

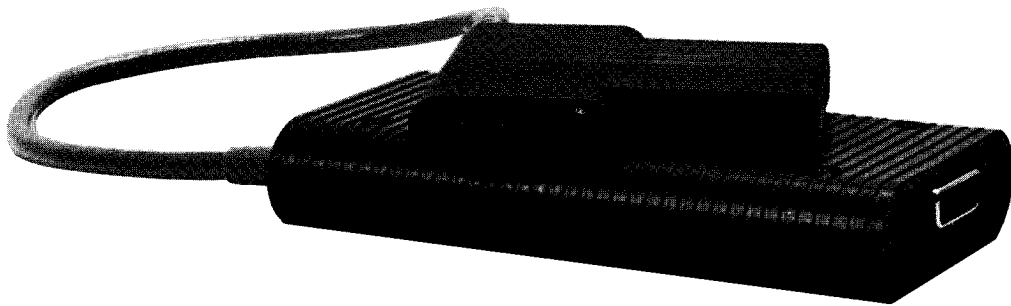
Hewlett-Packard 9815A/S Calculator
98135A HP-IB Interface
Operating and Service



Operating and Service

TECHNICAL COMPUTER
GROUP - MELBOURNE
LIBRARY COPY

HP 98135A HP-IB Interface



Hewlett-Packard Desktop Computer Division
3404 East Harmony Road, Fort Collins, Colorado 80525
(For World-wide Sales and Service Offices see back of manual.)
Copyright by Hewlett-Packard Company 1976



TABLE OF CONTENTS

General Information

Introduction	1
Equipment Supplied	1
Installation	1
Calculator Option 002	1
Connecting the Interface	2
Interconnecting Cables	2
Cable Length Restrictions	3
Metric Conversion Kit	3
Technical Specifications	3
Bus Signal Lines	4
Memory Usage	4
Logic Levels	4
Select Code	4
Bus Address Characters	4
Operating Temperature	5
Power Requirements	5
Data Transfer Rate	5
Overview of the HP-IB	5
Bus Messages	6
Transfer Parameters	7

Section 2: Operation

Key Overlay	9
Definition of Terms	10
Interface Parameters	10
Register Number Parameter	10
Byte Number Parameter	10
Number of Bytes Parameter	11
Address Code Parameter	11
Program Step Parameter	12
The Data Message	12
Data Output Instructions	14
The WRITE X Instruction	14
The FIELD Instruction	15
The WRITE BYTE Instruction	15
The OUTPUT Instruction	15
The COMMAND Instruction	18
Data Input Instructions	20
The READ X Instruction	20
The READ BYTE Instruction	22
The INPUT Instruction	22

HP Computer Museum
www.hpmuseum.net

For research and education purposes only.

Program Character Storage	25
The $\alpha \rightarrow \text{STR}$ Instruction	25
The $\text{STR} \rightarrow \alpha$ Instruction	26
Logical Operations	30
The AND Instruction	30
The OR Instruction	30
Program Instructions	30
The DUMP PROGRAM Instruction	31
The LOAD PROGRAM Instruction	31
Bus Control Messages	31
The COMMAND Instruction	31
The Trigger Message	32
The Clear Messages	32
The Device Clear Message	33
The Selected Device Clear Message	33
The Remote Message	33
The Local Message	33
The Local-Lockout Message	33
The Clear Lockout and Set Local Message	34
The Pass Control Message	34
The Abort Message	34
Service Requests and Polling	34
The Require Service Message	35
The Status Instruction	35
Responding to Service Requests	36
Serial Polling – The Status Byte Message	36
The SERIAL POLL Instruction	37
Sending Service Requests	37
Example Program	38

Section 3: Service

Address Code and System Controller Jumpers	41
Disassembly Procedure	41
The Address Jumpers	42
The System Controller Jumper	42
Theory of Operation	43
ROM Enable	44
ROM Power-Up	44
Control Logic	44
Input Multiplexer	44
Handshake Logic	44
HP-IB Functions	45
The Listener Function	45
The Talker Function	45

The Controller Function	45
The Serial Poll Function	45
The Device Clear Function	45
Replaceable Parts	46
XA1 Wire Connections	47
P2 Wire Connections	47
A1 Component Locator	48
A1 Schematic	49
A2 Component Locator	50
A2 Schematic	51
A3 Component Locator	52
A3 Schematic	52

Appendix

Binary Coding and Conversions	54
Binary-Decimal Conversions	54
Octal-Binary Conversions	54
ASCII	55
ASCII Character Codes	56
HP-IB Addresses	58
The HP Interface Bus	60
Interface Functions	63
Sales and Service Offices	66
Summary of the Interface Instructions	68
Subject Index	70
Error Messages	72

Figures

Installing the Interface	2
Standard HP-IB Cables	2
HP-IB Key Overlay	9
Disassembled A2 Board, A3 Board and Case	41
Address Jumpers on the A3 Board	42
System Controller Jumper on the A3 Board	42
Interface Block Diagram	43
A1 Component Locator	48
A1 Schematic	49
A2 Component Locator	50
A2 Schematic	51
A3 Component Locator	52
A3 Schematic	53
Bus Signal Lines	60
Functions Implemented on the 98135A Interface	64

Tables

Data Message Instructions	13
HP 3490A Control Characters	19
Replaceable Parts	46
XA1 and P2 Wire Connections	47
ASCII Character Codes	56
Available Bus Addresses and Codes	59
Command and Address Codes	62
HP-IB Interface Functions	63
Functions Used by the Bus Messages	64
Bus Functions Available with the 98135A Interface	65
Summary of the Interface Instructions	68
Error Messages	70

1 General Information

Introduction

The HP 98135A HP-IB Interface connects the HP 9815A Calculator to the HP Interface Bus. The interface conforms to the IEEE Standard 488-1975¹, allowing the calculator to perform a wide variety of operations via an HP-IB system.

This manual describes how to install, operate and service the 98135A Interface. In addition, a general description of HP-IB operation is given in this section.

Equipment Supplied

One of each of the following items is supplied with each 98135A Interface:

- HP 98135A HP-IB Interface Operating and Service Manual, HP P/N 98135-90000
- HP 98135A HP-IB Interface Quick Reference, HP P/N 98135-90010
- Key Overlay, HP P/N 1720-5355
- 2-meter cable, HP P/N 10631B



Installation

Calculator Option 002

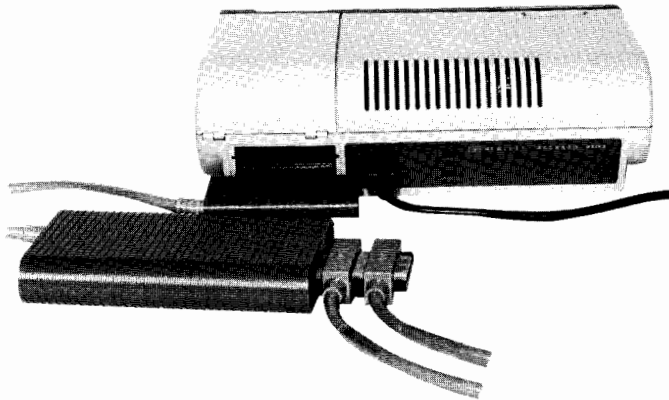
Before an interface can be installed, the calculator must be equipped with option 002, two channel I/O. This option provides the calculator with the interface connectors and internal I/O compatibility.

¹IEEE Standard Digital Interface for Programmable Instrumentation. This standard describes the functional, electrical, and mechanical elements of this interface system.

Connecting the Interface

The procedure used to connect the interface to the calculator is as follows:

1. Switch the calculator off.
2. Insert the interface into either of the I/O slots at the back of the calculator as shown below. Press the card firmly into the slot.
3. Switch the calculator back on.

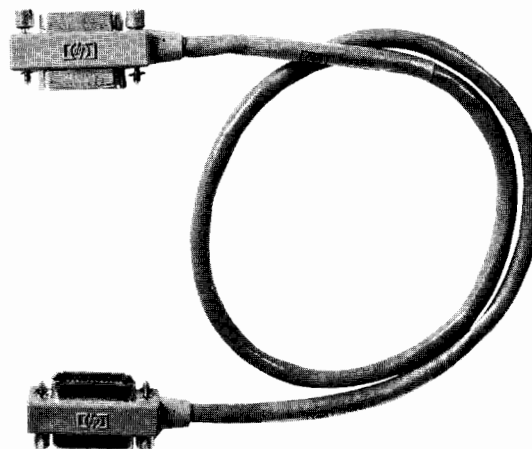


Installing the Interface

Interconnecting Cables

A 2-meter cable is supplied with the bus card. Other devices may be added to the bus by using the standard bus cables listed below.

Length	Accessory Number
1 meter	10631A
2 meters	10631B
4 meters	10631C



Standard HP-IB Cables

Cabling Length Restrictions

In order to ensure proper operation of the bus, two rules must be observed regarding the total length of bus cables when they are connected together:

- The total length of cable permitted in one bus system must be less than or equal to two meters times the number of devices connected together (the interface card is counted as one device).
- The total length of cable must not exceed 20 meters.

For example, a system containing 6 devices can be connected together with cables that have a total length less than or equal to 12 meters (6 devices \times 2m/device = 12m). The individual lengths of cable can be distributed in any manner desired as long as the total length does not exceed the allowed maximum. If more than 10 devices are to be connected together, cables shorter than 2 meters must be used between some of the devices to keep the total cable length less than 20 meters.

The maximum number of devices that can be connected together in one bus system is 15.

There are no restrictions to the ways cables may be connected together. However, it is recommended that no more than four piggy-back connectors be stacked together on one device. The resulting structure could exert enough force on the connector mounting to damage it.

Metric Conversion Kit

The HP-IB cable furnished with the 98135A Interface is supplied with mounting fasteners having metric threads. Other HP-IB instruments, however, may have either National Coarse (American) threads or metric threads. The American threaded fasteners are chromium plated, while the metric threaded fasteners are black.

Since metric and American threads cannot be connected together, a conversion kit is available. Use this kit to replace the mounting fasteners on any bus cable. The conversion kits can be ordered using HP Part Number 5060-0138.

Technical Specifications

The interface's electrical characteristics are listed below. For complete details on HP-IB electrical, mechanical, and timing requirements, refer to IEEE Standard 488-1975.

Bus Signal Lines

The bus consists of sixteen signal lines that are defined as follows:

DIO1	Data Input/Output 1
•	•
•	•
•	•
DIO8	Data Input/Output 8
DAV	Data Valid
NRFD	Not Ready for Data
NDAC	Data Not Accepted
IFC	Interface Clear
ATN	Attention
SRQ	Service Request
REN	Remote Enable
EOI	End or Identify

Memory Usage

When the HP-IB interface is plugged in, it uses 8 steps of program memory. This loss of 8 steps is indicated by the remaining number of steps shown in the display when the Program mode is set.

Logic Levels

All signals use negative-true logic
(low = logical 1) →

High >2.4V
Low <0.4V

Select Code

Each calculator interface has its own select-code number. The 98135A Interface is set to a select code of 5. This select code must be specified in each key sequence instruction. The select code is independent of the bus address characters that specify the calculator's talker and listener addresses.

Bus Address Characters

The interface is preset to ASCII talk address "U" and listen address "5". Any one of the 30 other pairs of talk/listen addresses can be selected on the card. The procedure used to change addresses is given in Section 3.

Operating Temperature

The operating temperature range is from 0° C thru 45° C.

Power Requirements

The calculator supplies all power for the card.

Data Transfer Rate

The 98135A Interface can transfer data at the rate of about 2.5k bytes (8 bit characters) per second. The actual I/O rate can be considerably slower, however, and is determined by the talker and listener(s) on line at any given time. The slowest device always determines the actual data rate.

Overview of the HP-IB

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

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

The interface system consists of 16 lines which are used to carry all data and control information. The bus structure is organized into three sets of signal lines:

- Data bus, 8 lines.
- Handshake or control, 3 lines.
- Interface management, 5 lines.

The data bus carries 8-bit data and control messages in bit-parallel, byte-serial form. Messages are transmitted bi-directionally and asynchronously. The handshake and management lines are used to control data transfer and timing on the bus. A more detailed description of the bus structure is given in the Appendix.

Bus Messages

The programmable capabilities of each device on the HP-IB can be exercised by using one or more of the HP-IB messages described here. A message constitutes a quantity of information to be transferred between devices on the bus. Messages can be transferred between:

- Device and Device
- Controller and Device
- Controller and Controller

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

Device Communicatons:

- Data – A string of data characters that are transferred between devices by various calculator instructions.

