax:

rds (select code [, variable $_1$[, variable $_2$[, variable $_3$]]]) → variable $_4$

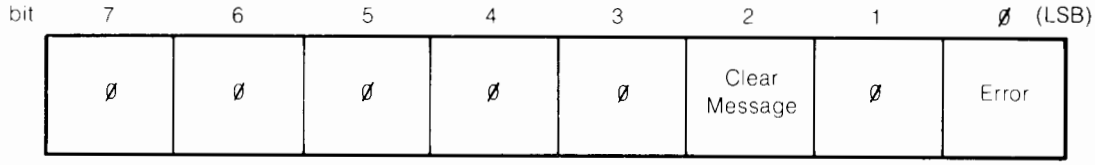
or

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.

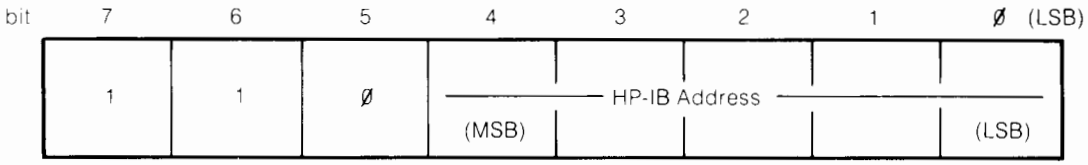
When only the interface select code is specified (2 thru 15), up to four variables can be specified to return status bytes from the 98034A Interface. The fourth variable returns the same status byte described in the General I/O Programming Manual. The interface status bytes are shown on the next pages. The first status byte is returned to variable$_1$, the second status byte is returned to variable$_2$, 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 beginning on page 26.



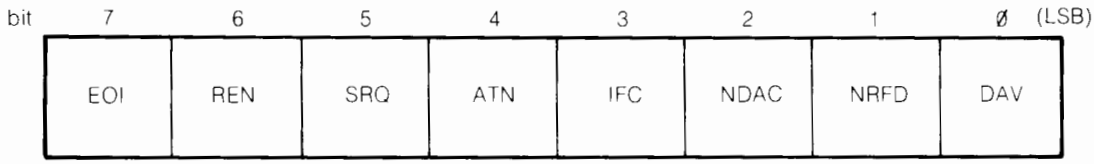
- Bit 0: Is 1 when error is detected.
- Bit 2: Is 1 when a Clear (DCL) message has been received.

First Status Byte



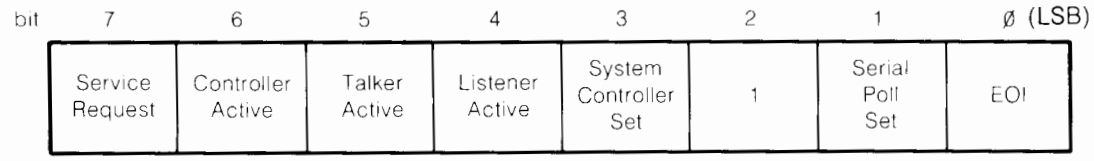
- Bits 0 thru 4: Indicate the bus address set on the 98034A Interface.

Second Status Byte



- Logical 1 indicates that the corresponding bus control line is true. These lines are described in the 98034A Installation and Service Manual.

Third Status Byte



- Bit 0: Is 1 when the EOI (end of data) line has been set true. This bit is cleared by a read status (rds) operation.
- Bit 1: Is 1 when the Serial Poll function is set.
- Bit 2: Is always 1.
- Bit 3: Is 1 when the System Controller function is set.
- Bit 4: Is 1 when the calculator is an active listener.
- Bit 5: Is 1 when the calculator is an active talker.
- Bit 6: Is 1 when the calculator is an active controller.
- Bit 7: Is 1 when an instrument has sent a Require Service message.

Fourth Status Byte

Chapter 4
Potpourri

This chapter describes additional operations available with the Extended I/O ROM.

Autostart

When the Extended I/O ROM is plugged in and a tape cartridge is installed, the calculator automatically executes a load program 0 (`ldp0`) statement from track 0 immediately after the calculator is switched on.

The autostart routine permits the calculator 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 calculator 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.

The Timeout Statement

Syntax:

`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, [STOP] 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.

The On Error Statement

Syntax:

`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 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.

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

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 (RESET , run command, erase command, or switch power on) also cancels the routine.

> **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
*10921
```

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.

The Conversion Table Statement

Syntax:

ctbl [string variable name]

The conversion table statement assigns a pre-dimensioned string variable (String ROM needed) to act as a conversion table for all General I/O and Extended I/O ROM input and output operations. This statement sets up a conversion table completely separate from the General I/O ROM's conversion 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 EIA[1] 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→I
```

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:

[1]Electronic 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: 1→I
17: red 3,X,Y
18: prt X,Y;spc 2
19: if (I+1→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,...

The Parity Statement

Syntax:

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.

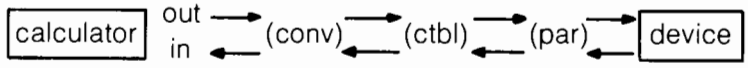
For example, the following program sequence inputs 50 data items from a paper tape punched in ASCII with even parity. If an error occurs in line 1, the program jumps to the error recovery routine, disables parity, and checks the error recovery variables. If error E7 has occurred, the calculator displays 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+1→I)=50
2: gto 8
3: "error":par 0
4: if rom=69 and ern=7;dsp "BAD DATA";beep;stp ;gto 0
5: rom→R;if R=0;32→R
6: fmt b,2f.0
7: wrt 0,"ERROR ",R,ern," 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

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 order shown below. 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.

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!

The Write Interface Statement

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.

Syntax:

wti 0, select code
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 (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.

The Read Interface Function

Syntax:

```
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.

The I/O Flag Function

Syntax:

```
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.

The I/O Status Function

Syntax:

```
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 iosS=0;gsb "down"
5: if iofS=0;jmp 0
6: if bit(2,rds(S));cmpD→D
7: wti 4,D
8: wti 7,0
9: ret
*3651
```

This program example imitates the input drivers:

```
0: "Input subroutine":
1: ent "Select code?",S
2: wti 0,S
3: if iosS=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
*16352
```

Chapter 5
Interrupt Control

Introduction

The Extended I/O ROM provides the 9825A Calculator 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.

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 ldk statement.

The On Interrupt Statement

Syntax:

on i 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.

The Enable Interrupt Statement

Syntax:

eir select code [, interrupt enable byte]

Once a service routine and interface have been specified via the oni statement, the eir statement actually enables the interface to interrupt. When the calculator is switched on or (RESET) is pressed, all interfaces are disabled from interrupting the calculator.

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 (page 72) automatically enables an interrupt when it is used.

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 specifi-

cally designed for data transfer with these devices. Examples of data transfer under automatic interrupt control are in Chapter 6.

The Interrupt Return Statement

Syntax:

```
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.

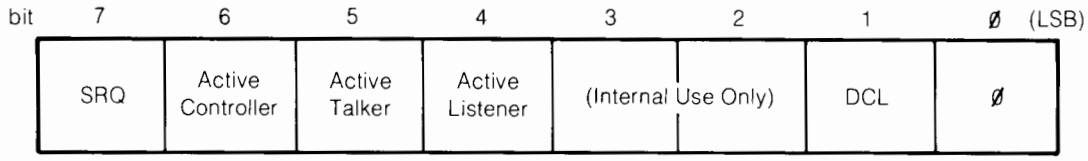
The following sequence shows one method to allow the calculator to rapidly calculate and store data points in an internal array, while a service routine outputs the points to the plotter. This enables the calculator to perform lengthy calculations, or even control other I/O devices, while also driving 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 19
      •
      •
      •
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: ena
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]

[1]These 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 3.)

```
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.

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 726,A[I];ret
56: "LISTEN":red 726,A$;•••;ret
57: "CONTROL": •••;pct 726;ret
58: end
```

Abortive Interrupts

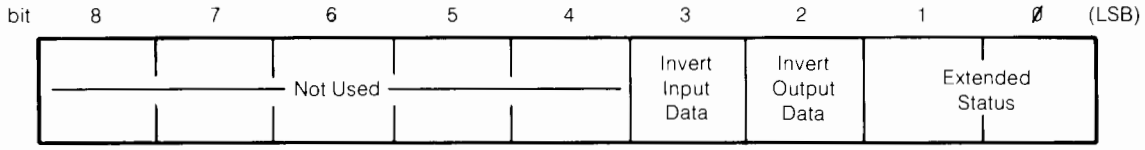
The oni statement described earlier in this chapter allows for an optional abort byte parameter. Normally, interrupts using the end-of-line (EOL) branching scheme described earlier are sufficient, and interrupts are serviced in a reasonable amount of time. Some extraordinary circumstances, however, may 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 [RESET] 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 the General I/O Programming Manual; only the four least-significant bits are described here:

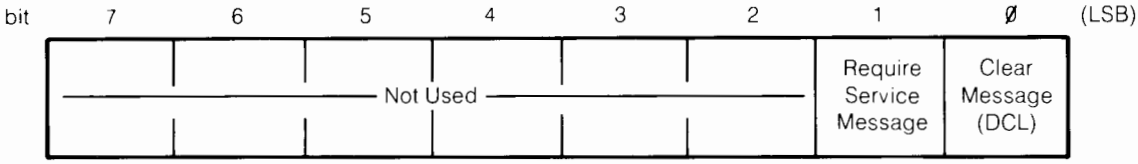


- Bits 0 and 1 – Indicate the state of optional device status-input lines (see the 98032A Installation and Service Manual for details).
- Bits 2 and 3 – Indicate states of logic levels preset on the 98032A Interface. Each of these bits is logical 1 when the corresponding jumper wire on the 98032A is installed to invert data.

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.

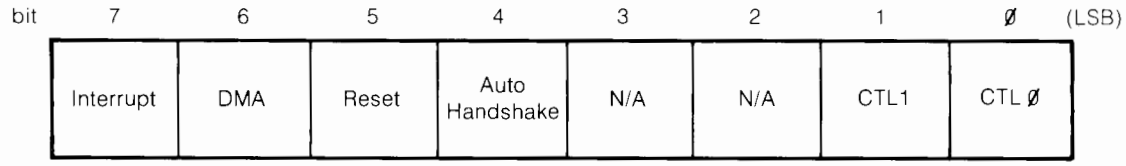
Normal EOL interrupts are serviced only when the program is running. Abortive interrupts and autostart on the Clear message are serviced anytime except during, or immediately after,

program editing (before run is executed). If an abortive interrupt is encountered while a program is being edited, pressing [RESET] 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 by the Extended I/O ROM 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) statement[1] 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.

[1]The 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. Similarily, 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 9825A are "global" variables (except p-numbers, see Advanced Programming Manual). This means that they are recognized and modifiable in all segments of the program. So care should be taken to ensure that an interrupt service rouine (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;wrt 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.

Chapter 6
Buffered I/O

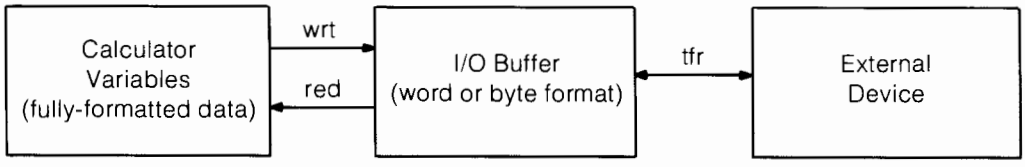
Introduction

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

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

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

[2]For byte transfers, the fast read/write buffer offers the most effecient 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. See page 75 for an explanation of buffer poiners.

The Buffer Statement

Syntax:

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 (with the String Variables ROM).

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 3) 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.

The Transfer Statement

Syntax:

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

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.

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:

Syntax:

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.

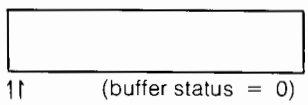
Remember that buffer status is −1 when a transfer is active, and 0 when the buffer is empty.

```
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
```

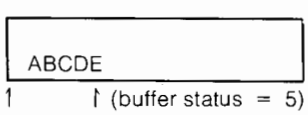
Buffer Pointers

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`



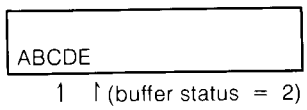
2. Write five characters into buffer:
   `wtb "A","ABCDE"`



3. Transfer three characters out:

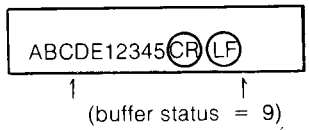
```
tfr "A",6,3
```

Note that ABC still remains in the buffer.



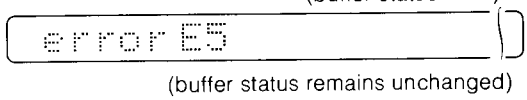
4. Write more characters in:

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



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

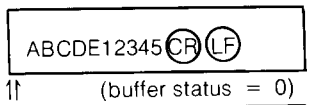
```
wtb "A","XXXXXXXXXXX"
```



(buffer status remains unchanged)

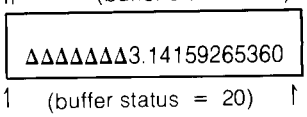
6. Transfer remainder of buffer out:

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



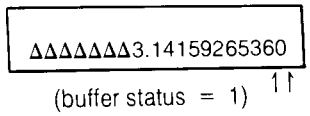
7. Now the buffer can be filled with new data: (Δ = a space):

```
fmt f20.11,z
wrt "A",π
```



8. Transfer 19 bytes out:

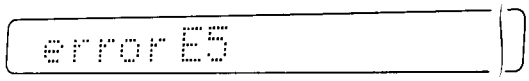
```
tfr "A",2,19
```



9. Attempt to write in one byte:

```
wtb "A","X"
```

Gives error E5 no room to store it.

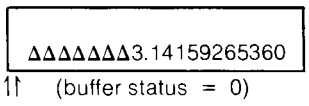


(buffer status unchanged)

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 using the read byte function. If an unknown number of words or bytes is left in the buffer, a sequence such as this could be used to empty the buffer into a "bit bucket":

```
rdb("A")→A jmp rds("A")=0
```

String Variables as Buffers

The size parameter in the buffer statement may be replaced by the name of simple string variable (String ROM). Substrings and strings of a string array are not allowed. This permits the string variable to serve also 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 word-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 reading it into calculator variables. Or data may be "pre-conditioned" to suit a particular format or data structure. Be aware, however, that these String ROM 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 considered as 10 bytes; of course they are not the same 10 bytes transferred from the peripheral since the buffer was modified from the keyboard.

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 cal-

culator 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,"PF4G2S1T"
2: " Type 3":buf "CBuf",G$,3
3: tfr 710,"CBuf",340
4: rds("CBuf")→B;if B=-1;jmp 0
5: fxd 0;prt "#Bytes=",B;spc ;prt "CBuff=",G$;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
*19908
```

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 10^6) and printed.

Printout:

```
#Bytes=          340
CBuff=
 +      10.870E+6+
↓ +      10.880E+6
+↓ +      10.880E+
6+↓ +      10.890E
+6+↓ +      10.890
E+6+↓ +      10.90
0E+6+↓ +      10.9
00E+6+↓ +      10.
910E+6+↓ +      10
.910E+6+↓ +      1
0.920E+6+↓ +
10.920E+6+↓ +
 10.920E+6+↓ +
  10.930E+6+↓ +
   10.930E+6+↓ +
    10.940E+6+↓
+     10.940E+6+↓
 +     10.950E+6+
↓ +      10.950E+6
+↓ +      10.960E+
6+↓ +      10.960E
+6+↓
```

```
Buffer=
 +      10.870e+6
 +      10.880e+6
 +      10.880e+6
 +      10.890e+6
 +      10.890e+6
 +      10.900e+6
 +      10.900e+6
 +      10.910e+6
 +      10.910e+6
 +      10.920e+6
 +      10.920e+6
 +      10.920e+6
 +      10.930e+6
 +      10.930e+6
 +      10.940e+6
 +      10.940e+6
 +      10.950e+6
 +      10.950e+6
 +      10.960e+6
 +      10.960e+6
```

```
Frequency=
 10.87 MHz
 10.88 MHz
 10.88 MHz
 10.89 MHz
 10.89 MHz
 10.90 MHz
 10.90 MHz
 10.91 MHz
 10.91 MHz
 10.92 MHz
 10.92 MHz
 10.92 MHz
 10.93 MHz
 10.93 MHz
 10.94 MHz
 10.94 MHz
 10.95 MHz
 10.95 MHz
 10.96 MHz
 10.96 MHz
```

Demonstration Programs

Buffered I/O allows more efficient use of the 9825A Calculator 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 9825A does no useful work between input samples.

Buffered I/O is a singificant 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
*15126
```

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
*31854
```

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$,1
4: wrt 716,"R";tfr 710,"CBuf",340
5: Y+1→Y;Y*ln(Y)→Z
6: if rds("CBuf")=-1;jmp -1
7: red 716,D
8: fxd 0;prt "Time,ms=",D-C;spc ;prt "Work=",Y;spc
9: for J=1 to 20
10: conv 69,101;red "CBuf",F$[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
*15126
```

Printouts:

Test Program

```
Time:ms=      113

Work=           5

Frequency=
     11.218 MHz
     11.217 MHz
     11.217 MHz
     11.217 MHz
     11.217 MHz
     11.218 MHz
     11.217 MHz
     11.217 MHz
     11.217 MHz
```

Program 1

```
Time:ms=    12622

Work=           0

Frequency=
     11.096 MHz
     11.096 MHz
     11.096 MHz
     11.096 MHz
     11.096 MHz
     11.096 MHz
     11.096 MHz
     11.096 MHz
     11.096 MHz
```

Program 2

```
Time:ms=    12640

Work=        1183

Frequency=
     11.144 MHz
     11.144 MHz
     11.144 MHz
     11.144 MHz
     11.144 MHz
     11.144 MHz
     11.144 MHz
     11.144 MHz
     11.144 MHz
```

Appendices

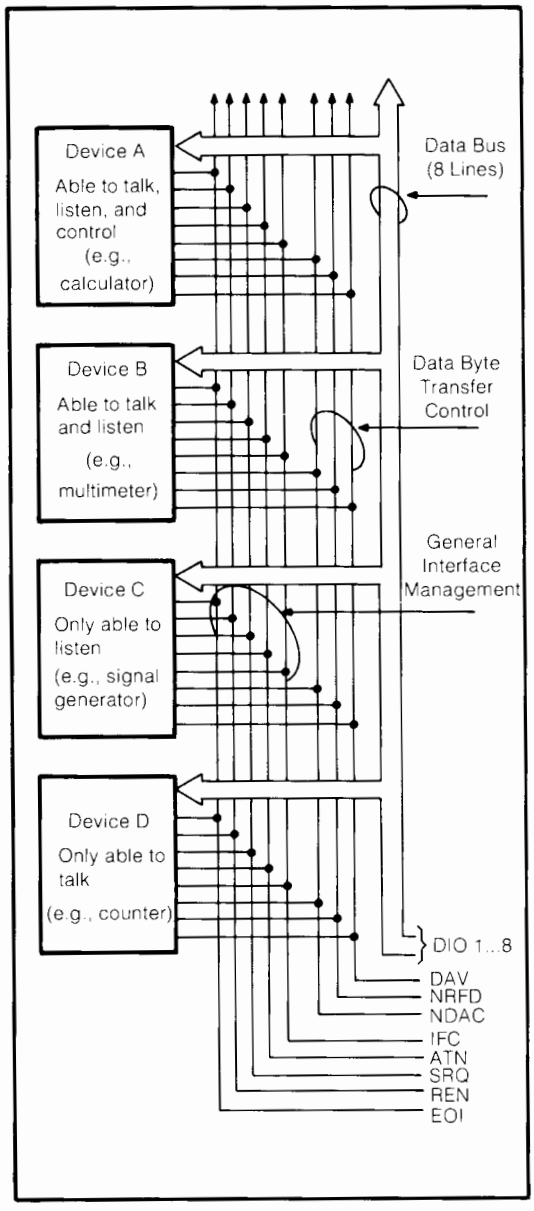
The HP Interface Bus

This appendix 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 in Chapter 4. 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 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 sender function → receiver function(s) (support functions)
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

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

Extended I/O Status Conditions

The following table shows status conditions for various Extended I/O operations and modes. Notice that the Erase, Erase All/Power on, and Run columns from Appendix D of the 9825A Operating and Programming Manual are combined into one column here. R = restored to power-on state; X = unchanged.

Extended I/O Operation or Mode	Calculator Operation			
	Power On Erase Erase All Run	RESET	Continue (after edit)	Continue (after stop)
Conversion and parity tables	R	X	X	X
Octal mode (reset to decimal)	R	R	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	Dec	ASCII Char.	EQUIVALENT FORMS Binary	Octal	Dec	ASCII Char.	EQUIVALENT FORMS Binary	Octal	Dec	ASCII Char.	EQUIVALENT FORMS Binary	Octal	Dec
NULL	00000000	000	0	space	00100000	040	32	@	01000000	100	64	`	01100000	140	96
SOH	00000001	001	1	!	00100001	041	33	A	01000001	101	65	a	01100001	141	97
STX	00000010	002	2	"	00100010	042	34	B	01000010	102	66	b	01100010	142	98
ETX	00000011	003	3	#	00100011	043	35	C	01000011	103	67	c	01100011	143	99
EOT	00000100	004	4	$	00100100	044	36	D	01000100	104	68	d	01100100	144	100
ENQ	00000101	005	5	%	00100101	045	37	E	01000101	105	69	e	01100101	145	101
ACK	00000110	006	6	&	00100110	046	38	F	01000110	106	70	f	01100110	146	102
BELL	00000111	007	7	'	00100111	047	39	G	01000111	107	71	g	01100111	147	103
BS	00001000	010	8	(	00101000	050	40	H	01001000	110	72	h	01101000	150	104
HT	00001001	011	9	)	00101001	051	41	I	01001001	111	73	i	01101001	151	105
LF	00001010	012	10	*	00101010	052	42	J	01001010	112	74	j	01101010	152	106
V_{TAB}	00001011	013	11	+	00101011	053	43	K	01001011	113	75	k	01101011	153	107
FF	00001100	014	12	,	00101100	054	44	L	01001100	114	76	l	01101100	154	108
CR	00001101	015	13	–	00101101	055	45	M	01001101	115	77	m	01101101	155	109
SO	00001110	016	14	.	00101110	056	46	N	01001110	116	78	n	01101110	156	110
SI	00001111	017	15	/	00101111	057	47	O	01001111	117	79	o	01101111	157	111
DLE	00010000	020	16	ø	00110000	060	48	P	01010000	120	80	p	01110000	160	112
DC_1	00010001	021	17	1	00110001	061	49	Q	01010001	121	81	q	01110001	161	113
DC_2	00010010	022	18	2	00110010	062	50	R	01010010	122	82	r	01110010	162	114
DC_3	00010011	023	19	3	00110011	063	51	S	01010011	123	83	s	01110011	163	115
DC_4	00010100	024	20	4	00110100	064	52	T	01010100	124	84	t	01110100	164	116
NAK	00010101	025	21	5	00110101	065	53	U	01010101	125	85	u	01110101	165	117
SYNC	00010110	026	22	6	00110110	066	54	V	01010110	126	86	v	01110110	166	118
ETB	00010111	027	23	7	00110111	067	55	W	01010111	127	87	w	01110111	167	119
CAN	00011000	030	24	8	00111000	070	56	X	01011000	130	88	x	01111000	170	120
EM	00011001	031	25	9	00111001	071	57	Y	01011001	131	89	y	01111001	171	121
SUB	00011010	032	26	:	00111010	072	58	Z	01011010	132	90	z	01111010	172	122
ESC	00011011	033	27	;	00111011	073	59	[	01011011	133	91	{	01111011	173	123
FS	00011100	034	28	<	00111100	074	60	\	01011100	134	92	¦	01111100	174	124
GS	00011101	035	29	=	00111101	075	61	]	01011101	135	93	}	01111101	175	125
RS	00011110	036	30	>	00111110	076	62	^	01011110	136	94	~	01111110	176	126
US	00011111	037	31	?	00111111	077	63	_	01011111	137	95	DEL	01111111	177	127

Buffered I/O Benchmarks

The table below summarizes results of 8 benchmark programs run to measure relative speeds of three methods of transferring data into the 9825A from 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. **9825A/5345A Counter with buffer type 1:**