Device Control:

- Trigger – Causes a group of selected devices to simultaneously initiate a device-dependent action.
- Clear – Initializes device-dependent functions to a predefined state.
- Remote – Switches selected devices to remote operating, 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 them to revert to local control.

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 – Not available with the 98135A Interface.

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 in this chapter. Remember that most instruments must be set to Remote before they will respond to other bus messages, and that a data message is used to transfer programming characters and data between devices.



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 to receive the message.

Instruments having HP-IB capability are assigned unique 7-bit ASCII characters as their talker and listener addresses. These address characters are used to specify the transfer parameters for messages sent on the bus.

The 98135A Interface uses an address code parameter to specify the transfer parameters for messages sent on the bus. Each instrument's ASCII talker and listener address characters can be converted to a single decimal value between 1 and 30. A complete explanation of the address code parameter is given in Section 2.

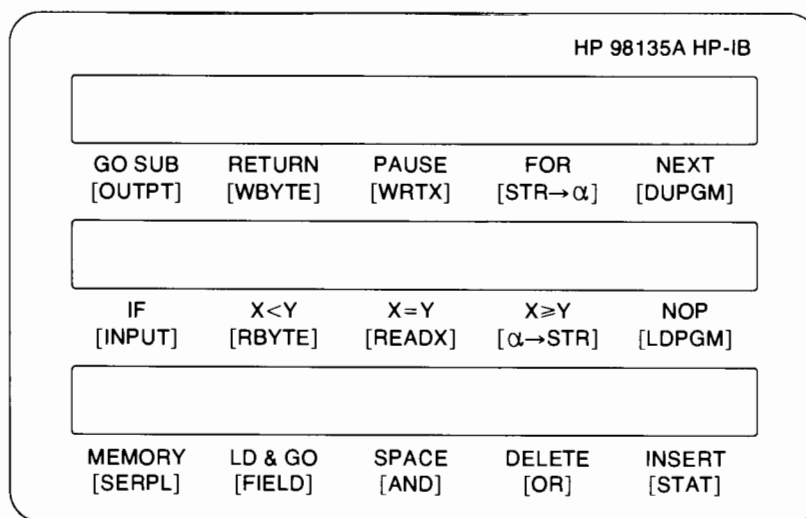
NOTES

2 Operation

This section explains the calculator instructions used for controlling systems via the HP Interface Bus. The preceding section should be read to ensure that the interface is properly installed. You should also know the bus operations for each device in your HP-IB system. This information can be found in their operating manuals. You should also be familiar with the operation of the calculator, as described in the HP 9815A Operating and Programming Manual.

Key Overlay

The interface provides the calculator with a set of key-sequence instructions that are used to control various operations on the bus. The instructions can be executed from either the keyboard or a program. A key overlay (HP P/N 7120-5355) showing the HP-IB instructions is provided with the interface. The overlay (shown below) should be placed over the keys A through O.



HP-IB Key Overlay

Definition of Terms

The following terms are used in this section.

Byte – One 8-bit binary value in either binary or decimal representation.

Data Value – A numeric value (in either fixed or floating-point form) or a decimal equivalent of an 8-bit binary byte.

Data – One or more ASCII coded characters or binary bytes.

Interface Parameters

Most of the interface instructions require that parameter values be entered into specific stack registers before the instruction is executed. These parameters specify such things as the register number and byte position of data being stored or output, bus address codes and program step addresses.

Register Number Parameter

The register number parameter specifies the storage register that an instruction will use to either store or recall data. If a register that has not been allocated is specified as a parameter, the calculator will print the error message "ILLEGAL ADDRESS" when the instruction is executed.

Instructions that can access either data registers or devices on the bus, interpret this parameter's function according to its sign.

A negative parameter specifies a register number and a positive parameter specifies a bus-address code. The format used to specify a bus-address code is described under "Address Code Parameter" in this section.

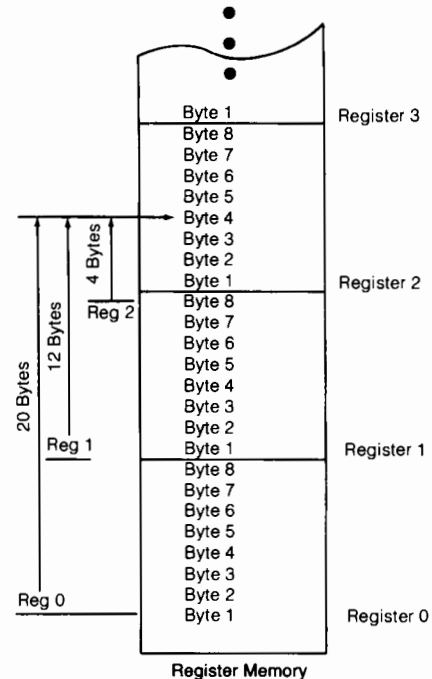
Byte Number Parameter

This parameter must be used whenever a register number parameter is specified. Together, they locate the particular data-byte location to be used by an instruction. A value of either 0 or 1 will reference the first byte of the specified register. Each register contains 8 byte positions (1 through 8).

The figure below shows how the register number and byte number parameters are used by the calculator to locate specific byte locations in the register memory.

The calculator counts the specified number of bytes up from the first byte of the register number specified. For example, the shaded byte location at the right can be referenced by any of the following byte and register number combinations:

- Register 0, Byte 20
- Register 1, Byte 12
- Register 2, Byte 4



Number of Bytes Parameter

The number of bytes parameter determines how many data bytes will be input, output or stored by an instruction. If the value is positive, the instruction will terminate after transferring the specified number of data bytes. A negative value will terminate the instruction when either the specified number bytes or a line feed character (LF) has been sent. If more bytes are specified than are available in the currently allocated data registers, the instruction will terminate after using all of the available data bytes and then print the error message "ILLEGAL ADDRESS".

Address Code Parameter

The HP-IB interface uses an address code value in the place of each device's ASCII address characters. Each pair of talker and listener address characters can be converted into a single decimal value between 01 and 30. A table showing the decimal value for each of the available talker and listener addresses is given on page 59. Using the address code, the interface automatically sends the appropriate ASCII character address (either talker or listener) for each instruction that is executed.

Each address code must consist of two digits between 01 and 30. If the value is a one digit number (e.g., 7) a leading zero must be added (e.g., 07).

The address code parameter must be entered into the appropriate stack register as a number in one of the following forms.

TT.L₁L₁L₂L₂L₃L₃L₄L₄

This parameter specifies the address of a device (TT.) that will send data to the calculator and, optionally, up to four additional listeners (.L₁L₁L₂L₂L₃L₃L₄L₄). For example, the value 16.12030521 addresses a device with an address code of 16 to send data and the devices with address codes of 12, 03, 05 and 21 to receive the data. The calculator's listener address is sent automatically and does not need to be included in the address parameter.

.L₁L₁L₂L₂L₃L₃L₄L₄L₅L₅

This parameter is used to specify from one to five listener devices that will receive data from the calculator. For example, .1203052125 addresses devices with address codes 12, 03, 05, 21 and 25 to receive data from the calculator. The calculator's talker address is sent automatically and does not need to be included in the address parameter.

An address parameter of 0 will disable the automatic addressing feature. The automatic addressing feature must be disabled whenever the calculator is not the controller in a system.

Program Step Parameter

This parameter specifies the program step address that an instruction will begin with when either inputting or outputting a string of program steps.

The Data Message

The data message refers to the string of data characters transferred between a talker and one or more listeners by a calculator instruction. Each data message typically consists of a string of 8 bit bytes in either ASCII or binary code.

Listed below is a brief summary of the calculator instructions that are used to send data messages on the bus. A detailed description of each instruction is given later in this section.

Data Message Instructions

Instructions	Parameters Entered Into the Stack				Description
	T	Z	Y	X	
Output Instructions WRITE X (WRTX) <small>CALL ALPHA</small> 5 <small>C</small>		Byte No	+(Address code) or -(Register No.)	Data Value	Sends the contents of the X register to the bus or stores it in calculator memory as ASCII coded characters with CR/LF.
FIELD <small>CALL ALPHA</small> 5 <small>L</small>				Field Width (1 thru 255)	Specifies the total number of characters sent by WRTX. Value is sent right-justified with leading spaces filling the excess field width.
WRITE BYTE (WBYTE) <small>CALL ALPHA</small> 5 <small>B</small>		Byte No.	+(Address Code) or -(Register No.)	Decimal Equivalent (0 thru 255)	Sends the binary equivalent of the value in X to either the bus or to a particular data register byte location.
OUTPUT (OUTPT) <small>CALL ALPHA</small> 5 <small>A</small>	Address Code	± Number of Bytes	Byte No.	Register No.	Sends a string of binary bytes from calculator register storage to the bus.
COMMAND (CMD) <small>CALL ALPHA</small> 5 <small>RUN STOP</small> (ASCII Character Keys) <small>CALL ALPHA</small>			Address Code	Data Value	Sends a string of ASCII characters to the devices specified by the address code parameter in Y.
Input Instructions READ X <small>CALL ALPHA</small> 5 <small>H</small>			Byte No.	+(Address Code) or -(Register No.)	Inputs ASCII coded data values from the bus or recalls a value from the calculator memory and enters the value in X.
READ BYTE (RBYTE) <small>CALL ALPHA</small> 5 <small>D</small>			Byte No.	+(Address Code) or -(Register No.)	Inputs a binary byte from the bus or recalls a specified byte from calculator memory. The decimal equivalent is entered into X.
INPUT <small>CALL ALPHA</small> 5 <small>F</small>	Address Code	± Number of Bytes	Byte No.	Register No.	Inputs a string of binary bytes from the bus and stores them directly into calculator data registers.
Program Instructions DUMP PROGRAM (DUPGM) <small>CALL ALPHA</small> 5 <small>E</small>			Address Code	Program Step No.	Sends the contents of the program memory in machine code to the bus.
LOAD PROGRAM (LDPGM) <small>CALL ALPHA</small> 5 <small>J</small>			Address Code	Program Step No.	Inputs program steps from the bus until an END step has been loaded.

Data Output Instructions

Note

If the calculator is not the active controller in the system when an output instruction is executed, the automatic addressing feature must be disabled by specifying 0 for the address code parameter. The calculator must also be addressed as a talker by the controller before the instruction is executed.

The WRITE X Instruction



Byte Number → Z

+(Address Code)

or → Y

-(Register Number)

Numeric Data Value → X

The WRITE X Instruction (WRTX) sends the sign, digits and decimal point of the current value in X, in ASCII code, to either the specified instruments on the bus or to the calculator data register memory. The value is sent or stored, right-justified, in a field of spaces and in the form set by the current calculator number format (FIX, SCI, or SCI 3). The size of the field can be varied and is explained next. CR and LF (carriage return and line feed) characters are automatically sent after the number. If the number is too large to be sent in a FIX format, a SCI format is automatically used.

To send the value in X to the bus, the appropriate address code value is entered into Y in the form .L₁L₁L₂L₂L₃L₃L₄L₄L₅L₅ for a maximum of five addresses. Each address code must consist of two digits; if a value has only one digit, a leading 0 should be added to it. A table of address code values for the ASCII address characters is provided in the Appendix. When the Y parameter is an address code, a parameter in Z is not required. **If the calculator is not active controller, a zero must be entered in Y to disable the automatic addressing.**

To store the value in X in the calculator memory, the desired data register number is entered into Y as a negative integer and the beginning byte number is entered into Z. The value is stored in consecutive byte locations beginning with the location referenced by the byte number and register number parameters. Each value is stored in ASCII code with one byte location used for each character (sign, digits, decimal point and the number of leading spaces required to fill the current field width) plus two additional bytes for the CR/LF characters. For example, a value in a field of 16 characters with CR/LF would occupy 18 bytes of data memory.



The FIELD Instruction



Field Width (1 thru 255) → X

The data field width used for WRITE X instructions is automatically set to 16 characters when the calculator is first switched on. To change this field width, enter the desired width into X and execute the FIELD instruction. The allowable range is from 1 thru 255 character spaces. If a number is too large to be sent or stored in the currently-set field, the calculator prints the error message "FIELD TOO SMALL" and the instruction is not executed.

The WRITE BYTE Instruction



Byte Number → Z

+(Address Code)

or → Y

-(Register Number)

±(0 thru 255) → X

The WRITE BYTE instruction (WBYTE) sends the 8 bit binary equivalent of the absolute integer value in X. This value can range from 0 thru 255. If the value specified is greater than 255 the calculator will print the error message "NUMBER TOO BIG", and terminate without sending a value to the bus.