```
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 1":buf "CBuf",G$,1
4: wrt 716,"R";tfr 710,"CBuf",337
5: if rds("CBuf")=-1;jmp 0
6: red 716,D
7: fxd 0;prt "Time,ms=",D;fxd 2;prt "Avg Time/rdg,ms=",(D-C)/28;spc
8: for J=1 to 28
9: conv 69,101;red "CBuf",F$[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
*11541
```

2. 9825A/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
*10557
```

3. 9825A/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
*21645
```

4. 9825A/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: 1→P;prt "Frequency"
9: for J=1 to 28
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
*5559
```

5. 9825A/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
*21964
```

6. 9825A/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
*20980
```

7. 9825A/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
*30767
```

8. 9825A/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: 1→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
*13109
```

Notes

Extended I/O Syntax Summary

Syntax Conventions

Dot Matrix	–	Characters printed in dot matrix must appear as shown.
[ ]	–	Items within brackets are optional.
Expression	–	A constant like 16.4, a variable like X or B[8] or r3 or A$, or an expression like 8↑4 or 6<A+B.
...	–	Dots indicate that successive parameters, separated by commas, are allowed.

Select Code Format – cc[dd[ee]][.f]

cc = interface select code.
dd = optional HP-IB address code (must be two digits).
ee = optional HP-IB secondary address (must be two digits).
.f = format number, for read (red) and write (wrt) only.

Binary Statements and Functions

moct

The mode octal statement places the calculator in a mode in which all binary-type parameters of the statements and functions of the I/O ROMs are taken to be octal values. See the table on page 6.

mdec

The mode decimal statement returns the calculator to the decimal mode, which is automatically set when the calculator is reset or turned on. See page 6.

dto (expression)

The decimal to octal function converts a decimal number in the 16-bit binary range to its octal equivalent value. See page 7.

otd (expression)

The octal to decimal function converts an octal value in the 16-bit binary range to its decimal equivalent. See page 7.

band (expression$_A$, expression$_B$)

The binary AND function combines two 16-bit values in a binary AND operation and returns the result. See page 7.

ior (expression $_A$, expression $_B$)

The inclusive OR function combines two 16-bit values in an inclusive OR operation and returns the result. See page 8.

eor (expression $_A$, expression $_B$)

The exclusive OR function combines two 16-bit values in an exclusive OR operation and returns the result. See page 7.

cmp (expression)

The complement function returns the 1's complement of a 16-bit binary value. See page 8.

rot (expression , + or − no. of places)

The rotate function performs an n-bit rotation of a 16-bit binary quantity to the left (negative no.) or to the right (positive no.) and returns the result. See page 8.

shf (expression , + or − no. of places)

The shift function performs an n-bit shift on a 16-bit binary quantity to the left (negative no.) or to the right (positive no.) and returns the result. See page 9.

add (expression A , expression B)

The add function performs addition on two 16-bit binary quantities, in octal arithmetic, if the octal mode is set. See page 10.

bit (bit position , expression)
bit ("mask" , expression)

The bit function tests a given 16-bit binary value for a specific bit or for a specified bit pattern (mask) and returns a 1 (true) or 0 (false). See page 10.

HP-IB Statements

dev "name$_1$" , select code$_1$ [, "name$_2$" , select code$_2$...]

The device statement associates names with interface cards and devices, for use in place of the select code parameter. Use of device names also allows addressing multiple listeners on the HP-IB. See page 17.

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

The command statement allows direct addressing of the HP-IB interface, using the bus protocol employed by the HP 9820A, 9821A, and 9830A/B Calculators. See page 35.

equ "name$_1$", data string$_1$ [, "name$_2$", data string$_2$...]

The equate statement allows equating names with HP-IB data sequences for use with the cmd statement. See page 37.

trg select code [device address]

The trigger statement sends a Group Execute Trigger (GET). See page 20.

clr select code [device address]

The clear statement sends a Universal Device Clear (DCL) or an addressed Selective Device Clear (SDC). See page 21.

rem select code [device address]

The remote statement sends the Remote (REN) message. See page 22.

lcl select code

The local statement sends either a Local (GTL) message or a Clear Lockout/Set Local (REN) message. See pages 23 and 24.

llo select code

The local lockout statement sends the Local Lockout (LLO) message. See page 23.

pct select code with device address

The pass control statement passes active control to a specified device on the given HP-IB. See page 30.

cli select code

The clear interface statement sends the Abort (IFC) message. See page 31.

rds (select code with device address)

The read status function with an HP-IB device address (e.g., rds(711)→A) conducts a serial poll and returns the Status Byte message. See pages 26 and 38.

rds (select code [, variable$_1$[, variable$_2$[, variable$_3$]]]) →variable$_4$

Returns up to four status bytes from the 98034A Interface card. See page 38.

pol (select code)

The parallel poll function conducts a parallel poll on the specified HP-IB and returns the current Status Bit message as a single byte. See page 29.

`polc` select code with device address ; status byte

The poll configure statement sends the Parallel Poll Configure (PPC) command and a poll configure byte to a selected device on the HP-IB. See page 30.

`polu` select code [device address]

The poll unconfigure statement sends the universal Parallel Poll Unconfigure (PPU) or the secondary Parallel Poll Disable (PPD) command. See page 30.

`rqs` select code [; status byte]

The Require Service statement allows the calculator (which is not currently the active controller) to request service from the active controller. See page 25.

The Timeout Statement

`time` time in milliseconds

Specifies a maximum time limit for any external peripheral device to respond before issuing `error E4`. See page 42.

The On Error Statement

`onerr` "label"

Allows the program to specify alternative action whenever an error is detected. The following three read-only variables are defined when an error occurs. For more details, see page 42.

`rom`

Indicates whether a mainframe error (0) or an add-on ROM error caused the branch to the error recovery label. An ASCII decimal-equivalent value indicates the letter of the add-on ROM (e.g., 69 = "E" for Extended I/O ROM).

`ern`

Stores the error number.

`erl`

Stores the program line number in which the error occurred.

The Conversion Table Statement

`ctbl` [string variable name]

Establishes a string variable as the full conversion table for automatic conversion between ASCII and another "foreign" code. Ctbl without a parameter cancels the previous conversion table. See page 44.

The Parity Statement

par parity type

Establishes the parity type to be set for output data, or to be checked on input data. See page 47.

Interface Control Operations

wti @, select code

This write interface statement specifies the interface select code for successive wti and rdi operations.

wti register no., expression

Allows direct output to the interface data registers. See page 49.

rdi (register no.)

The read interface function allows direct input from the interface data registers. See page 50.

iof (select code)

The I/O flag function returns a 1 or 0, indicating the state of the specified interface flag line. See page 50.

ios (select code)

The I/O status function returns a 1 or 0, indicating the state of the specified interface status line. See page 50.

Interrupt Control Statements

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

The on interrupt statement specifies a location within a program to which control is to be transferred whenever an interrupt is generated by a specified external device. See page 55.

eir select code [, interrupt enable byte]

The enable interrupt statement enables an external device to generate an interrupt on the occurance of certain specified conditions. See page 56.

iret

The interrupt return statement terminates an interrupt service routine, and returns control to the line of the program that would have been executed if the interrupt had not occurred. See page 57.

Buffered I/O Statements

buf "name" [, buffer size or string variable buffer type]

The buffer statement reserves a segment of read/write memory to be used in an automatic data transfer. The buffer type determines the mode of data transfer to be performed:

Buffer Type	Description
0	Interrupt Buffer, 16-bit words
1	Interrupt Buffer, 8-bit bytes
2	Fast Read/Write, 16-bit words
3	Fast Read/Write, 8-bit bytes
4	DMA buffer, 16-bit words

The buffer is cleared (emptied) by executing the syntax: buf "name". See page 70.

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

The transfer statement automatically transfers data between an I/O buffer and a peripheral device. The buffer type determines the mode of transfer. The optional last-character parameter is used for input transfers only. See page 72.

Extended I/O ROM Error Messages

error E0
- Extended I/O operation executed when a General I/O ROM is not installed.
- HP-IB Error under interrupt: When an HP-IB interrupts with status clear and the ERR bit in the status byte is set, select code 0 is logged in. At the end-of-line service routine, this error is issued.

error E1 Wrong Number of Parameters:
- Bit manipulation functions do not have 2 parameters.
- The on err statement does not have a label.
- The oni statement has less than 2 parameters.
- The polc or rqs statement has less than 2 parameters.
- The tfr statement has less than 2 parameters.
- The cmd statement with bus address has no second parameters.
- The equ or dev statement has an odd number of parameters.
- New buffer allocation with less than 3 parameters.

error E2 Improper Buffer, Device or Equate Table Usage:
- Attempt to add a name in a buffer device or equate table list when that name already exists.
- Buffer, device, or equate name is a null string.
- Attempt to declare multiple listeners with one of the entries not addressing a 98034A Card, or not all on the same HP-IB.
- Read status of multiple listeners.
- Multiple listeners name list ends in a comma.
- Attempt to read to, or write from, a busy buffer.
- Entry in buffer, device, or equate table not found.

error E3 Wrong Parameter Type:
- Parameter of ctbl statement is not a string variable.
- Numeric parameter found when string parameter expected.
- String parameter found when numeric parameter expected.
- Mask parameter in bit function has more than 16 characters.
- Null string found for required string parameter.

error E4 Timeout Error: Specified time ran out without response from peripheral.

error E5 Buffer Overflow or Underflow:

- Attempt to read from an empty buffer or write to a full buffer.
- Attempt to transfer to or from an empty buffer.

error E6 Parameter Overflow:

- Decimal parameter not in range of from −32768 thru 32767 with flag 14 clear.
- Octal parameter not in range of from 0 thru 177777 with flag 14 clear.
- Octal representation contains an 8 or a 9.
- Extended bus address not in range of from 0 thru 31 decimal.
- Buffer type not in the range of from 0 thru 4.
- Negative parameter for buffer size specification.
- Allocating a string as a buffer: After taking 16 characters for working storage, no room left in the string for buffer area.
- Abort byte in eir statement, interrupt enable byte in eir, or character parameter in tfr statement is more than 8 bits; i.e., not in range of from 0 thru 255 decimal or from 0 thru 377 octal.

error E7 Parity Failure: Parity bit of character read does not match specified parity type 1, 2, or 3.

error E8 Improper Interrupt Procedure:

- Attempt to execute an iret statement that is not in a running program, or when no interrupt service routine is active.
- A new program was loaded after an interrupt occured and before the end-of-line service branch, and the service routine was overlayed.
- A new program was loaded from an interrupt service routine and the intercepted line (destination of the iret statement) was overlayed.
- Attempt to transfer a DMA (type 4) buffer with a 98034A HP-IB Interface.
- Attempt to address a select code or a buffer that has not completed the transfer operation. Attempt to read or write with a busy buffer or select code.

error E9 Illegal HP-IB Operation:

- Attempt to address the HP-IB while calculator is not active controller.
- Illegal HP-IB command sequence.
- Attempt to request service on an HP-IB when calculator is active controller.

The Extended I/O ROM adds these meanings to General I/O error messages G4 and G9:

error G4 Improper Select Code:

- Select code parameter of an eir or oni statement is not in range of from 2 thru 15.
- Parameter of an iof or ios statement is not in range of from 0 thru 15.
- Attempt to declare a device name for select code 0 or 1.
- Transfer statement source and destination parameters specify two buffers or two peripherals, rather than one buffer and one peripheral.
- HP-IB control statement used with non-HPIB select code or buffer.
- HP-IB control statement select code specifies bus when only addressed device allowed or addressed device when only bus allowed.

error G9 Improper Hardware Configuration: HP-IB bus functions addressed to non-HP-IB interface card or empty slot.

General I/O ROM Error Messages

error G1 Incorrect format numbers:

- Format number in format statement not in range of $0 \leq n \leq 9$.
- Referenced format number not executed.

error G2 Referenced format statement has an error:

- Incorrect format spec.
- Numeric overflow in format statement.

error G3 Incorrect I/O Parameters:

- Parameter not number or string.
- Negative parameter with f z numeric spec.
- Numeric parameter with c edit spec.
- Binary parameter not in range of $-32768 \leq n \leq 32767$.
- More than one parameter for read binary or read status function.
- Missing parameter or a non-numeric parameter for write control statement.

error 64 Incorrect select code:

- Select code is non-numeric or greater than 4 digits.
- Select code is greater than 16 for read status.
- Select code is not in range from 0 thru 16.
- Select code 1 allowed only for read status.
- HP-IB device address code not in range from 0 thru 31.
- Read from select code 0 not allowed.

error 65 Incorrect read parameter:

- Constant in read list.
- String not filled by read operation.
- Numeric parameter references C format spec.

error 66 Incorrect parameter in conversion statement:

- More than 20 parameters.
- Odd number of parameters.
- Non-numeric parameter.
- Parameter not in range $0 \leq n \leq 127$.

error 67 Unacceptable input data:

- More than one decimal point or "E" read.
- 511 characters read without a LF.
- "E" with no leading digit.
- More than 158 numeric characters read.

error 68 Peripheral device down:

- Incorrect status bits – device not ready or power is off.
- (STOP) cancelled operation.

error 69 Interface hardware problem:

- Improper HP-IB operation.
- Empty I/O slot.
- Select code does not match interface card (e.g., wrt 711 when a 98032A is set to 7, or wrt 6 when 98034A is set to 6).
- Write Control addressed to a 98034A HP-IB Card.

HEWLETT hp PACKARD

SALES & SERVICE OFFICES

AFRICA, ASIA, AUSTRALIA

AMERICAN SAMOA
Calculators Only
Oceanic Systems Inc.
P.O. Box 777
Pago Pago Bayfront Road
Pago Pago 96799
Tel: 633-5513
Cable: OCEANIC-Pago Pago

ANGOLA
Telectra
Empresa Técnica de Equipamentos Eléctricos, S.A.R.L.
R. Barbosa Rodrigues, 42-1°DT.°
Caixa Postal, 6487
Luanda
Tel: 35515/6
Cable: TELECTRA Luanda

AUSTRALIA
Hewlett-Packard Australia Pty. Ltd.
31-41 Joseph Street
Blackburn, Victoria 3130
P.O. Box 36
Doncaster East, Victoria 3109
Tel: 89-6351
Telex: 31-024
Cable: HEWPARD Melbourne

Hewlett-Packard Australia Pty. Ltd.
31 Bridge Street
Pymble
New South Wales, 2073
Tel: 449-6566
Telex: 21561
Cable: HEWPARD Sydney

Hewlett-Packard Australia Pty. Ltd.
153 Greenhill Road
Parkside, 5063, S.A.
Tel: 27-2591
Telex: 82536 ADEL
Cable: HEWPARD ADELAIDE

Hewlett-Packard Australia Pty. Ltd.
141 Stirling Highway
Nedlands, W.A. 6009
Tel: 86-5455
Telex: 93859 PERTH
Cable: HEWPARD PERTH

Hewlett-Packard Australia Pty. Ltd.
121 Wollongong Street
Fyshwick, A.C.T. 2609
Tel: 95-3733
Telex: 62650 Canberra
Cable: HEWPARD CANBERRA

Hewlett Packard Australia Pty. Ltd.
5th Floor
Teachers Union Building
495-499 Boundary Street
Spring Hill, 4000 Queensland
Tel: 29-1544
Telex: 42133 BRISBANE

GUAM
Medical/Pocket Calculators Only
Guam Medical Suply, Inc.
Jay Ease Building, Room 210
P.O. Box 8383
Tamuning 96911
Tel: 646-4513
Cable: EARMED Guam

HONG KONG
Schmidt & Co (Hong Kong) Ltd.
P.O. Box 297
Connalight Centre
39th Floor
Connaught Road, Central
Hong Kong
Tel: H-255291-5
Telex: 74766 SCHMC HX
Cable: SCHMIOTCO Hong Kong

INDIA
Blue Star Ltd.
Kasturi Buildings
Jamshedji Tata Rd.
Bombay 400 020
Tel: 29 50 21
Telex: 2156
Cable: BLUEFROST

Blue Star Ltd.
Sahas
414/2 Vir Savarkar Marg
Prabhadevi
Bombay 400 025
Tel: 45 78 87
Telex: 4093
Cable: FROSTBLUE

Blue Star Ltd.
Band Box House
Prabhadevi
Bombay 400 025
Tel: 45 73 01
Telex: 3751
Cable: BLUESTAR

Blue Star Ltd.
14/40 Civil Lines
Kanpur 208 001
Tel: 6 88 82
Telex: 292
Cable: BLUESTAR

Blue Star Ltd.
7 Hare Street
P.O. Box 506
Calcutta 700 001
Tel: 23-0131
Telex: 7655
Cable: BLUESTAR

Blue Star Ltd.
Blue Star House,
34 Mahatma Gandhi Rd.
Lajpatnagar
New Delhi 110 024
Tel: 62 32 76
Telex: 2463
Cable: BLUESTAR

Blue Star Ltd.
Blue Star House
11/11A Magarath Road
Bangalore 560 025
Tel: 55668
Telex: 430
Cable: BLUESTAR

Blue Star Ltd.
Meenakshi Mandiran
xxx/1678 Mahatma Gandhi Rd.
Cochin 682 016 Kerala
Tel: 32069, 32161, 32282
Telex: 046-514
Cable: BLUESTAR

Blue Star Ltd.
1-1-117/1
Sarojini Devi Road
Secunderabad 500 003
Tel: 70126, 70127
Cable: BLUEFROST
Telex: 459

Blue Star Ltd.
23/24 Second Line Beach
Madras 600 001
Tel: 23954
Telex: 379
Cable: BLUESTAR

Blue Star Ltd.
Nathraj Mansions
2nd Floor Bistupur
Jamshedpur 831 001
Tel: 7383
Cable: BLUESTAR
Telex: 240

INDONESIA
BERCA Indonesia P.T.
P.O. Box 496
1st Floor JL. Cikini Raya 61
Jakarta
Tel: 56038, 40369, 49886
Telex: 42895
Cable: BERCACON

BERCA Indonesia P.T.
63 JL. Raya Gubeng
Surabaya
Tel: 44309

ISRAEL
Electronics & Engineering Div.
of Motorola Israel Ltd.
16, Kremenetski Street
P.O. Box 25016
Tel-Aviv
Tel: 03-389 73
Telex: 33569
Cable: BASTEL Tel-Aviv

JAPAN
Yokogawa-Hewlett-Packard Ltd.
Ohashi Building
1-59-1 Yoyogi
Shibuya-ku, **Tokyo**
Tel: 03-370-2281/92
Telex: 232-2024YHP
Cable: YHPMARKET TOK 23-724