To send the value in X to the bus, the appropriate address code values are entered into Y in the form. L₁L₁L₂L₂L₃L₃L₄L₄L₅L₅ for a maximum of five 2 digit values. If the calculator is not the controller, 0 must be entered in Y to disable the automatic addressing.

To store the value in X at a specific byte position, the desired register number is entered in Y as a negative integer, and the specific byte number is entered in Z.

The Output Instruction



Address Code → T

Number of Bytes → Z

Byte Number → Y

Register Number → X

The OUTPUT instruction (OUTPT) is used to send a specific number of stored data bytes to instruments on the bus.

The address code values are entered into T in the form .L₁L₁L₂L₂L₃L₃L₄L₄L₅L₅, for maximum of five 2-digit values. **If the calculator is not the active controller, 0 must be entered in T to disable the automatic addressing.**

The number of bytes to be sent to the bus is entered into Z. A positive parameter does not recognize a LF character as a terminating delimiter and a negative parameter does.

The output begins at the byte location established by referencing the byte-number entered in Y and the register number entered in X.

The following example program uses FIELD, WRITE X, OUTPUT, and WRITE BYTE instructions to print a number, its squared value and its square root value in a three column format using a 9871A Option 001 Printer. The printer's address code is 01.

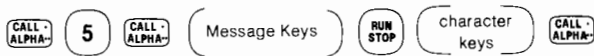
The first number is stored in a field of 10 character spaces by a WRTX instruction (steps 0011 thru 0020). This stored value occupies the first 10 bytes, starting at register 0, byte 1. The CR and LF characters that are automatically sent by WRTX are stored at bytes 11 and 12. Next, the squared value is stored in a field of 14 characters by another WRTX instruction (steps 0022 thru 0034). This value is stored from byte 11 thru byte 24. Note that this value is stored over the CR and LF characters of the previous WRTX instruction. These characters must be eliminated in this manner from all but the last value in each row to allow a column format to be printed. The last value is stored in a field of 17 characters in bytes 25 thru 43, which overlaps the CR and LF of the previous WRTX (steps 0036 thru 0049). The entire block of data characters now occupies bytes 1 thru 43 when referenced from the first byte of register 0. The block is sent to the printer (address code 01) by the OUTPUT instruction (steps 0051 thru 0058). A WBYTE instruction is used to send an LF character (decimal 10) to the printer after 10 rows of values have been printed. Shown below is the program printout and the resulting table.

0000	0		
0001	STO	A	
0002	STO	B	
0003	I		
0004	0		
0005	STO	F	
0006	STO	G	
0007	FOR	A→F	
0008	FOR	B→G	
0009	FIX	0	
0011	I		
0012	0		
0013	FIELD	5	

} Initial Register preparation
for for-next loops.

} Set Fix 0 number format and
the field width to 10 spaces.

The COMMAND Instruction



Address Code → Y
 Numeric Data Value → X

The COMMAND instruction (CMD) is used to send the various bus messages. The COMMAND instruction is divided into two sections that are separated by the (space) key. All of the bus messages, except data messages, are sent in the section before the key (with ATN “true”). The key clears the ATN line (sets it “false”) and any characters sent after that key are sent as data. This section is used to send strings of ASCII data characters to the addressed devices on the bus.

The various BUS control messages are explained under “BUS Control Messages” on page 31. Only the portion of the COMMAND instruction that sends ASCII data characters (the portion following the key) is explained next.

The addressing for both sections of the CMD instruction is done automatically by the interface using the address code parameter in Y. **If the calculator is not the active controller in the system, 0 must be entered into Y in place of the address code to disable the automatic addressing.**



Address Code → Y
 Numeric Data Value → X

The key sequence shown above is used to send a string of ASCII characters to the bus. The address code of the devices that are to receive the data is entered into Y before the CMD key sequence is executed. A table of the keys that send the various ASCII characters is in the Appendix. To terminate the CMD mode, press again.

Since the interface provides the calculator with the capability of sending more characters than there are keys on the keyboard, there are three keyboard states available. Pressing sets the “SHIFT” state; the keys in the table prefixed by an S indicate the SHIFT state. Pressing sets the “TAB” state; the keys in the table prefixed by a T indicate the TAB state. The keys that are not prefixed by an S or a T are in the unshifted keyboard state. To cancel either the SHIFT or TAB state, press the key that set the state. To change from one state to another, press the key that sets the new state. At the start of a CMD instruction, the keyboard is in the unshifted state. Pressing from any of the states executes a WRTX instruction without sending the CR/LF characters.

The editing procedure for the CMD instruction is the same as for the calculator Alpha mode. See “Editing Alpha” in Section 3 of the HP 9815A Operating and Programming Manual.

For example, the HP 3490 Multimeter can be programmed remotely by sending specific ASCII characters to it. Here is a list of its control characters:

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

The program segment below uses the CMD instruction to send the appropriate characters to the 3490 to set it to a desired range and function and to take a reading. The READX instruction inputs the resulting data reading. Note that the address code 22, entered into X and Y in steps 0027 thru 0029, is sent as the voltmeter's listener address by the CMD instruction and then is sent as the voltmeter's talk address by the READX instruction.

```

0024
0025
0026
0027 2          Address Code
0028 2
0029 ENTER↑
0030 CMD      5
0032
0033 F          DC Volts Function
0034 0

0035 R          10V Range
0036 4
0037 T          Internal Trigger
0038 1
0039 M          Output Mode
0040 3
0041 E          Execute reading
0042 END↵
0043 READX    5
0045

```

Data Input Instructions

Note

If the calculator is not the active controller in the system when an input instruction is executed, the automatic addressing feature must be disabled and the calculator must be addressed as a listener by the controller before the instruction is executed.

The READ X Instruction



Byte Number → Y
 +(Address Code)
 or → X
 -(Register Number)

The READ X instruction inputs ASCII coded values and enters them into the stack registers. Data values can be read from either an instrument on the bus or from the calculator memory.

A data value can consist of a plus or minus sign, the digits 0 through 9, a decimal point and an exponent. An exponent is specified by an E, a plus sign, or a minus sign in any one of these forms:

+ or - (digits) E (exponent digits)
 + or - (digits E + or - (exponent digits)
 + or - (digits) + or - (exponent digits)

To input data values from an instrument on the bus, the instrument's address code is entered into X as a positive two-digit integer. **A zero is entered into X if the calculator is not the active controller in the system.**

The instruction will ignore all non-numeric characters input until a data value is read. Then, after reading the data value, reading any non-numeric character will enter the value into X, perform a stack up operation on any values already in the stack, and continue reading characters until another data value is encountered. If more than four values are input with a READX, only the last four will remain in the stack. A space character is always ignored by READX and a line feed character will terminate the instruction. EOI will also terminate a READX (EOI is set automatically by a talker after the last data value is sent during an operation).



To read a data value from the calculator memory, the first byte position to be read is specified by entering the register number in X as a negative integer and the specific byte number in Y.

The instruction reads the bytes in the same manner specified for data being input from the bus, except that the first non-numeric character (except a space character) read after the data value terminates the instruction. The value is entered in X and a stack up operation occurs.

This program uses an HP 59309 Digital Clock to time a ten minute interval between voltage readings taken from an HP 3490A Multimeter. The clock's output is formatted as follows:

(space) (space) MMDDHHMMSS (CR) (LF)

The program begins by taking an initial time reading from the clock (address code 16) with a READX and stores the value in Register A (steps 0000 thru 0004). 1000 is added to the value in A, which increases the time reading by 10 minutes (steps 0005 thru 0009). Another time reading is taken from the clock and it is compared with the value in register A (steps 0010 thru 0018). If the reading is greater than the value in A, the program branches to step 0020. If the reading is less than the value in A, the program branches to step 0012 and takes another reading. At step 0020, the multimeter (address code 22) is triggered by an "E" from the CMD mode. The resulting voltage reading is input with a READX and printed (steps 0020 thru 0031). The program branches to step 0005 where 1000 (10 minutes) is again added to the value stored in register A and the program is repeated.

```
0000 1          Clock Address Code      0016 GOTO      0020
0001 6
0002 READX    5
0004 STO      A
0005 1
0006 0
0007 0
0008 0
0009 STO+    A
0010 1          Clock Address Code      0018 GOTO      0010
0011 6
0012 READX    5
0014 RCL     A
0015 IF XY
0020 2          Voltmeter Address Code
0021 2
0022 ENTER↑
0023 CMD      5
0025
0026 E
0027 ENDα
0028 READX    5
0030 PRINT
0031 SPACE
0032 GOTO     0005
0034 END
```

The READ BYTE Instruction



Byte Number → Y
 +(Address Code)
 or → X
 -(Register Number)

The READ BYTE instruction (RBYTE) inputs one 8 bit binary byte from either an instrument on the bus or the calculator memory. The byte is entered into X as its decimal equivalent value. The usable range is from 0 thru 255.

To read a byte from an instrument on the bus, the instrument's address code is entered in X as a positive two digit integer. **A zero must be entered in X if the calculator is not the active controller in the system.**

To read a byte from the calculator memory, the byte location is specified by entering the register number in X as a negative integer and the specific byte number in Y.

The INPUT Instruction



Address Code → T
 ± Number of Bytes → Z
 Byte Number → Y
 Register Number → X

The INPUT instruction inputs a string of binary bytes from an instrument on the bus and stores them in consecutive bytes of data register memory.

The address code of the instrument that will output the data is entered in T. **A zero must be entered in T if the calculator is not the active controller in the system.**

The total number of bytes to be input by the instruction is entered in Z. The instruction will terminate after inputting the specified number of bytes if the parameter is positive. If the parameter is negative, the instruction will also recognize a line feed character (LF) as a terminator.

The instruction begins storing the incoming bytes at the byte location specified by the byte number parameter in Y and the register number parameter in X. If enough data registers are not allocated to hold the number of bytes being input, the calculator fills all available byte locations, prints the error message "ILLEGAL ADDRESS", and terminates the instruction.

The following example takes a reading from an HP 59309 Digital Clock and prints the hours, minutes and seconds as follows:

```

HOURS =    11

MINUTES =   22

SECONDS =   33

```

The clock's output is formatted as follows:

(space) (space) MMDDHHMMSS (CR) (LF)

The reading from the clock (address code 16) is taken with an INPUT instruction (steps 0002 thru 0011). Note that the negative parameter in Z causes the instruction to terminate after inputting the LF character. A for-next loop (C → HD) is initialized to begin at 7 and end at 12 in increments of 2 (steps 0013 thru 0019). The value stored in register B (10000) is used in the calculation that separates the desired digits from the reading (steps 0020 thru 0023). The value in register C is used as the byte number parameter of the READX instruction which returns a value, beginning with the 7th byte of register 0, into X (HHMMSS) (steps 0024 thru 0028). This value is divided by the value in register B (10000) and the integer function (INT) leaves only the two digits that represent the hours (steps 0030 thru 0033). The value in register B is divided by 100 for use in the next loop of the program (steps 0034 thru 0037). The number in register C is checked to determine which sub-routine should be used to print the value (steps 0038 thru 0048). The program loop is repeated with a 9 in register C to obtain the minute digits (MM) and finally it is repeated, with an 11 in register C, to obtain the seconds digits (SS).

Here is a listing of the program

```

0000 FIX      0
0002 1          Clock Address Code
0003 6
0004 ENTER↑
0005 1
0006 5          Number of Bytes
0007 +←-
0008 ENTER↑
0009 0          Starting Byte No. and
0010 ENTER↑    Register No.
0011 INPUT    5
0013 7
0014 STO      C
0015 1
0016 2
0017 STO      H
0018 2
0019 STO      D
0020 1
0021 EXE

```

```

0023 STO B
0024 FOR C+HD
0025 RCL C
0026 0
0027 +*-
0028 READX 5
0030 RCL B
0031 +
0032 INT
0033 STO A
0034 1
0035 0
0036 0
0037 STO+ B
0038 RCL C
0039 7
0040 IF X=Y
0041 GOSUB A
0043 RCL C
0044 9
0045 IF X=Y
0047
0048
0049
0050 GOTO 0118
0052 LBL
----- A
0054 RCL A
0055 PRNTα
0057 H
0058 0
0059 U
0060 R
0061 S
0062
0063
0064
0065 =
0066 PRINT
0067
0068
0069
0070 LINE
0071 ENDα
0072 NEXT C
0073 RETURN

```

Label A prints out:
"HOURS = HH" subroutine.

```

0074 LBL
----- B
0076 RCL A
0077 PRNTα
0079 M
0080 I
0081 N
0082 U
0083 T
0084 E
0085 S
0086
0087 =
0088 PRINT
0089
0090
0091
0092 LINE
0093 ENDα
0094 NEXT C
0095 RETURN
0096 LBL
-----
0097
0098 RCL A
0099 PRNTα
0101 S
0102 E
0103 C
0104 0
0105 N
0106 D
0107 S
0108
0109 =
0110 PRINT
0111
0112
0113
0114 LINE
0115 ENDα
0116 NEXT C
0117 RETURN
0118 SPACE
0119 SPACE
0120 SPACE
0121 SPACE
0122 END

```

Label B prints out:
"MINUTES = MM" subroutine.