Yokogawa-Hewlett-Packard Ltd.
Nissei Ibaraki Building
2-8 Kasuga 2-chrome, Ibaraki-shi
Osaka 567
Tel: (0726) 23-1641
Telex: 5332-385 YHP OSAKA

Yokogawa-Hewlett-Packard Ltd.
Nakamo Building
24 Kami Sasajima-cho
Nakamura-ku, **Nagoya** , 450
Tel: (052) 571-5171

Yokogawa-Hewlett-Packard Ltd.
Tanigawa Building
2-24-1 Tsuruya-choo
Kanagawa-ku
Yokohama, 221
Tel: 045-312-1252
Telex: 382-3204 YHP YOK

Yokogawa-Hewlett-Packard Ltd.
Mito Mitsui Building
105, 1-chrome, San-no-maru
Mito, Ibaragi 310
Tel.: 0292-25-7470

Yokogawa-Hewlett-Packard Ltd.
Inoue Building
1348-3, Asahi-cho, 1-chome
Atsugi, Kanagawa 243
Tel: 0462-24-0452

KENYA
Technical Engineering Services (E.A.)Ltd.,
P.O. Box 18311
Nairobi
Tel: 557726/556762
Cable: PROTON

Medical Only
International Aeradio(E.A.)Ltd.,
P.O. Box 19012
Nairobi Airport
Nairobi
Tel: 336055/56
Telex: 22201/22301
Cable: INTAERIO Nairobi

KOREA
American Trading Company Korea
C.P.O. Box 1103
Dae Kyung Bldg., 8th Floor
107 Sejong-Ro
Chongro-Ku, **Seoul**
Tel: (4 lines) 73-8924-7
Telex: K-28338
Cable: AMTRACD Seoul

MALAYSIA
Teknik Mutu Sdn. Bhd.
2 Lorong 13/6A
Section 13
Petaling Jaya, **Selangor**
Tel: Kuala Lumpur-54994 or 54916
Telex: MA 37605

Protel Engineering
P.O. Box 1917
Lot 259, Satok Road
Kuching, **Sarawak**
Tel: 20262
Cable: PROTEL ENG

MOZAMBIQUE
A.N. Goncalves, Lta.
162, 1° Apt. 14 Av. D. Luis
Caixa Postal 107
Lourenco Marques
Tel: 27091, 27114
Telex: 6-203 Negon Mo
Cable: NEGON

NEW ZEALAND
Hewlett-Packard (N.Z.) Ltd.
4-12 Cruickshank Street
Kilbirnie, Wellington 3
Mailing Address: Hewlett-Packard (N.Z.) Ltd.
P.O. Box 9443
Courtney Place
Wellington
Tel: 877-199
Telex: NZ 3839
Cable: HEWPACK Wellington

Hewlett-Packard (N.Z.) Ltd.
Pakuranga Professional Centre
267 Pakuranga Highway
Box 51092
Pakuranga
Tel: 569-651
Telex: NZ 3839
Cable: HEWPACK,Auckland

Analytical/Medical Only
Medical Supplies N.Z. Ltd.
Scientific Division
79 Carlton Gore Rd., Newmarket
P.O. Box 1234
Auckland
Tel: 75-289
Telex: 2958 MEDISUP
Cable: DENTAL Auckland

Analytical/Medical Only
Medical Supplies N.Z. Ltd.
P.O. Box 1994
147-161 Tory St.
Wellington
Tel: 850-799
Telex: 3858
Cable: DENTAL, Wellington

Analytical/Medical Only
Medical Supplies N.Z. Ltd.
P.O. Box 309
239 Stanmore Road
Christchurch
Tel: 692-019
Cable: DENTAL, Christchurch

Analytical/Medical Only
Medical Supplies N.Z. Ltd.
303 Great King Street
P.O. Box 233
Dunedin
Tel: 88-817
Cable: DENTAL, Dunedin

NIGERIA
The Electronics Instrumentations Ltd.
N6B/770 Oyo Road
Oluseun House
P.M.B. 5402
Ibadan
Tel: 61577
Telex: 31231 TEIL Nigeria
Cable: THETEIL Ibadan

The Electronics Instrumentations Ltd.
144 Agege Motor Road, Mushin
P.O. Box 6645
Lagos
Cable: THETEIL Lagos

PAKISTAN
Mushko & Company, Ltd.
Oosman Chambers
Abdullah Haroon Road
Karachi-3
Tel: 511027, 512927
Telex: KR894
Cable: COOPERATOR Karachi

Mushko & Company, Ltd.
38B, Satellite Town
Rawalpindi
Tel: 41924
Cable: FEMUS Rawalpindi

PHILIPPINES
The Online Advanced Systems Corporation
Filcapital Bldg.
11th Floor, Ayala Ave.
Makati, **Rizal**
Tel: 86-40-81, ext. 223,263
Telex: 3274 ONLINE

RHODESIA
Field Technical Sales
45 Kelvin Road North
P.O. Box 3458
Salisbury
Tel: 705231 (5 lines)
Telex: RH 4122

SINGAPORE
Hewlett-Packard Singapore (Pte.) Ltd.
Blk. 2, 6th Floor, Jalan Bukit Merah
Redhill Industrial Estate
Alexandra P.O. Box 58,
Singapore 3
Tel: 633022
Telex: HPSG RS 21486
Cable: HEWPACK, Singapore

SOUTH AFRICA
Hewlett-Packard South Africa (Pty.), Ltd.
Private Bag Wendywood
Sandton, Transvaal 2144
Hewlett-Packard House
Daphne Street, Wendywood,
Sandton, Transvaal 2144
Tel: 802-104016
Telex: SA43-4782JH
Cable: HEWPACK JOHANNESBURG

Hewlett-Packard South Africa (Pty.), Ltd.
P.O. Box 120
Howard Place, Cape Province, 7450
Pine Park Center, Forest Drive,
Pinelands, Cape Province, 7405
Tel: 53-7955 thu 9
Telex: 57-0006

Hewlett-Packard South Africa (Pty.), Ltd.
P.O. Box 37099
Overport, Durban 4067
641 Ridge Road, Durban
Durban, 4001
Tel: 88-7478,88-1080,88-2520
Telex: 6-7954
Cable: HEWPACK

TAIWAN
Hewlett-Packard Far East Ltd.,
Taiwan Branch
39 Chung Shiao West Road
Sec. 1, 7th Floor
Taipei
Tel: 389160,1,2,3
Telex: 21824 HEWPACK
Cable: HEWPACK TAIPEI

Hewlett-Packard Far East Ltd.
Taiwan Branch
68-2, Chung Cheng 3rd. Road
Kaohsiung
Tel: (07) 242318-Kaohsiung

Analytical Only
San Kwang Instruments Co., Ltd.,
No. 20, yung Sui Road
Taipei, 100
Tel: 3715171-4 (4 lines)
Telex: 22894 SANKWANG
Cable: SANKWANG TAIPEI

TANZANIA
Medical Only
International Aeradio (E.A.), Ltd.
P.O. Box 861
Daressalaam
Tel: 21251 Ext. 265
Telex: 41030

THAILAND
UNIMESA Co., Ltd.
Elcom Research Building
Bangjak Sukumvit Ave.
Bangkok
Tel: 932387, 930338
Cable: UNIMESA Bangkok

UGANDA
Medical Only
International Aeradio(E.A.), Ltd.,
P.O. Box 2577
Kampala
Tel: 54388
Cable: INTAERIO Kampala

ZAMBIA
R.J. Tilbury (Zambia) Ltd.
P.O. Box 2792
Lusaka
Tel: 73793
Cable: ARJAYTEE, Lusaka

OTHER AREAS NOT LISTED, CONTACT:
Hewlett-Packard Intercontinental
3200 Hillview Ave.
Palo Alto, California 94304
Tel: (415) 493-1501
TWX: 910-373-1267
Cable: HEWPACK Palo Alto
Telex: 034-8300, 034-8493

CANADA

ALBERTA
Hewlett-Packard (Canada) Ltd.
11748 Kingsway Ave.
Edmonton T5G 0X5
Tel: (403) 452-3670
TWX: 610-831-2431 EDTH

Hewlett-Packard (Canada) Ltd.
915-42 Avenue S.E. Suite 102
Calgary T2G 1Z1
Tel: (403) 287-1672
TWX: 610-821-6141

BRITISH COLUMBIA
Hewlett-Packard (Canada) Ltd.
837 E. Cordova Street
Vancouver V6A 3R2
Tel: (604) 254-0531
TWX: 610-922-5059 VCR

MANITOBA
Hewlett-Packard (Canada) Ltd.
513 Century St.
St. James
Winnipeg R3H 0L8
Tel: (204) 786-7581
TWX: 610-671-3531

NOVA SCOTIA
Hewlett-Packard (Canada) Ltd.
800 Windmill Road
P.O. Box 9331
Dartmouth B2Y 3Z6
Tel: (902) 469-7820
TWX: 610-271-4482 HFX

ONTARIO
Hewlett-Packard (Canada) Ltd.
1785 Woodward Dr.
Ottawa K2C 0P9
Tel: (613) 225-6530
TWX: 610-562-8968

Hewlett-Packard (Canada) Ltd.
6877 Goreway Drive
Mississauga L4V 1M8
Tel: (416) 678-9430
TWX: 610-492-4246

QUEBEC
Hewlett-Packard (Canada) Ltd.
275 Hymus Blvd.
Pointe Claire H9R 1G7
Tel: (514) 697-4232
TWX: 610-422-3022
TLX: 05-821521 HPCL

Hewlett-Packard (Canada) Ltd.
2376 Galvani Street
Ste-Foy G1N 4G4
Tel: (418) 688-8710
TWX: 610-571-5525

FOR CANADIAN AREAS NOT LISTED:
Contact Hewlett-Packard (Canada) Ltd. in Mississauga.