Label C prints out:
"SECONDS = SS" subroutine

Program Character Storage

The $\alpha \rightarrow \text{STR}$ and $\text{STR} \rightarrow \alpha$ instructions are used to transfer strings of program characters from $\text{PRINT}\alpha$ strings in the program memory to data registers and back again. Each Alpha character is stored in ASCII code and occupies one byte of register storage.

The $\alpha \rightarrow \text{STR}$ Instruction




Program Step Number \rightarrow T
 Byte Number \rightarrow Z
 Register Number \rightarrow Y
 Numeric Data Value \rightarrow X

This instruction transfers the characters contained in a $\text{PRINT}\alpha$ string of a program to calculator data registers as ASCII coded bytes. The instruction begins the operation with the character at the program step specified in T and stores it at the byte position referenced by the byte number in Z and the register number in Y. The following string of characters is stored in consecutive byte positions, one byte per character until a step containing and "END α " is encountered or the end of the program memory is reached. Either of these conditions terminates the instruction and the error message "ILLEGAL ADDRESS" will result if the program memory limit is reached. If the instruction was executed from a program, the program continues from the next step after the $\alpha \rightarrow \text{STR}$ instruction. If a PRINT is encountered among a string of steps being transferred, the current data value in X is stored as described by the WRTX instruction, except that the CR/LF characters are not added.

For example, this program segment stores the binary form of the characters "HP-IB INTER-FACE" from the $\text{PRINT}\alpha$ string at step 0059 into consecutive byte locations starting with byte 1 of register 0.

```
0057 PRNT $\alpha$ 
0059 H
0060 P
0061 -
0062 I
0063 B
0064
```



```
0065 I
0066 N
0067 T
0068 E
0069 R
0070 F
0071 A
0072 C
0073 E
0074 END $\alpha$ 
0075 5
0076 9
0077 ENTER $\uparrow$ 
0078 1
0079 ENTER $\uparrow$ 
0080 0
0081 ENTER $\uparrow$ 
0082  $\alpha \rightarrow \text{STR}$  5
```

Program Step No.
 Starting Byte No.
 Register No.
 Data Value (Not Used)

The STR $\rightarrow\alpha$ Instruction



Program Step Number \rightarrow T

\pm Number of Bytes \rightarrow Z

Byte Number \rightarrow Y

Register Number \rightarrow X

The STR $\rightarrow\alpha$ instruction transfers a string of ASCII coded characters from register storage to the program memory. The characters begin loading at the program step specified in T. The instruction loads, in consecutive step addresses, the number of bytes specified in Z, starting with the byte position referenced by the byte number in Y and the register number in X. If the parameter in Z is positive, the instruction will terminate after loading the specified number of bytes or when the end of either program memory or register storage memory is reached. The error message "ILLEGAL ADDRESS" is printed if either memory limit is reached. **The last character loaded is always an END α instruction.** If the parameter in Z is negative, the instruction also terminates when a line feed character is encountered.

For example, the following program transfers a string of characters from a PRINT α instruction to data register storage. Two voltmeter readings are input and stored at specific byte location within this stored string. The string, which then contains the voltage readings, is transferred back to the PRINT α instruction and both readings are printed on one line.

The program begins by transferring the characters from the Alpha string at step 0057 to consecutive byte locations beginning with byte 1 of register 0 (steps 0000 thru 0006).

The registers A and F are initialized for a for-next loop from 1 thru 10 (steps 0008 thru 0013). Next, both voltmeters are addressed and receive an "E" character (which triggers them) from the CMD instruction (steps 0014 thru 0024). A 4 byte sample is input from the voltmeter at address code 15 and stored in bytes 1 thru 4 of register 0 by the first INPUT instruction (steps 0025 thru 0032). The next INPUT instruction stores the reading from the voltmeter at address code 18 in bytes 12 thru 15 of register 0 (steps 0034 thru 0043). The stored string of characters, which now contains both voltmeter readings, is transferred to the PRINT α string at step 0057 by the STR $\rightarrow\alpha$ instruction (steps 0045 thru 0053). The line comparing the two readings is printed out and the program is repeated 10 times.

0000	5			0039	1		
0001	7	Program Step No.		0040	2	Starting Byte No.	
0002	ENTER↑			0041	ENTER↑		
0003	0	Starting Byte No.		0042	0	Register No.	
0004	ENTER↑	Starting Reg. No.		0043	INPUT 5		
0005	ENTER↑	Data Value (Not Used)		0045	5		
0006	←+STR 5			0046	7	Program Step No.	
0008	1			0047	ENTER↑		
0009	STO A			0048	1		
0010	1			0049	6	Number of Bytes	
0011	0			0050	ENTER↑		
0012	STO F			0051	0	Starting Byte No.	
0013	FOR A→F			0052	ENTER↑	Register No.	
0014	.			0053	STR+← 5		
0015	1			0055	PRNT←		
0016	5	Voltmeter #1 Address Code		0057	1		
0017	1			0058	2		
0018	8	Voltmeter #2 Address Code		0059	3		
0019	ENTER↑			0060	4		
0020	CMD 5			0061	V		
0022				0062			
0023	E			0063			
0024	END←			0064	V		
0025	1	Voltmeter #1 Address Code		0065	6		
0026	5			0066			
0027	ENTER↑			0067			
0028	4	Number of Bytes		0068	1		
0029	ENTER↑			0069	2		
0030	0	Starting Byte No.		0070	3		
0031	ENTER↑	Register No.		0071	4		
0032	INPUT 5			0072	V		
0034	1			0073	LINE		
0035	8	Voltmeter #2 Address Code		0074	END←		
0036	ENTER↑			0075	NEXT A		
0037	4	Number of Bytes		0076	END		
0038	ENTER↑						

This example program uses the STR→ α instruction to transfer a formatted string of characters to a PRINT α instruction. A reading is input into register 0 from an HP 59309 HP-IB Digital Clock. Then the hours, minutes and seconds digits are formatted with colons between them in register 3. The resulting printout is in the following form:

HH : MM : SS

The digital clock output is formatted as follows:

(space) (space) MMDDHHMMSS (CR) (LF)

The time reading is input from the digital clock (address code 16) by the INPUT instruction (steps 0000 thru 0010). The reading is stored in consecutive byte locations beginning with byte 1 of register 0. The negative parameter in Z of the INPUT instruction terminates the input after the LF character.

A for-next loop (A → F) is initialized from 0 thru 2. The value in register C (7) corresponds to the byte location of the first hour's digit. The value in B will be used to specify consecutive byte locations of register 3, where the final printout values will be formatted.

The seventh byte (register C) is returned to X by the RBYTE instruction (steps 0026 thru 0029). The byte is then stored at byte location 1 of register 3 by a WBYTE instruction (steps 0031 thru 0036). The value in C is increased by 1 and used as the byte number parameter of the RBYTE instruction to return byte 8 of register 0 to X. This number is then stored at byte 2 of register 3 by the WBYTE instruction. Next, a colon (decimal value 58) is stored at byte 3 of register 3 by a WBYTE instruction (steps 0041 thru 0047). The loop is repeated and bytes 9 and 10 of register 0 are returned to X by the RBYTE instruction and stored at bytes 4 and 5 of register 3 by the WBYTE instruction. Another colon is stored at byte 6 of register 3 and the loop repeated for the final time. This time, bytes 11 and 12 of register 0 are stored at bytes 7 and 8 of register 3.

The STR→ α instruction transfers the 8 bytes of register 3 to steps 0064 thru 0071 (steps 0052 thru 0060). The PRNT α instruction prints the line and program stops.

0000	CLEAR			0041	RCL	B
0001	1	Clock Address Code		0042	3	Byte No.
0002	6			0043	+⇌-	Register No.
0003	ENTER↑			0044	ENTER↑	
0004	2	Number of Bytes (Negative parameter)		0045	5	ASCII Coded Colon (:)
0005	0			0046	8	
0006	+⇌-			0047	WBYTE	5
0007	ENTER↑			0049	1	
0008	0	Starting Byte No.		0050	STO+	B
0009	ENTER↑	Register No.		0051	NEXT	A
0010	INPUT	5		0052	6	Program Step No.
0012	7			0053	8	
0013	STO	C		0054	ENTER↑	
0014	0			0055	8	Number of Bytes
0015	STO	A		0056	ENTER↑	
0016	2			0057	0	Starting Byte No.
0017	STO	F		0058	ENTER↑	
0018	1			0059	3	Register No.
0019	STO	B		0060	STR+α	5
0020	FOR	A+F		0062	PRNTα	
0021	RCL	B		0064		
0022	1			0065		
0023	+			0066		
0024	STO	G		0067		
0025	FOR	B+G		0068	0	
0026	RCL	C	Byte No.	0069	0	
0027	0			0070	:	
0028	+⇌-	Register No.		0071	3	
0029	RBYTE	5		0072	3	
0031	RCL	B	Byte No.	0073	:	
0032	X⇌Y			0074	4	
0033	3	Register No.		0075	8	
0034	+⇌-			0076	ENDα	
0035	X⇌Y			0077	SPACE	
0036	WBYTE	5		0078	SPACE	
0038	1			0079	SPACE	
0039	STO+	C		0080	SPACE	
0040	NEXT	B		0081	END	



Logical Operations

The AND Instruction



0 thru 255 → Y

0 thru 255 → X

This instruction combines the binary equivalents of X and Y in an AND logic operation (i.e., $X \cdot Y$). The result is placed in X and the value in Y remains unchanged. The usable range is from 0 thru 255.

During an AND instruction, the numbers in X and Y are first converted to 8-bit binary numbers. Then each bit is compared with its corresponding Y bit; if both are 1, the resulting X bit is 1. But if either bit is 0, then the resulting X bit is 0. Finally, the result is converted to its decimal equivalent.

For example, if: 25 → Y(00011001)

37 → X(00100101)

The AND gives: 25 → Y(00011001)

1 → X(00000001)

The OR Instruction



0 thru 255 → Y

0 thru 255 → X

This instruction combines the 8 bit binary equivalents of X and Y in an inclusive OR logic operation (i.e., $X + Y$). The result is placed in X and the value in Y remains unchanged. The usable range is from 0 thru 255.

During an OR operation, each X bit is compared with its corresponding Y bit. If either bit is 1, the resulting X bit is 1. But if both bits are 0, the resulting X bit is 0. The final result is converted to its decimal equivalent and placed in X.

For example, if: 89 → Y(01011001)

45 → X(00101101)

The OR gives: 89 → Y(01011001)

125 → X(01111101)

Program Instructions

These instructions enable programs to be transferred between a device on the bus and the calculator. The program steps are transferred in an 8-bit binary code unique to the calculator.

The DUMP PROGRAM Instruction

Address Code → Y
Program Step Number → X

The DUMP PROGRAM instruction (DUPGM) sends the contents of the program memory, starting at the program step specified in X and stopping after an END instruction or at the end of the program memory. The program is sent to the devices specified by the address code entered in Y. **A zero must be entered in Y if the calculator is not the active controller in the system.**

Note

Programs for the 9815A Calculator are transferred in a complex binary code that is unique to the calculator. For this reason, only programs that have been sent by the DUMP PROGRAM instruction should be input by the LOAD PROGRAM instruction.

The LOAD PROGRAM Instruction






  

Address Code → Y
Program Step Number → X



The LOAD PROGRAM instruction (LDPGM) inputs previously dumped program steps from an instrument on the bus. The address code value of the instrument sending the program steps is specified as a two digit integer in Y. The program steps are loaded into the program memory, starting at the program step specified in X. The instruction terminates when an END instruction has been loaded.



Bus Control Messages

The COMMAND Instruction

   (Message Keys)  (character keys) 

Address Code → Y
Numeric Data Value → X

The COMMAND instruction (CMD) is used to send the various bus messages. The COMMAND instruction is divided into two sections that are separated by the  (space) key. All of the bus messages, except data messages, are sent in the section before the  key (with ATN

“true”). The  key clears the ATN line (sets it “false”) and any characters sent after that key are sent as data. If data characters are not sent after a particular bus message, the  key can be eliminated from the key sequence.

The interface will automatically address the bus according to the address code parameter in Y for all addressable messages sent from the CMD mode.

The Trigger Message



Address Code → Y

The Trigger message sends a Group Execute Trigger (GET) command to initialize present functions or actions on the devices specified by the address code value in Y.

The program segment shown here sends a trigger message from the CMD mode to a voltmeter (bus address code 12) and then inputs the reading with a READX instruction.

```

●
0021 1
0022 2 Voltmeter Address Code
0023 ENTER↑
0024 CMD 5
0026 @
0027 H
0028 END⊗
0029 READX 5
●
●

```

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.

The Clear Messages

Both of the clear messages initialize each device to its “power-on state”, as described in its operating manual. The calculator is not affected by either of the clear messages when it sends them. If the calculator is not the controller and it receives a clear message, the entire program memory is cleared and the calculator is set to its “power-on state”.

The Device Clear Message



The key sequence above sends the Device Clear message to all devices on the bus.


The Selected Device Clear Message



The key sequence above sends the Selected Device Clear message to only the devices specified by the address code in Y.

The Remote Message



The Remote Message (REN) sets the remote line on the bus. The  key then addresses the devices specified by the address code in Y which sets them to their remote state. They can now be programmed via data messages; see the COMMAND Instruction on page 18 for details about sending ASCII data characters. **The Remote Message can only be sent by the system controller.**

The Local Message



The key sequence shown above sends the Local message (GTL) to the devices specified by the address code in Y. The Local message cancels a Remote message and returns the specified devices to front panel control. **The Local message does not cancel a Local Lockout message.**

The Local Lockout Message



The Local Lockout message prevents local (front or rear panel) control of devices on the bus, to prevent accidental return to local control during a bus operation. To cancel local lockout, a Clear Lockout and Set Local message must be sent. The Abort (IFC) message does not cancel local lockout.

The Clear Lockout and Set Local Message



This key sequence sends the Clear Lockout and Set Local message (REN) to all devices on the bus. This message can be used to cancel Remote messages as well as cancelling Local Lockout messages. **This message can only be sent by the system controller.**

The Pass Control Message



Address Code → Y

Executing the above key sequence sends the Pass Control message to a specified device on the bus. This results in the device (which must be a talker) becoming the active controller. The calculator may now be addressed from the new controller. If the device that has become the active controller cannot pass control back to the calculator, the calculator (or system controller) must use an Abort message to halt all bus operations and regain active control.

The Abort Message



The Abort message (IFC) is sent to the bus to halt all bus operations and return control to the system controller. **The Abort message can only be sent by the system controller.**

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 a Require Service message to ask for the attention of the controller. The calculator typically uses the status instruction to check for a service request condition on the bus and then uses polling to locate the source and cause of the service request. Polling, however, is not limited to situations involving service requests.

The polling method available with the 98135A Interface is the serial poll. A device responds to a serial poll by sending a Status Byte message containing up to 8 bits of status information. Use of the serial poll enables receiving a full byte (8 bits) of information from one device at a time.

Every bus-compatible instrument 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 a service request. The operating manual for each device describes whether it can transmit Require Service messages and how the device responds to a poll.

When the calculator is not the active controller (either it passed control to another device or the System Controller Jumper on the interface is moved), it can transmit the Require Service message and be programmed to respond to a serial poll. This is described later, under "Sending Service Requests".

The Require Service Message

Some devices that operate on the HP-IB have the ability to request service from the active controller. A device may request service when it has completed a measurement, when it has detected a critical condition, or for any other reason. The operating manual for a device tells whether it can send a Require Service message and if so, for what purpose.

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.

The Status Instruction



CALL ALPHA 5 0

The STATUS instruction (STAT) returns two interface status bytes to the stack. The decimal equivalent of the binary bytes are entered in X and Y as follows:

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
y =	End or Identify	Remote Enable	Service Request	Interface Clear	Attention	Not Data Accepted	Not Ready For Data	Data Valid
x =	System Controller Active State	Controller Active State	Talker Active State	Listener Active State	Serial Poll Mode State	Attention	Not Data Accepted	Not Ready For Data

A logical 1 indicates a true condition of the indicated line and a 0 indicates a false condition.

For example, this program segment checks bit 5 of the byte in Y to see if a service request has been sent.

When a service request is detected, the program branches to a subroutine that performs a serial poll to determine the required action.

```

0095      •
0096      •
0097  STAT      5
0099  X≠Y
0100  3
0101  2
0102  AND      5
0104  3
0105  2
0106  IF X=Y
0107  GOSUB  A
0109  GOTO    0097
0111      •
          •
          •
    
```

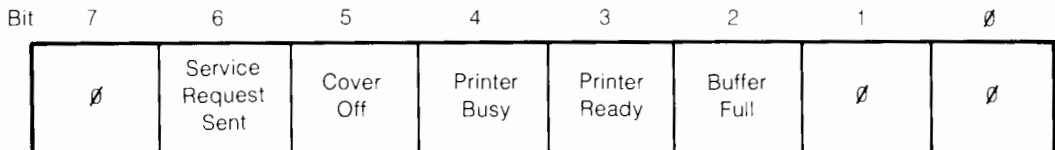
Responding to Service Requests

When the calculator is the 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 maintain proper system operations. Once a service request is recognized, serial polling is used to identify the source of a service request and reveal the cause.

Serial Polling – 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-pollled, 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, the output has stabilized at a new level or the device has requested manual service. All of the bits except bit 6 are unique to each device. Bit 6 always indicates whether or not a service request was sent by the device. Once the calculator has serviced each device that has been requesting service, the SRQ line is cleared (assuming no new requests are received).

For example, this is the status byte sent by an HP 9871A Option 001 Printer:



- Bits 0, 1 and 7 are always logical 0.
- Bit 2 is set to 1 when the bufer has room for 16 characters or less.
- Bits 3 & 4 are 1 when the printer is not ready to accept data.
- Bit 5 is 1 when the printer's cover is off.
- Bit 6 is 1 when the printer has sent a Require Service message.

The SERIAL POLL Instruction




Address Code → X

The SERIAL POLL instruction (SERPL) polls devices on the bus, one at a time. When polled, a device sends a single byte of information that indicates its status. This status byte is entered into X as a decimal equivalent and a stack-up operation is performed.

For example, the subroutine below performs a serial poll on each of five devices that have consecutive address codes of 01 thru 05.

The for-next loop A → F is used to generate the address codes 01 thru 05 of the device being polled. Each device's status byte is checked with an AND instruction for a value of 64 (bit 6 set to a logical 1) to see if that device sent a service request. If a service request is found the status byte is stored in a register corresponding to the address code (01 thru 05) of the device. This byte is typically used later in the program to determine the cause of the service request.



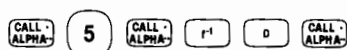
```

0258 LBL
---- A
0260 1
0261 STO A
0262 5
0263 STO F
0264 RCL A
0265 SERPL 5
0267 6
0268 4
0269 AND 5
0271 6
0272 4
0273 IF X=Y
0274 GOTO 0278
0276 NEXT A
0277 RETURN
0278 ROLL↓
0279 STO I A
0281 GOTO 0276

```

Sending Service Requests

When the calculator is not the active controller (either it has passed control to another device or the system controller jumper on the interface is moved), it can send a Require Service message:



After sending the require service message, the calculator (when polled) must execute a WRITE BYTE instruction to send the calculator's Status Byte message from X. When a value is output in response to a serial poll, the Require Service message is automatically cleared.

For example, this program segment sends a Require Service message to the bus. The next steps use the STATUS instruction to check for a talker active state in a serial poll mode (bits 5 & 3 of the X register status byte).

```

0085      ●
0086      ●
0087  CMD      5
0089  @
0090  D
0091  ENDα
0092  STAT     5
0094  4
0095  0
0096  AND      5
0098  4
0099  0
0100  IF X=Y
0101  GOTO     0105
0103  GOTO     0092
0105  CLEAR
0106  7
0107  2
0108  WBYTE   5
0110      ●

```

When this condition is found, the value 72 is sent from X with a WRITE BYTE instruction (bit 6 is a 1 to indicate that the calculator sent the Require Service Message and bit 3 is a 1 due to the particular program designs).

The Require Service message can be cancelled by executing the following key sequence:



Example Program

HP-IB systems often utilize one calculator as the system controller and one or more calculators that are not system controllers. The following program shows a typical sequence of bus messages that a calculator, which is not system controller, can use to request and receive control from a system controller. It then returns control when the system controller requests it.

The interface status is checked by executing a STAT instruction and using an AND instruction to check for a value of 64 (bit 6 of the status byte in X) to determine if the calculator is active controller (steps 0000 thru 0009).

If the calculator is active controller, the program branches to step 0047. If not, the CMD instruction sends the Require Service message (steps 0010 thru 0016).

The interface status is checked again until the calculator is addressed as a talker in the serial poll mode (bits 5 and 3 of the status byte in X total 40 – steps 0017 thru 0025). When this condition occurs, the value 72 is sent with a WBYTE instruction as the Status Byte message in response to the serial poll. The 72 represents bit 6 (which specifies that this calculator sent the service request) and bit 3 (which was chosen for program use to request control – steps 0030 thru 0032).

The interface status is checked until the value 96 is returned. The value 96 specifies that the active controller state (bit 6) and the talker active state (bit 5) are both set to a logical 1 (steps 0034 thru 0042).

The calculator now prints "CONTROLLER" (steps 0047 thru 0059). At this point, most system applications would branch to a subroutine to perform some programmed task and return when finished.

The interface status is now checked to locate a service request condition (bit 5 of the status byte in Y) (steps 0060 thru 0069). When this condition is found, the calculator uses the SERPL instruction to answer the service request by polling the system controller (address code 21) and checks the byte value returned for bit 6 (specifying that the system controller sent the service request) and for bit 3 (chosen for this program to request control – steps 0074 thru 0084).

When this condition is true (a value of 72 is returned) the CMD instruction is used to send the Pass Control Message to the calculator with the address code of 21 (steps 0089 thru 0095).

The calculator now prints "NOT CONTROLLER" and repeats the program.

```

0000 CLEAR
0001 STAT 5
0003 6
0004 4
0005 AND 5
0007 6
0008 4
0009 IF X=Y
0010 GOTO 0047
0012 CMD 5
0014 @
0015 D
0016 END@
0017 STAT 5
0019 4
0020 0
0021 AND 5
0023 4
0024 0
0025 IF X=Y
0026 GOTO 0030
0028 GOTO 0017
0030 7
0031 2
0032 WBYTE 5
0034 STAT 5
0036 9
0037 6
0038 AND 5
0040 9

```

0041	6			0080	AND	5
0042	IF X=Y			0082	7	
0043	GOTO	0047		0083	2	
0045	GOTO	0034		0084	IF X=Y	
0047	PRNT α			0085	GOTO	0089
0049	C			0087	GOTO	0047
0050	0			0089	2	
0051	N			0090	1	
0052	T			0091	CMD	5
0053	R			0093	@	
0054	0			0094	I	
0055	L			0095	END α	
0056	E			0096	PRNT α	
0057	R			0098	N	
0058	LINE			0099	0	
0059	END α			0100	T	
0060	STAT	5		0101		
0062	X \leftrightarrow Y			0102	C	
0063	3			0103	0	
0064	2			0104	N	
0065	AND	5		0105	T	
0067	3			0106	R	
0068	2			0107	0	
0069	IF X=Y			0108	L	
0070	GOTO	0074		0109	E	
0072	GOTO	0050		0110	R	
0074	2			0111	LINE	
0075	1			0112	END α	
0076	SERPL	5		0113	GOTO	0000
0078	7			0115	END	
0079	2					

3 Service

This section gives the procedure used to change the bus address and system controller jumpers. The section also contains a brief description of interface operation, complete circuit diagrams and a list of replaceable parts.

If you have difficulty repairing the interface or if you would rather have HP repair it, contact the nearest sales and service office; office locations are listed in the Appendix.