CENTRAL AND SOUTH AMERICA

ARGENTINA
Hewlett-Packard Argentina S.A.
Av. Leandro N. Alem 822 - 12°
1001**Buenos Aires**
Tel: 32-4461/62/63/64
Telex: Public Booth

BOLIVIA
Stambuk & Mark (Bolivia) Ltda.
Av. Mariscal, Santa Cruz 1342
La Paz
Tel: 40626, 53163, 52421
Telex: 3560014
Cable: BUKMAR

BRAZIL
Hewlett-Packard do Brasil I.E.C. Ltda.
Rua Frei Caneca, 1140/52 Bela Vista
01307-**São Paulo**-SP
Tel: 288-71-11, 287-81-20, 287-61-93
Telex: 391-1102-3602 HPBR-BR
Cable: HEWPACK São Paulo

Hewlett-Packard do Brasil I.E.C. Ltda.
Rua Padre Chagas, 32
90000-**Porto Alegre**-RS
Tel: (0512) 22-2998, 22-5621
Cable: HEWPACk pohto Alegre

Hewlett-Packard do Brasil I.E.C. Ltda.
Rua Siqueira Campos, 53, 4° andar-Copacabana
20000-**Rio de Janeiro**-GB
Tel: 257-80-94-DDD (021)
Telex: 391-212-1905 HEWP-BR
Cable: HEWPACK Rio de Janeiro

CHILE
Calcagni y Metcalfe Ltda.
Alameda 580-01. 807
Casilla 2118
Santiago, 1
Tel: 390813
Telex: 3520001 CALMET
Cable: CALMET Santiago

Medical Only
General Machinery Co., Ltda.
Paraguay 494
Casilla 13910
Santiago
Tel: 31124
Cable: GEMCO Santiago

COLOMBIA
Instrumentación
Henrik A. Langebaek & Kier S.A.
Carrera 7 No. 48-75
Apartado Aéreo 6287
Bogotá, I.D.E.
Tel: 69-88-77
Cable: AARIS Bogotá
Telex: 044-400

COSTA RICA
Científica Costarricense S.A.
Calle Central, Avenidas 1 y 3
Apartado 10159
San José
Tel: 21-86-13
Cable: GALGUR San José

ECUADOR
Medical Only
A.F. Viscaino Compañía Ltda.
Av. Rio Amazonas No. 239
P.O. Box 2925
Quito
Tel: 242-150,247-033/034
Cable: Astor Quito

Calculators Only
Computadoras y Equipos Electrónicos
P.O. Box 2695
990 Toledo (y Cordero)
Quito
Tel: 525-982
Telex: 02-2113 Sagita Ed
Cable: Sagita-Quito

EL SALVADOR
Instrumentacion y Procesamiento Electronico de el Salvador
Bulevar de los Heroes II-48
San Salvador
Tel: 252787

GUATEMALA
IPESA
Avenida La Reforma 3-48,
Zona 9
Guatemala City
Tel: 63627, 64786
Telex: 4192 Teletro Gu

MEXICO
Hewlett-Packard Mexicana, S.A. de C.V.
Torres Adalid No. 21, 11° Piso
Col. del Valle
Mexico 12, D.F.
Tel: (905) 543-42-32
Telex: 017-74-507

Hewlett-Packard Mexicana, S.A. de C.V.
Ave. Constitución No. 2184
Monterrey, N.L.
Tel: 48-71-32, 48-71-84
Telex: 038-843

NICARAGUA
Roberto Terán G.
Apartado Postal 689
Edificio Terán
Managua
Tel: 25114, 23412,23454
Cable: ROTERAN Managua

PANAMA
Electrónico Balboa, S.A.
P.O. Box 4929
Calle Samuel Lewis
Ciudad de Panama
Tel: 64-2700
Telex: 3431103 Curunda, Canal Zone
Cable: ELECTRON Panama

PARAGUAY
Z.J. Melamed S.R.L.
División: Aparatos y Equipos Médicos
División: Aparatos y Equipos Científicos y de Investigación
P.O. Box 676
Chile-482, Edificio Victoria
Asunción
Tel: 4-5069, 4-6272
Cable: RAMEL

PERU
Compañía Electro Médica S.A.
Los Flamencos 145
San Isidro Casilla 1030
Lima 1
Tel: 41-3703
Cable: ELMED Lima

PUERTO RICO
Hewlett-Packard Inter-Americas
Puerto Rico Branch Office
P.O. Box 2908I
65th Inf. Station
San Juan 00929
Calle 272, Urb. Country Club
Carolina 00639
Tel: (809) 762-7355/7455/7655
Telex: HPIC-PR 3450514

URUGUAY
Pablo Ferrando S.A.
Comercial e Industrial
Avenida Italia 2877
Casilla de Correo 370
Montevideo
Tel: 40-3102
Cable: RADIUM Montevideo

VENEZUELA
Hewlett-Packard de Venezuela C.A.
Apartado 50933, Caracas 105
Edificio Segre
Tercera Transversal
Los Ruices Norte
Caracas 107
Tel: 35-01-07, 35-00-84, 35-00-65, 35-00-31
Telex: 25146 HEWPACK
Cable: HEWPACK Caracas

FOR AREAS NOT LISTED, CONTACT:
Hewlett-Packard
Inter-Americas
3200 Hillview Ave.
Palo Alto, California 94304
Tel: (415) 493-1501
TWX: 910-373-1260
Cable: HEWPACK Palo Alto
Telex: 034-8300, 034-6493

EUROPE, NORTH AFRICA AND MIDDLE EAST

AUSTRIA
Hewlett-Packard Ges.m.b.H.
Handelskai 52
P.O. box 7
A-1205 **Vienna**
Tel: (0222) 35 16 21 to 27
cable: HEWPAK Vienna
Telex: 75923 hewpak a

BELGIUM
Hewlett-Packard Benelux
S.A./N.V.
Avenue de Col-Vert, 1,
(Groenkraaglaan)
B-1170 **Brussels**
Tel: (02) 672 22 40
Cable: PALOBEN Brussels
Telex: 23 494 paloben bru

CYPRUS
Kypronics
19, Gregorios & Xenopoulos Rd.
P.O. Box 1152
CY-**Nicosia**
Tel: 45628/29
Cable: KYPRONICS PANOEHIS
Telex: 3018

CZECHOSLOVAKIA
Vyvojova a Provozni Zakladna
Vyzkumnych Ustava v Bechovicich
CSSR-25097
Bechovice u Prahy
Tel: 89 93 41
Telex: 121333

DDR
Entwicklungslabor der TU Oresden
Forschungsinstitut Meinsberg
ODR-7305
Waldheim/Meinaberg
Tel: 37 667
Telex: 518741

DENMARK
Hewlett-packard A/S
Datavej 52
DK-3460 **Birkerod**
Tel: (02) 81 66 40
Cable: HEWPACK AS
Telex: 166 40 hpas

Hewlett-Packard A/S
Navervej 1
DK-8600 **Silkeborg**
Tel: (06) 82 71 66
Telex: 166 40 hpas
Cable: HEWPACK AS

FINLAND
Hewlett-Packard OY
Nahkahousuntie 5
P.O. Box 6
SF-00211 **Helsinki** 21
Tel: 5923031
Cable: HEWPACKDY Helsinki
Telex: 12-1563

FRANCE
Hewlett-Packard France
Ouartier de Courtaboeuf
Boite Postale No. 6
F-91401 **Orsay** Cédex
Tel: (1) 907 78 25
Cable: HEWPACK Orsay
Telex: 600048

Hewlett-Packard France
"Le Saquin"
Chemin des Mouilles
Boite Postale No. 12
F-69130 **Ecully**
Tel: (78) 33 81 25,
Cable: HEWPACK Eculy
Telex: 310617

Hewlett-Packard France
Agence Régionale
Péricentre de la Cépière
Chemin de la Cépière, 20
F-31300 **Toulouse-Le Mirail**
Tel.(61) 40 11 12
Cable: HEWPACK 51957
Telex: 510957

Hewlett-Packard France
Agence Régionale
Aéroport principal de
Marseille-Marignane
F-13721**Marignane**
Tel: (91) 89 12 36
Cable: HEWPACK MARGN
Telex: 410770

Hewlett-Packard France
Agence Régionale
63, Avenue de Rochester
Boîte Postale
F-35014 **Rennes** Cédex
Tel: (99) 36 33 21
Cable: HEWPACK 74912
Telex: 740912

Hewlett-Packard France
Agence Régionale
74, Allée de la Robertsau
F-67000 **Strasbourg**
Tel: (88) 35 23 20/21
Telex: 890141
Cable: HEWPACK STRBG

Hewlett-Packard France
Agence Régionale
Centre Vauban
201, rue Colbert
Entrée A2
F-59000 **Lille**
Tel: (20) 51 44 14
Telex: 820744

GERMAN FEDERAL REPUBLIC
Hewlett-Packard GmbH
Vertriebszentrale Frankfurt
Bernerstrasse 117
Postfach 560 140
D-6000 **Frankfurt** 56
Tel: (0611) 50 04-1
Cable: HEWPACKSA Frankfurt
Telex: 04 13249 hpfmd

Hewlett-Packard GmbH
Technisches Buero Böblingen
Herrenbergerstrasse 110
D-7030 **Böblingen**, Württemberg
Tel: (07031) 667-1
Cable: HEPAK Böblingen
Telex: 07265739 bbn

Hewlett-Packard GmbH
Technisches Buero Düsseldorf
Emanuel-Leutze-Str 1 (Seestern)
D-4000 **Düsseldorf**
Tel: (0211) 59 71-1
Telex: 085/86 533 hpdd d

Hewlett-Packard GmbH
Technisches Buero Hamburg
Wendenstrasse 23
D-2000 **Hamburg** 1
Tel: (040) 24 13 93
Cable: HEWPACKSA Hamburg
Telex: 21 63 032 hphh d

Hewlett-Packard GmbH
Technisches Buero Hannover
Mellendorfer Strasse 3
D-3000 **Hannover-Kleefeld**
Tel: (0511) 55 60 46
Telex: 092 3259

Hewlett-Packard GmbH
Technisches Buero Nuremberg
Neumeyer Str. 90
D-8500 **Nuremberg**
Tel. (0911) 56 30 83/85
Telex: 0623 860

Hewlett-Packard GmbH
Technisches Buero München
Unterhachinger Strasse 28
ISAR Center
D-8012 **Ottobrunn**
Tel. (089) 601 30 61/7
Cable: HEWPACKSA München
Telex: 0524985

Hewlett-Packard GmbH
Technisches Buero Berlin
Keith Strasse 2-4
D-1000 **Berlin** 30
Tel. (030) 24 90 86
Telex: 18 3405 hpbln d

GREECE
Kostas Karayannis
18, Ermou Street
GR-**Athens** 126
Tel: 3237731
Cable: RAKAR Athens
Telex: 21 59 62 rkar gr

Analytical Only
"INTECO"
G. Papathanassiou & Co.
Marni 17
GR - **Athens** 103
Tel: 522 1915
Cable: INTEKNIKA Athens
Telex: 21 5329 INTE GR

Medical Only
Technomed Hellas Ltd.
52, Skoufa Street
GR - **Athens** 135
Tel. 626 972,663 930,614 959
Cable: ETALAK Athens
Telex: 21-4693 ETAL GR

HUNGARY
MTA
Müszerügyi és Méréstechnikai
Szolgalata
Lenin Krt. 67
1391 **Budapest** VI
Tel: 42 03 36
Telex: 22 51 14

ICELAND
Medical Only
Elding Trading Company Inc.
Hafnarhvoli - Tryggvatotu
IS-**Reykjavik**
Tel.: 1 58 20
Cable: ELDING Reykjavik

IRAN
Hewlett-Packard Iran Ltd.
Mir-Emad Avenue
14th Street No. 13
P.O. Box 41/2419
IR-**Tehran**
Tel: 85 10 82/86
Telex: 21 25 74 Khrm ir

IRELAND
Hewlett-Packard Ltd.
King Street Lane
GB-**Winnersh**,Wokingham
Berks, RG11 5AR
Tel: (0734) 78 47 74
Telex: 847178/848179

ITALY
Hewlett-Packard Italiana S.p.A.
Casella postale 3645
I-20100 **Milano**
Tel: (2) 6251 (10 lines)
Cable: HEWPACKIT Milano
Telex: 32046

Hewlett-Packard Italiana S.p.A.
Via Pietro Maroncelli 40
(ang. Via Visentin)
I-35100 **Padova**
Tel: (49) 66 48 88
Telex: 41612 Hewpacki

Medical only
Hewlett-Packard Italiana S.p.A.
Via d'Aghiardi, 7
I-56100 **Pisa**
Tel: (050) 2 32 04
Telex: 32046 via Milano

Hewlett-Packard Italiana S.p.A.
Via G. Armellini 10
I-00143 **Roma**
Tel: (06) 54 69 61
Telex: 61514
Cable: HEWPACKIT Roma

Hewlett-Packard Italiana S.p.A.
Via San Quirtino, 46
I-10121 **Torino**
Tel: (011) 52 82 64/54 84 68
Telex: 32046 via Milano

Medical/Calculators Only
Hewlett-Packard Italiana S.p.A.
Via Principe Nicola 43 G/C
I-95126 **Catania**
Tel:(095) 37 05 04

Hewlett-Packard Italiana S.p.A.
Via Amerigo Vespucci, 9
I-80142 **Napoli**
Tel: (081) 33 77 11

Hewlett-Packard Italiana S.p.A.
Via E. Masi, 9/B
I-40137 **Bologna**
Tel: (051) 30 78 87

KUWAIT
Al-Khaldiya Trading &
Contracting Co.
P.O. Box 830
Kuwait
Tel: 42 49 10
Cable: VISCOUNT

LUXEMBURG
Hewlett-Packard Benelux
S.A./N.V.
Avenue du Col-Vert, 1,
(Groenkraaglaan)
B-1170 **Brussels**
Tel: (02) 672 22 40
Cable: PALOBEN Brussels
Telex: 23 494

MOROCCO
Gerep
190, Blvd. Brahim Roudani
Casablanca
Tel: 25-16-76/25-90-99
Cable: Gerep-Casa
Telex: 23739

NETHERLANDS
Hewlett-Packard Benelux N.V.
Van Heuven Goedhartlaan 121
P.O. Box 667
NL- **Amstelveen** 1134
Tel: (020) 47 20 21
Cable: PALOBEN Amsterdam
Telex: 13 216 hepa nl

NORWAY
Hewlett-Packard Norge A/S
Nesveien 13
Box 149
N-1344 **Hasium**
Tel. (02) 53 83 60
Telex: 16621 hpnas n

POLAND
Biuro Informacji Technicznej
Hewlett-Packard
Ul Stawki 2 6P
00-950**Warsaw**
Tel: 39 67 43
Telex: 81 24 53 hepa pl

UNIPAN
Zaklad Doswiadczalny
Budowy Aparatury Naukowej
Ul. Krajowej Rady
Narodowej 51/55
00-800 **Warsaw**
Tel: 20 62 21
Telex: 81 46 48

Zaklady Naprawcze Sprzetu
Medycznego
Plac Komuny Paryskiej 6
90-007 **Lodz**
Tel: 334-41, 337-83

PORTUGAL
Telectra-Empresa Técnica de
Equipamentos Eléctricos S.a.r.l.
Rua Rodrigo da Fonseca 103
P.O. Box 2531
P-**Lisbon** 1
Tel: (19) 68 60 72
Cable: TELECTRA Lisbon
Telex: 12598

Medical only
Mundinter
Intercambio Mundial de Comércio
S.a.r.l.
Av.A.A.de Aguiar 138
P.O. Box 2761,
P - **Lisbon**
Tel: (19) 53 21 31/7
Cable: INTERCAMBIO Lisbon

RUMANIA
Hewlett-Packard Reprezentanta
BD.N. Balcescu 16
Bucharest
Tel: 158023/138885
Telex: 10440

I.I.R.U.C.
Intreprinderea Pentru
Intretinerea
Si Repararea Utilajelor de Calcul
B-dul prof. Dimitrie Pompei 6
Bucharest-Sectorul 2
Tel: 12 64 30
Telex: 01183716

SAUDI ARABIA
Modern Electronic Establishment
King Abdul Aziz str.(Head office)
P.O. Box 1228
Jeddah
Tel: 31173-332201
Cable: ELECTRA

P.O. Box 2728 (Service center)
Riyadh
Tel: 62596-66232
Cable: RAOUFCO

SPAIN
Hewlett-Packard Española, S.A.
Jerez No. 3
E-**Madrid** 16
Tel:(1) 458 26 00 (10 lines)
Telex: 23515 hpe

Hewlett-Packard Española, S.A.
Milanesado 21-23
E-**Barcelona** 17
Tel. (3) 203 6200 (5 lines)
Telex: 52603 hpbe e

Hewlett-Packard Española, S.A.
Av Ramón y Cajal. 1-9°
(Edificio Sevilla I)
E-**Seville 5**
Tel: 64 44 54/58

Hewlett-Packard Española S.A.
Edificio Albia II 7° B
E-**Bilbao**-1
Tel: 23 83 06/23 82 06

Calculators Only
Hewlett-Packard Española S.A.
Gran Via Fernando El Catdlico, 67
E-**Valencia**-8
Tel: 326 67 28/326 85 55

SWEDEN
Hewlett-Packard Sverige AB
Enghetsvägen 3
Fack
S-161 20 **Bromma** 20
Tel: (08) 730 05 50
Cable: MEASUREMENTS
Stockholm
Telex: 10721

Hewlett-Packard Sverige AB
Frötallsgatan 30
S-421 32 **Västra Frölunda**
Tel: (031) 49 09 50
Telex: 10721 via Bromma Office

SWITZERLAND
Hewlett-Packard (Schweiz) AG
Zurcherstrasse 20
P.O. Box 307
CH-8952 **Schlieren-Zurich**
Tel: (01) 730 52 40
Cable: HPAG CH
Telex: 53933 hpag ch

Hewlett-Packard (Schweiz) AG
9, Chemin Louis-Pictet
CH-1214 **Vernier-Geneva**
Tel: (022) 41 49 50
Cable: HEWPACKAG Geneva
Telex: 27 333 hpag ch

SYRIA
Medical/calculator only
Sawah & Co.
Place Azmé
B.P. 2308
SYR-**Dameascus**
Tel: 16367, 19697, 14268
Cable: SAWAH, Damascus

TURKEY
Telekom Engineering Bureau
P.O. Box 437
Beyoglu
TR-**Istanbul**
Tel: 49 40 40
Cable: TELEMATION Istanbul
Telex: 23609

Medical only
E.M.A.
Muhendisiik Kollektif Sirketi
Adakale Sokak 41/6
TR-**Ankara**
Tel: 175622

Analytical only
Yilmaz Ozyurek
Milli Mudafaa Cad No. 16/6
Kizilay
TR-**Ankara**
Tel: 25 03 09
Telex: 42576 Dzek tr

UNITED KINGDOM
Hewlett-Packard Ltd.
King Street Lane
GB-**Winnersh**, Wokingham
Berks. RG11 5AR
Tel: (0734) 78 47 74
Cable. Hewpie London
Telex:847178/9

Hewlett-Packard Ltd.
"The Graftons"
Stamford New Road
GB-**Altrincham**
Cheshire WA14 1DQ
Tel: (061) 9289021
Cable: Hewpie Manchester
Telex: 668068

Hewlett-Packard Ltd
Lygon Court
Dudley Road
GB-**Halesowen**, Worcs
Tel: (021) 550 7053
Telex: (021) 550 7273

Hewlett-Packard Ltd.
Wedge House
799, London Road
GB-**Thornton Heath**
Surrey CR4 6XL
Tel: (01) 6840105
Telex: 946825

Hewlett-Packard Ltd
c/o Makro
South Service Wholesale Centre
Wear Industrial Estate
Washington
GB-**New Town**, County Durham
Tel: Washington 464001 ext. 57/58

USSR
Hewlett-Packard Representative
Office USSR
Pokrovsky Boulevard 4/17, KV 12
Moscow 101000
Tel: 294-2024
Telex: 7825 hewpak su

YUGOSLAVIA
Iskra-standard/Hewlett-Packard
Miklosiceva 38/VII
61000 **Ljubljana**
Tel: 31 59 79/32 16 74
Telex: 31300

MEDITERRANEAN AND MIDDLE EAST COUNTRIES NOT SHOWN PLEASE CONTACT:
Hewlett-Packard S.A.
Mediterranean and Middle
East Operations
35, Kolokotroni Street
Platia Kefallariou
GR-Kifissia-**Athens**, Greece
Tel: 8080337/359/429

SOCIALIST COUNTRIES NOT SHOWN PLEASE CONTACT:
Hewlett-Packard Ges.m.b.H
P.O. Box 7
A-1205 **Vienna**, Austria
Tel: (0222) 35 16 21 to 27

FOR OTHER AREAS NOT LISTED CONTACT
Hewlett-Packard S.A.
7, rue du Bois-du-Lan
P.O. Box
CH-1217 Meyrin 2 - **Geneva**
Switzerland
Tel: (022) 41 54 00

UNITED STATES

ALABAMA
8290 Whitesburg Dr., S.E.
P.O. Box 4207
Huntsville 35802
Tel: (205) 881-4591

Medical Only
228 W. Valley Ave.,
Room 220
Birmingham 35209
Tel: (205) 942-2081

ARIZONA
2336 E. Magnolia St.
Phoenix 85034
Tel: (602) 244-1361

2424 East Aragon Rd.
Tucson 85706
Tel: (602) 294-3148

***ARKANSAS**
Medical Service Only
P.O. Box 5646
Brady Station
Little Rock 72205
Tel:)501) 664-8773

CALIFORNIA
1430 East Orangethorpe Ave.
Fullerton 92631
Tel: (714) 870-1000

3939 Lankershim Boulevard
North Hollywood 91604
Tel: (213) 877-1282
TWX: 910-499-2170

6305 Arizona Place
Los Angeles 90045
Tel: (213) 649-2511
TWX: 910-328-6147

***Los Angeles**
Tel: (213) 776-7500

3003 Scott Boulevard
Santa Clara 95050
Tel: (408) 249-7000
TWX: 910-338-0518

***Ridgecrest**
Tel: (714) 446-6165

2220 Watt Ave.
Sacramento 95825
Tel: (916) 482-1463

9606 Aero Drive
P.O. Box 23333
San Diego 92123
Tel: (714) 279-3200

COLORADO
5600 South Ulster Parkway
Englewood 80110
Tel: (303) 771-3455

CONNECTICUT
12 Lunar Drive
New Haven 06525
Tel: (203) 389-6551
TWX: 710-465-2029

FLORIDA
P.O. Box 24210
2806 W. Oakland Park Blvd.
Ft. Lauderdale 33307
Tel: (305) 731-2020