Address Code and System Controller Jumpers

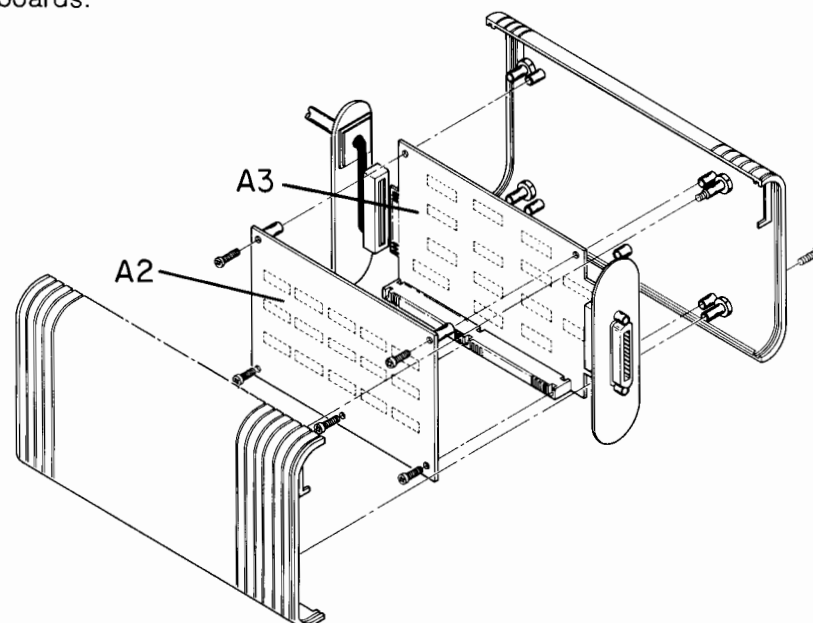
The interface is shipped from the factory with the address jumpers specifying an address code of 21 (ASCII talk address of U and listen address of 5). The interface is also shipped with the system controller jumper specifying the calculator as the system controller.

The following procedures can be used to change either or both of these settings.



Disassembly Procedure

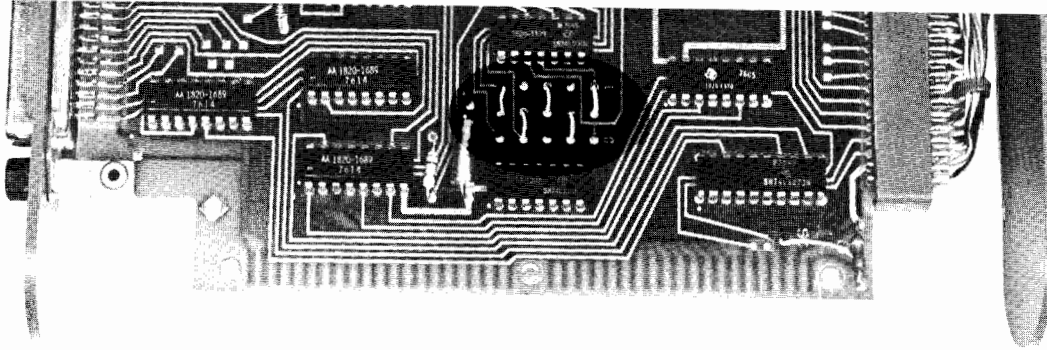
1. Switch the calculator off and disconnect the interface from the bus cables.
2. Remove the four rubber pads from the larger (A2 and A3) case. Remove the four screws that are located under the pads and remove the plastic cover.
3. Remove the five screws from the A2 circuit board (see the next figure) and separate the boards.



Disassembled A2 Board, A3 Board and Case

The Address Jumpers

1. Locate the address jumpers (J1 thru J5) on the A3 board (98135-66502) as shown below. The left-most jumper in the figure below is J1 (corresponding to bit 1 of the address code) and the right-most jumper is J5 (corresponding to bit 5). Refer to the A3 Component Locator on page 52, if necessary, for further clarification of the jumper designations.



Address Jumpers on the A3 Board

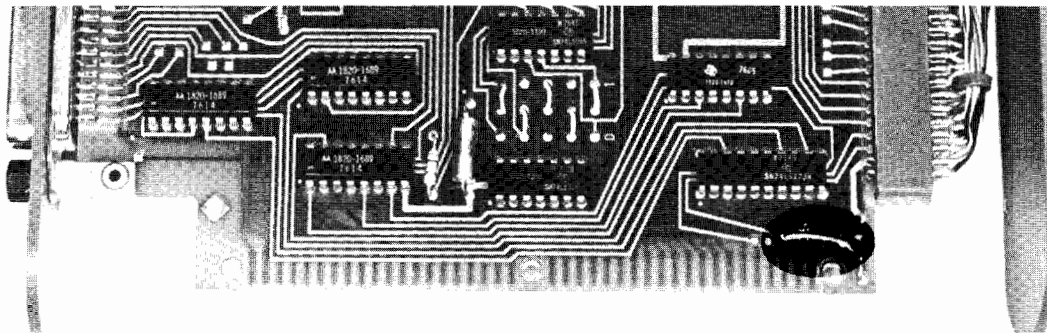
2. One end of each jumper is soldered in one of the center holes and the other end is soldered in either a "1" or a "0" hole, according to the bit being specified. Refer to the table on page 59 to find the five bit positions that will specify the new address code.

Using a low temperature soldering iron and long-nose pliers, resolder the jumpers to correspond to the jumper designations from the table.

3. If you want to stop the calculator from being system controller, follow the procedure below under "The System Controller Jumper". If the calculator is to remain the system controller, reassemble the A2 and A3 boards in their case (as shown in the figure on page 41) and reconnect the interface to the calculator and the bus cables.

The System Controller Jumper

1. Locate the system controller jumper (shown below) on the A3 board (98135-66502). Using a low temperature soldering iron and long-nose pliers, remove the jumper from the board. With this jumper removed, the calculator is not the system controller.



System Controller Jumper on the A3 Board

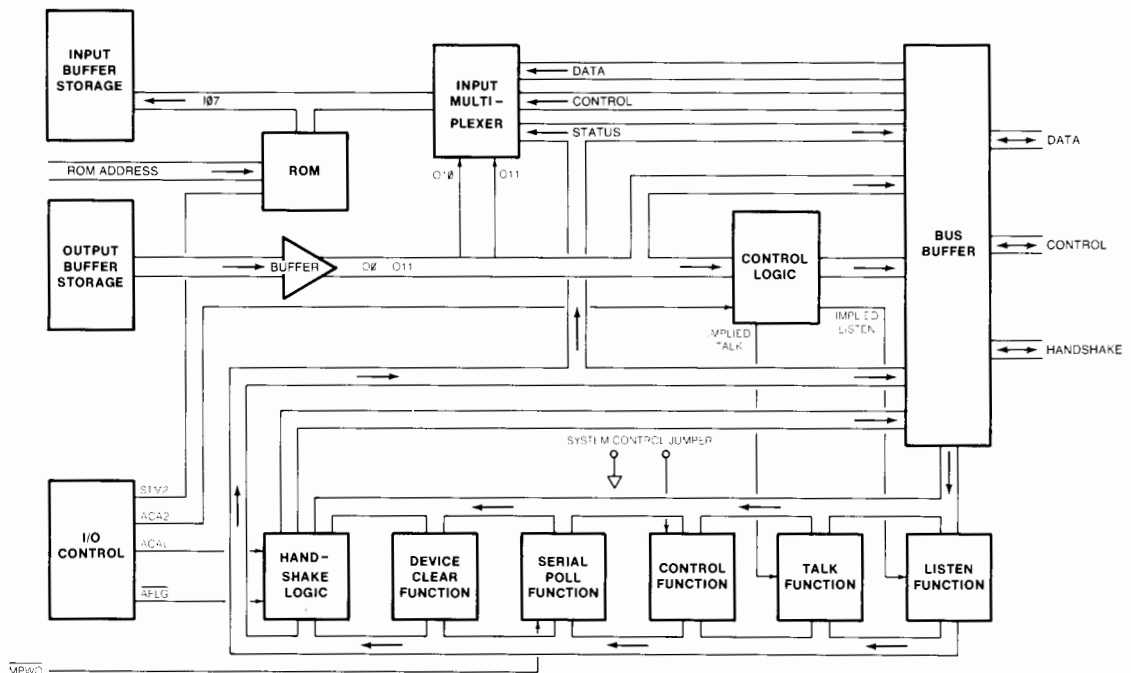
2. Reassemble the A2 and A3 boards in their case (as shown in the figure on page 41) and reconnect the interface to the calculator and the bus cables.

Note

When the calculator is not system controller, the automatic addressing feature must be disabled for any instructions that utilize it. This requires that 0 be entered in place of the address code parameter in each instruction.

Theory of Operation

A block diagram of the interface is shown below. The interface consists of three printed circuit assemblies; the A1 board (98135-66503), the A2 board (98135-66501) and the A3 board (98135-66502).



Interface Block Diagram

The A1 board contains an instruction ROM and data buffers. The A2 board contains the handshake logic and the circuits for the various bus functions. The A3 boards contain the bus input/output buffers, the control logic and the input multiplexers. Note that the data being input, the control data and the bus status information is transferred to the instruction ROM on the eight input lines (DI1 thru DI8).

Note

Refer to the Circuit Diagrams on pages 49 thru 53 when reading the following sections

ROM Enable

A1U8 is a BCD-to-decimal decoder which, together with its associated circuitry, enables the ROM as each I/O instruction is executed. The signals on the ROM address bus are strobed continuously and have no steady state.

ROM Power-Up

A1Q1 controls the $-12V$ supply for the ROM (A1U7). A1Q1 is switched off except when the ROM is enabled by an I/O instruction. A1C1 enables A1Q1 to switch-on more rapidly.

Control Logic

A3U14, an 8-bit latch, enables the various bus control functions (EOI, REN, SRQ, IFC, and ATN). It also enables the calculator's implied listener or talker address when the automatic addressing is used.

Input Multiplexer

A3U3, A3U7 and A3U11 are multiplexers and are controlled by the sense of the signals on O10 and O11. The multiplexers determine which information (Data, Status or Control) is input into the calculator and the instruction ROM, A1U7.

Handshake Logic

The handshake logic uses the ACAL signal to drive the DAV line true on the bus when the calculator is a talker. The AFLG line is controlled by the NDAC line to signal the calculator that the word it sent to the bus has been accepted. The signal on COT enables ATN if the data being sent is a control message and clears ATN if a data message is being sent.

When the calculator is a listener and ATN is not true, ACAL controls the NRFD line and DAV controls AFLG.

HP-IB Functions

The logic for the various bus functions is located on the A2 board. These circuits interpret the following bus control messages and transfer parameters when the ATN line is true.

The Listener Function

The listener logic specifies the interface as a listener when it receives either its listen address or an implied listen signal (IMPL). This logic specifies an unlisten condition when it receives an unlisten command, the interface talker address, or an Abort message (IFC). The interface is also set to unlisten when the calculator is first switched on.

The Talker Function

The talker logic specifies the interface as the talker when it receives either its talk address or an implied talk signal (IMPT). The talker logic specifies an untalker when it receives either another talker address, its listener address or an Abort message (IFC). The untalker condition is set when the calculator is first switched on.

The Controller Function

The calculator is specified as active controller by the control logic if the system controller jumper is connected and either an Abort message (IFC) is sent or the calculator is first switched on. The calculator also becomes the controller when a Pass Control message specifying its talker address is received.

The calculator is specified as not-controller if the system controller jumper has been removed and either an Abort message (IFC) is received or the calculator is first switched on. The calculator is also specified as not-controller when a Pass Control message is sent with another device's talk address specified.

The Serial Poll Function

The serial poll circuitry enables the serial poll mode when the serial poll instruction is executed.

The Device Clear Function

The device clear logic sets the calculator to its power-on state when a Clear message (DCL or SDC) is received.