***Jacksonville**
Medical Service only
Tel: (904) 725-6333

P.O. Box 13910
6177 Lake Ellenor Dr.
Orlando 32809
Tel. (305) 859-2900

P.O. Box 12826
Pensacola 32575
Tel: (904) 434-3081

GEORGIA
P.O. Box 105005
Atlanta 30348
Tel. (404) 955-1500
TWX:810-766-4890

Medical Service Only
***Augusta** 30903
Tel: (404) 736-0592

HAWAII
2875 So. King Street
Honolulu 96814
Tel: (808) 955-4455

ILLINOIS
5500 Howard Street
Skokie 60076
Tel: (312) 677-0400
TWX: 910-223-3613

Effective Nov.1, 1976
35201 Tollview Dr.
Rolling Meadows 60008
Tel: (312) 255-9800
TWX: 910-687-2260

INDIANA
7301 North Shadeland Ave.
Indianapolis46250
Tel: (317)842-1000
TWX: 810-260-1796

IOWA
1902 Broadway
Iowa City 52240
Tel: (319) 338-9466
Night: (319) 338-9467

KENTUCKY
Medical Only
Atkinson Square
3901 Atkinson Dr.,
Suite 207
Louisville 40218
Tel: (502) 456-1573

LOUISIANA
P.O. Box 840
3239 Williams Boulevard
Kenner 70062
Tel. (504) 721-6201

MARYLAND
6707 Whitestone Road
Baltimore 21207
Tel: (301) 944-5400
TWX: 710-862-9157

2 Choke Cherry Road
Rockville 20850
Tel: (301) 948-6370
TWX: 710-828-9684

MASSACHUSETTS
32 Hartwell Ave.
Lexington 02173
Tel: (617) 861-8960
TWX: 710-326-6904

MICHIGAN
23855 Research Drive
Farmington Hills 48024
Tel: (313) 476-6400
TWX: 810-242-2900

MINNESOTA
2400 N. Prior Ave.
Roseville 55113
Tel. (612) 636-0700
TWX: 910-563-3734

MISSISSIPPI
***Jackson**
Medical Service only
Tel: (601) 982-9363

MISSOURI
11131 Colorado Ave.
Kansas City 64137
Tel: (816) 763-8000
TWX: 910-771-2087

148 Weldon Parkway
Maryland Heights 63043
Tel: (314) 567-1455
TWX: 910-764-0830

NEBRASKA
Medical Only
7171 Mercy Road
Suite 110
Omaha 68106
Tel. (402) 392-0948

NEW JERSEY
W. 120 Century Rd.
Paramus 07652
Tel: (201) 265-5000
TWX: 710-990-4951

NEW MEXICO
P.O. Box 11634
Station E
11300 Lomas Blvd., N.E.
Albuquarque 87123
Tel: (505) 292-1330
TWX: 910-989-1185

156 Wyatt Drive
Las Cruces 88001
Tel: (505) 526-2485
TWX: 910-983-0550

NEW YORK
6 Automation Lane
Computer Park
Albany 12205
Tel: (518) 458-1550
TWX: 710-441-8270

201 South Avenue
Poughkeepsie 12601
Tel: (914) 454-7330
TWX: 510-248-0012

39 Saginaw Drive
Rochester 14623
Tel: (716) 473-9500
TWX: 510-253-5981

5858 East Molloy Road
Syracuse 13211
Tel: (315) 454-2486
TWX: 710-541-0482

1 Crossways Park West
Woodbury 11797
Tel: (516) 921-0300
TWX: 710-990-4951

NORTH CAROLINA
P.O. Box 5188
1923 North Main Street
High Point 27262
Tel: (919) 885-8101

OHIO
16500 Sprague Road
Cleveland 44130
Tel: (216) 243-7300
TWX: 810-423-9431

330 Progress Rd.
Dayton 45449
Tel: (513) 859-8202
TWX: 810-474-2818

1041 Kingsmill Parkway
Columbus 43229
Tel: (614) 436-1041

OKLAHOMA
P.O. Box 32008
Oklahoma City 73132
Tel: (405) 721-0200

OREGON
17890 SW Lower Boones
Ferry Road
Tualatin 97062
Tel: (503) 620-3350

PENNSYLVANIA
111 Zeta Drive
Pittsburgh 15238
Tel: (412) 782-0400
TWX: 710-795-3124

1021 8th Avenue
King of Prussia Industrial Park
King of Prussia 19406
Tel. (215) 265-7000
TWX: 510-660-2670

SOUTH CAROLINA
6941-0 N. Trenholm Road
Columbia 29260
Tel: (803) 782-6493

TENNESSEE
***Knoxville**
Medical Services only
Tel: (615) 523-5022

***Nashville**
Medical Service only
Tel: (615) 244-5448

TEXAS
P.O. Box 1270
201 E. Arapaho Rd.
Richardson 75080
Tel: (214) 231-6101

P.O. Box 27409
6300 Westpark Drive
Suite 100
Houston 77027
Tel: (713) 781-6000

205 Billy Mitchell Road
San Antonio 78226
Tel: (512) 434-8241

UTAH
2160 South 3270 West Street
Salt Lake City 84119
Tel: (801) 487-0715

VIRGINIA
Medical Only
P.O. Box 12778
No. 7 Koger Exec. Center
Suite 212
Norfolk 23502
Tel:(804) 497-1026/7

P.O.Box 9854
2014 Hungary Springs Road
Richmond 23228
Tel. (804) 285-3431

WASHINGTON
Bellefield Office Pk.
1203-114th Ave. S.E.
Bellevue 98004
Tel. (206) 454-3971
TWX: 910-443-2446

***WEST VIRGINIA**
Medical/Analytical Only
Charleston
Tel: (304) 345-1640

WISCONSIN
9004 West Lincoln Ave.
West Allis 53227
Tel: (414) 541-0550

FOR U.S. AREAS NOT LISTED:
Contact the regional office
nearest you: Atlanta, Georgia...
North Hollywood, California...
Rockville, Maryland...Rolling Meadows,
Illinois. Their complete
addresses are listed above.

***Service Only**

Subject Index

a

Abort byte 55,61
Abort message (cli) 14,30,31
Abortive interrupts 61
Add 10
Address, HP-IB 16,17,35
  codes 88
  non-active controller 17
AND (band) 7
ASCII table 91
Automatic interrupt 68
Autostart 41

b

Benchmarks, buffered I/O 92
Binary AND (band) 7
Binary representation 5
Bit function 10
Bit bucket 76
Brackets, square 2
Buffer
  DMA 68,70
  fast read/write 68,69
  interrupt 68,69
  overflow 70
  pointers 75
  status 74
  string variable 77
  types 68
  underflow 70
Buffer statement (buf) 70
Buffered I/O 67
  benchmarks 92
Byte (8 bits) 71

c

Clear interface (cli – abort) 30,31
Clear lockout/set local message 14,24
Clear message (clr) 14,21
Code conversion 44,48
Command codes (HP-IB) 88
Command (cmd) 19,20,35
Complement 5
Complement (cmp) 5,8
Controller 13,17,31,87Conversion table (ctbl) 44

d

Data, inverted 77
Data message 14
  sending 19
  receiving 20
Data transfer
  output 72
  input 73
Decimal mode (mdec) 5,6
Decimal to octal (dto) 7
Device address 16
Device (dev) 17
Direct memory access (DMA) 64,68,70
Dot matrix 2
Drivers, I/O 51

e

Enable Interrupt (eir) 53,56,63
Equate (equ) 37
Error line (erl) 42
Error number (ern) 42
Error Recovery42 ..... Errors 103
  buffer underflow or overflow (E5) ... 70
  messages 103
  out of range (E6) 5
  parity (E7) 47
  time limit (E4) 42
  unnecessary parameters 2
Exclusive OR (eor) 7
Extended Address 16
Extended I/O
  description 1
  modes 90
  status conditions 90
Extended read status (rds) 38

f

Find file (fdf) 64
Flags 14 & 15 5
Format (fmt) 18,71

g

General I/O error messages 105
Global variables 64

Subject Index

h

HP-IB 13,85
interface functions 88
interrupt 58
lines 85
messages 13
operations 15
sample application 32

i

Inclusive OR (ior) 8
Inspection 2
Installation 2
Interface registers 49
Interrupt
abortive 61
application 57
automatic 68
end-of-line (EOL) 54
HP-IB 58
lockouts 64
programmable 53
vectored (EOL) 54
Interrupt enable (eir) 54,56
Interrupt return (iret) 54,57
I/O
drivers 51
buffered 67
I/O flag (iof) 50
I/O status (ios) 50

l

List 18,19
Listenaddress 16,35
Listener 18
Live Keyboard53,55 Local lockout message (llo) 14,23
Local message (lcl) 14,23
Logical operators (and, or, xor, not) 5

m

Modes
Extended I/O 90
Octal/Decimal 6
Multiple listeners 18

n

Non-active controller 30,31
Non-active controller address 17

o

Octal mode (moct) 6
Octal to decimal (otd) 7
On error (on err) 42
On interrupt (oni) 54,55
OR
exclusive (eor) 7
inclusive (ior) 8
Overflow, buffer 70

p

Parity (par) 47
Pass control message (pct) 14,30
Poll (pol) 20,29
Poll configure (polc) 30
Poll unconfigure (polu) 30
Polling 24
parallel 24,29
serial 24,26

r

Range of integers 5
Read binary (rdb) 20
Read interface (rdi) 50
Read-only variables 42
Read (red) 20
Read status (rds) 20,38
buffer 74
serial polling 26
Read/write memory 1

Remote message (rem) 14,22
Require service message (rqs) ... 14,24,25
Requirements 3
Rom (read-only variable) 42
Rotate (rot) 8

s

Sales and Service Offices 108
Select codes 16,17
Serial polling 24,26
Service Requests 24,26
Shift (shf) 9
Speed of peripherals 67,69
Status bit message 14
  receiving (parallel polling) 29
  sending 28
Status byte message 14
  receiving (serial polling) 26
  sending 26
Status bytes 38
Status conditions 90
String variable buffers 77
Syntax 2,97

t

Timeout (time) 42
Transfer parameters 16
Transfer (tfr) 19,20,31,72
Trigger message (trg) 14,20
Truth tables 7,8
Types of buffers 68

u

Underflow of buffer 70
Unlisten command 16,36

v

Variables 42,64
Vectored Interrupt 54

w

Word (16-bits) 71
Write Control 63
Write interface (wti) 49

HEWLETT hp PACKARD

PART NO. 09825-90025
MICROFICHE NO. 09825-99025

PRINTED IN U.S.A.
Oct. 1, 1976