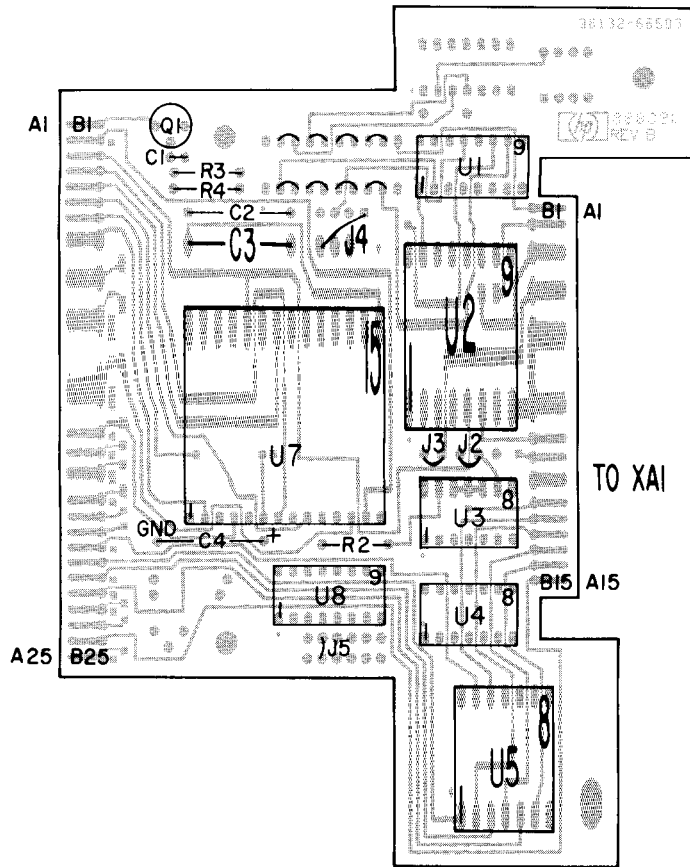
DESIGNATOR	PART NO.		
A1	98135-69503	1	HP-IB ROM P.C. Assembly
A1C1	0160-4387	1	Cap: 47pf, 200V
A1C2	0180-0228	1	Cap: 22 μ f, 15V
A1C3,C4	0180-1704	2	Cap: 47 μ f, 6V
A1Q1	1853-0419	1	Transistor, Si, PNP
A1R2,R4	0683-1025	2	Res: 1K Ω , 5%, 1/4W
A1R3	0683-6825	1	Res: 6.8k Ω , 5%, 1/4W
A1U1,U2	1820-1492	2	IC: SN74LS368N
A1U4,U5	1820-1199	2	IC: 74LS04N
A1U7	1818-2640	1	IC: 16k ROM
A1U8	1820-1418	1	IC: 74LS42
A2	98135-66501	1	Control P.C. Assembly
A2C1,C2	0180-1735	2	Cap: 22 μ f, 35V
A2C3,C5,C6,C7	0160-3847	4	Cap: .01 μ f, 25V
A2C4	0180-1704	1	Cap: 47 μ f, 6V
A2R1	0683-1045	1	Res: 100k Ω , 5%, 1/4W
A2R2,R3,R4	0683-4725	3	Res: 4.7k Ω , 5%, 1/4W
A2R5	0883-1515	1	Res: 150 Ω , 5%, 1/4W
A2Q1,Q2	1854-0071	2	Transistor: Si, NPN
A2U1	1820-1197	1	IC: 74LS00N
A2U2,U8	1820-1206	2	IC: SN74LS27
A2U3	1820-1416	1	IC: 74LS14N
A2U4	1820-1199	1	IC: SN74LS04N
A2U5,U12	1820-1210	2	IC: SN74LS51
A2U6	1820-1440	1	IC: SN74LS279
A2U7	1820-1425	1	IC: SN74LS132
A2U9,U10	1820-1144	2	IC: SN74LS02N
A2U11	1820-1201	1	IC: SN74LS08N
A2U13	1820-1418	1	IC: 74LS42
A2U14	1820-1204	1	IC: SN74LS20N
A2U15	1820-1112	1	IC: SN74LS74N
A3	98135-66502	1	I/O P.C. Assembly
A3C1	0160-3456	1	Cap: 1000pf, 10%, 1000V
A3C2,C3,C4	0160-3847	5	Cap: .01 μ f, 25V
C7,C9			
A3C5	0160-0576	1	Cap: .01 μ f, 50V
A3C6	0160-3874	1	Cap: 10pf, 200V
A3C8	0180-1704	1	Cap: 47 μ f, 6V
A3Q1,Q2	1854-0071	2	Transistor: Si, NPN
A3R1	0683-1025	1	Res: 1k Ω , 5%, 1/4W
A3R2	0683-2035	1	Res: 20k Ω , 5%, 1/4W
A3R3	0683-4725	1	Res: 4.7k Ω , 5%, 1/4W
A3U1	1820-1425	1	IC: SN74LS132
A3U2	1820-1206	1	IC: SN74LS27
A3U3,U7,U11	1820-1470	3	IC: SN74LS157
A3U4,U9,U10			
U12	1820-1689	4	IC: MC3446P
A3U5,U8	1820-1199	2	IC: 74LS04N
A3U6,U13	1820-1207	2	IC: SN74LS30
A3U14	1820-1730	1	IC: 74LS273
A3J1	1251-3283	1	Connector
W1	98133-61601	1	Cable Assembly
XA1,P2	1251-4327	2	Connector: (2X15)
E1	1251-4010	1	Connector
	5040-7781	1	A1 Cover, Top
	5040-7782	1	A1 Cover, Bottom
	5040-8035	1	A2 and A3 Cover, Top
	5040-8034	1	A2 and A3 Cover, Bottom
	5040-8025	1	A2 and A3 Cover, End
	98135-90000	1	Operating and Service Manual
	98135-90010	1	Reference Card
	7120-5355	1	Key Overlay

XA1 Wire Connections

Pin No.	Function	Wire Color
A1	ALTCH	Wht/Blu
A2	DI4	Wht/Vio
A3	DI5	Wht/Gra
A4	+5V	Wht/Blk/Brn
A5	GND	Blk
A6	GND	Shield
A7	AFLG	Wht/Blk/Red
A8		
A9		
A10	ACAL	Wht/Blk/Yel
A11	O10	Wht/Blk/Grn
A12	O9	Wht/Blk/Blu
A13	O0	Wht/Blk/Vio
A14	O1	Wht/Blk/Gra
A15	MPWO	Wht/Brn/Red
B1	DI3	Brn
B2	DI2	Red
B3	DI1	Orn
B4	DIO0	Yel
B5	DI6	Grn
B6	DI7	Blu
B7	O4	Vio
B8	O8	Gra
B9	O6	Wht
B10	O11	Wht/Blk
B11	O7	Wht/Brn
B12	O5	Wht/Red
B13	O3	Wht/Orn
B14	O2	Wht/Yel
B15	ACA2	Wht/Grn

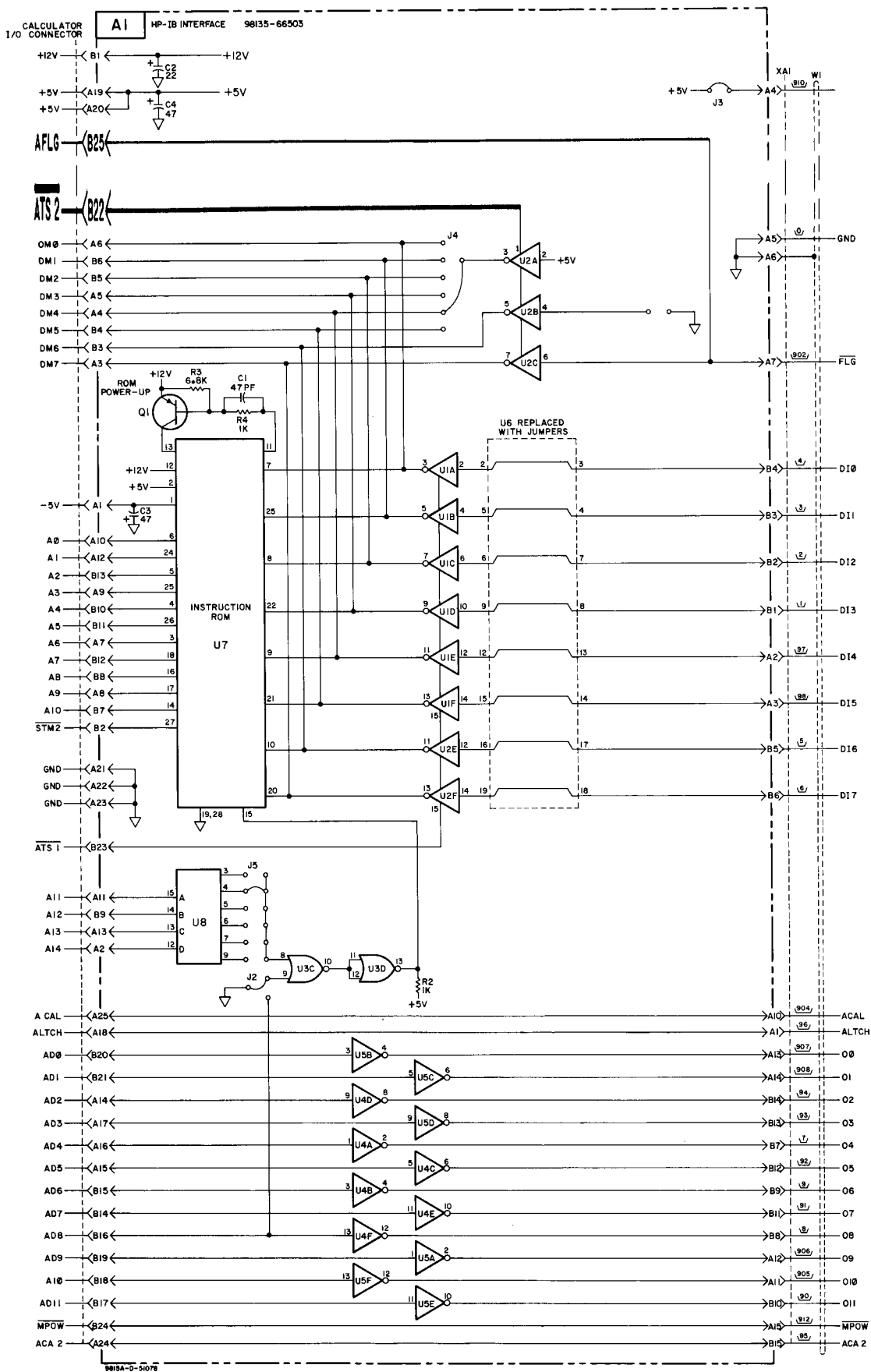
P2 Wire Connections

Pin No.	Function	Wire Color
A1	DRAIN	Shield
A2	O4	Vio
A3	O5	Wht/Red
A4	O6	Wht
A5	O7	Wht/Brn
A6	O10	Wht/Blk/Grn
A7	O11	Wht/Blk
A8	I4	Wht/Vio
A9	I5	Wht/Gra
A10	I6	Grn
A11	I7	Blu
A12	ACAL	Wht/Blk/Yel
A13	AFLG	Wht/Blk/Red
A14	+5V	Wht/Blk/Brn
A15	GND	Blk
B1	O0	Wht/Blk/Vio
B2	O1	Wht/Blk/Gra
B3	O2	Wht/Yel
B4	O3	Wht/Orn
B5	O8	Gra
B6	O9	Wht/Blk/Blu
B7	I0	Yel
B8	I1	Orn
B9	I2	Red
B10	I3	Brn
B11	ACA2	Wht/Grn
B12		Wht/Blu
B13	MPWO	Wht/Brn/Red
B14	+5V	
B15	GND	

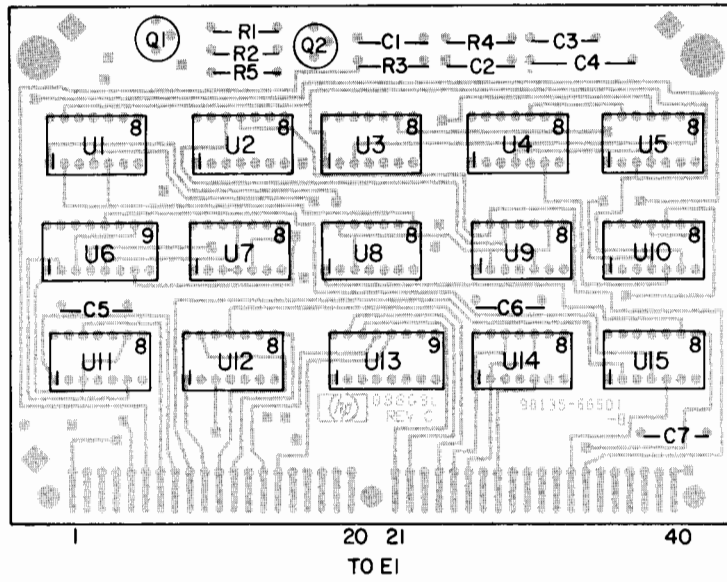


A1 ROM Board

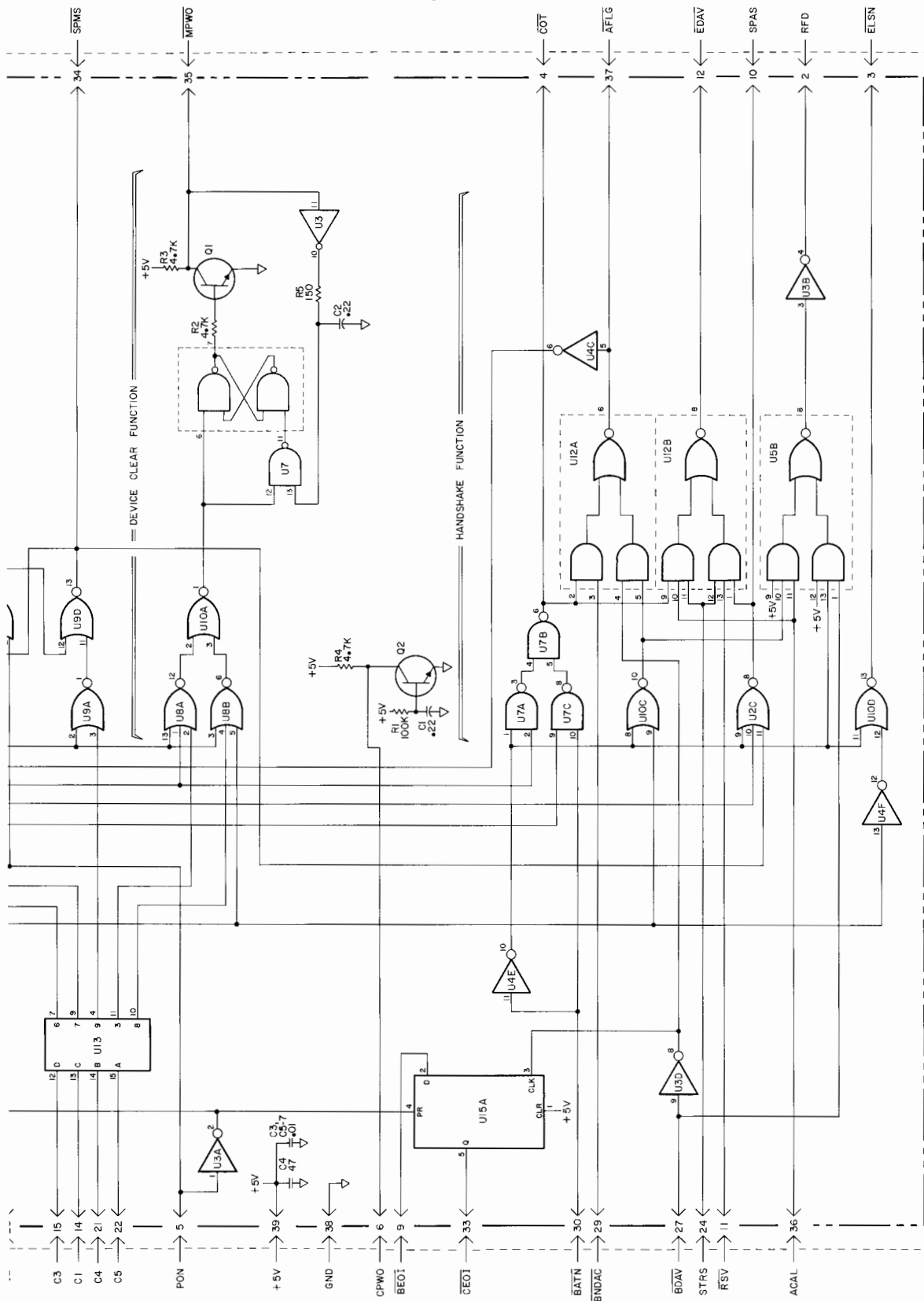
A1 Schematic



A2 Component Locator



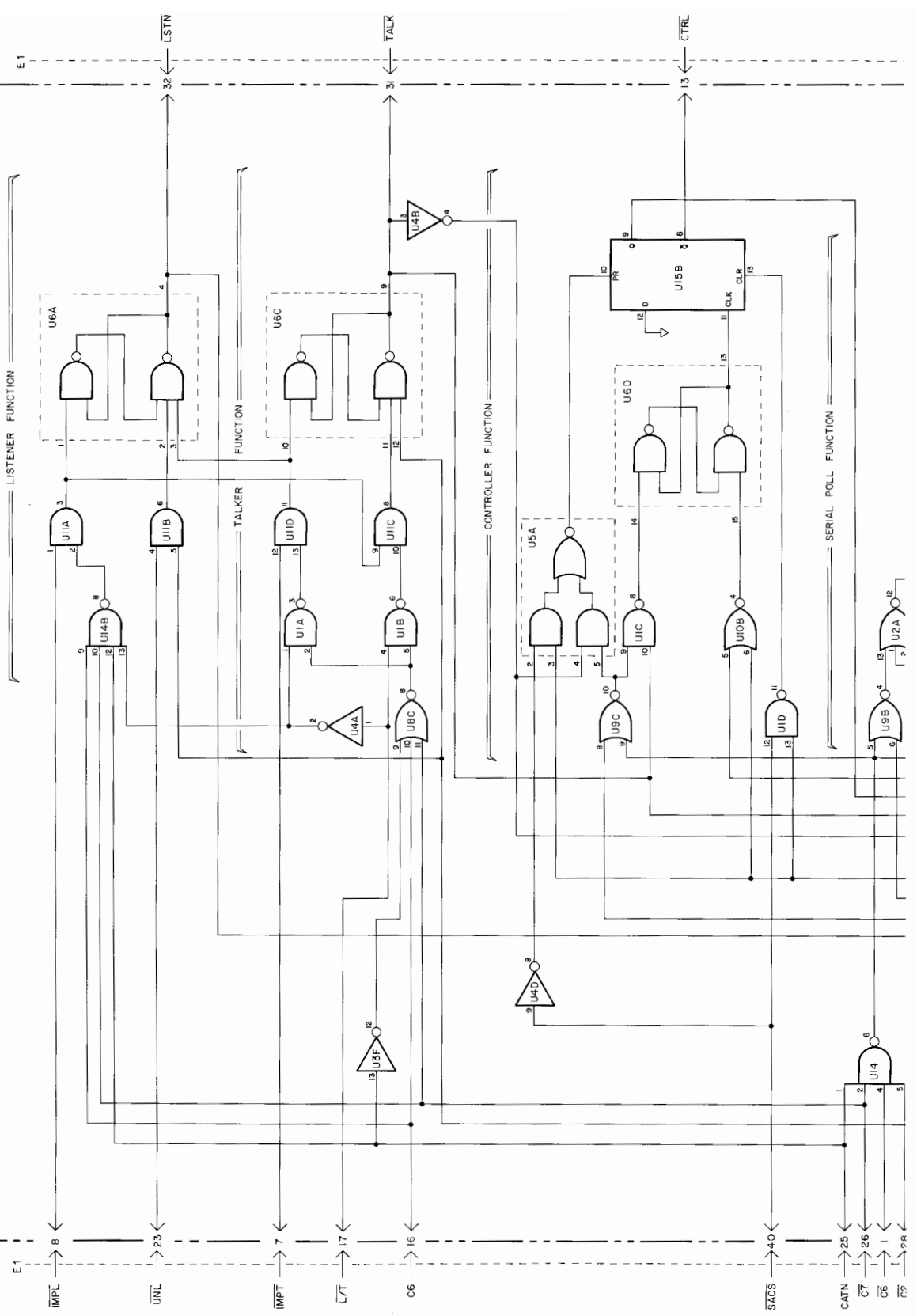
A2 Control Board



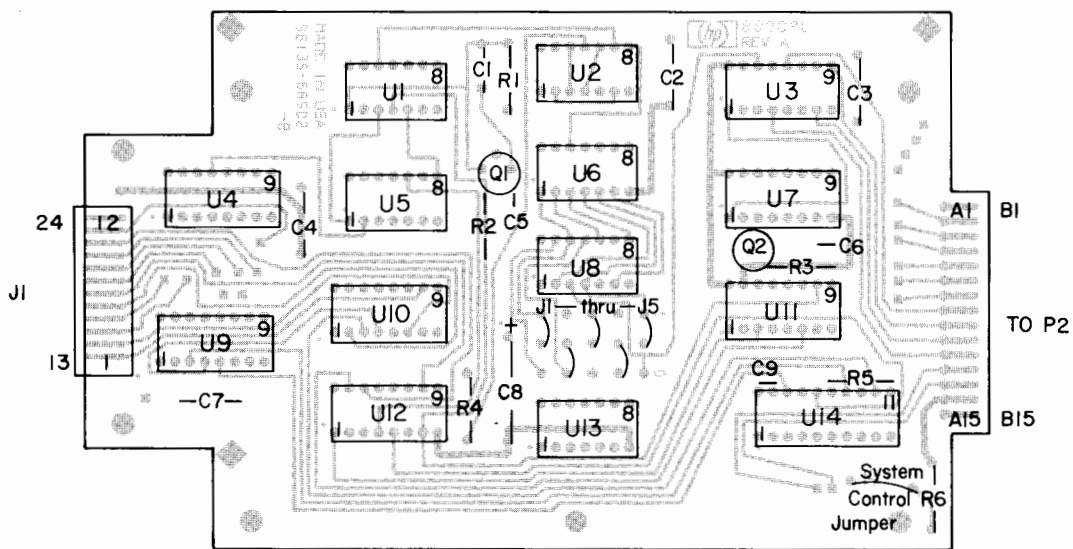
A2 Schematic

91035-1-51079

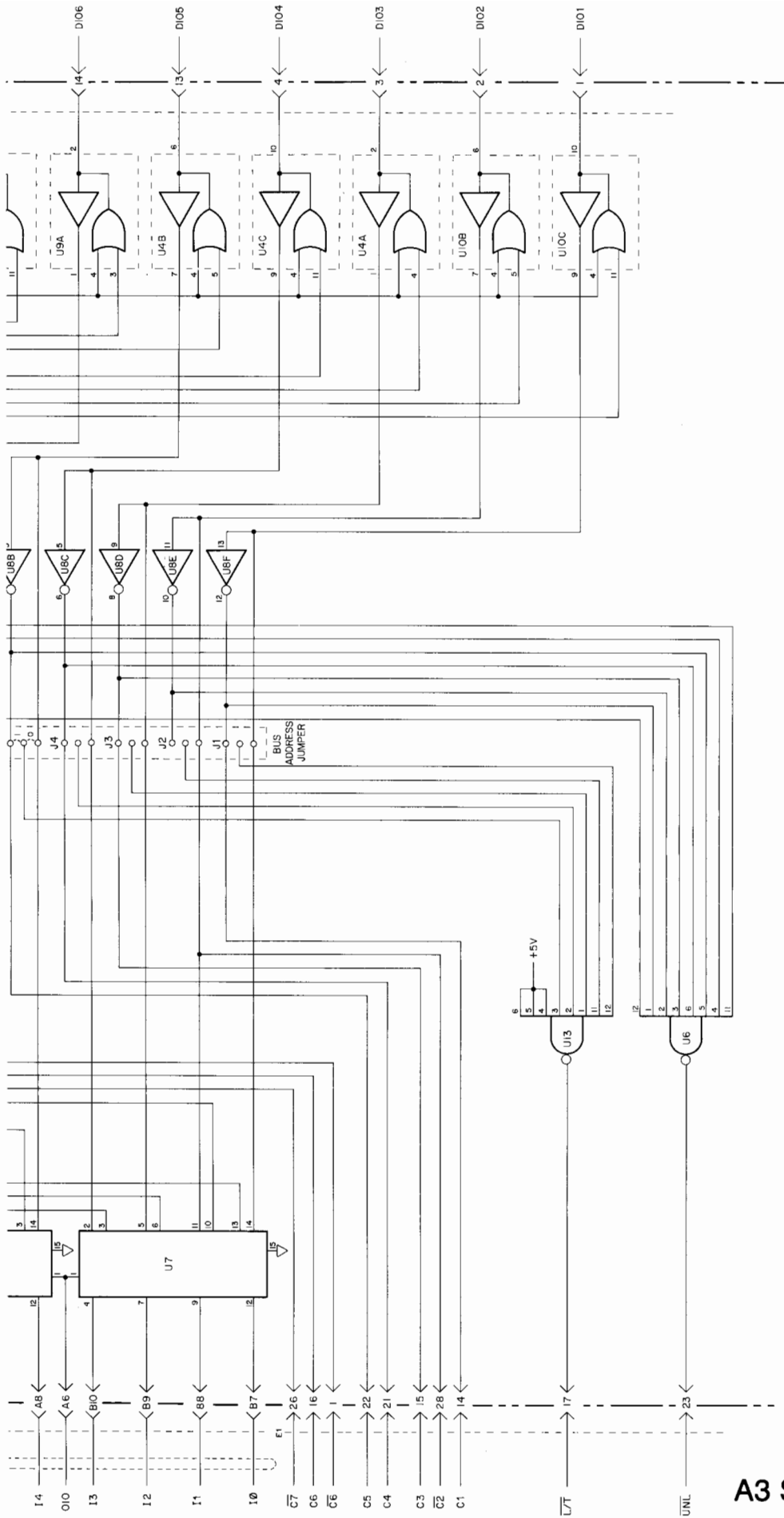
A2 HP-1B INTERFACE 98135-66501



A3 Component Locator

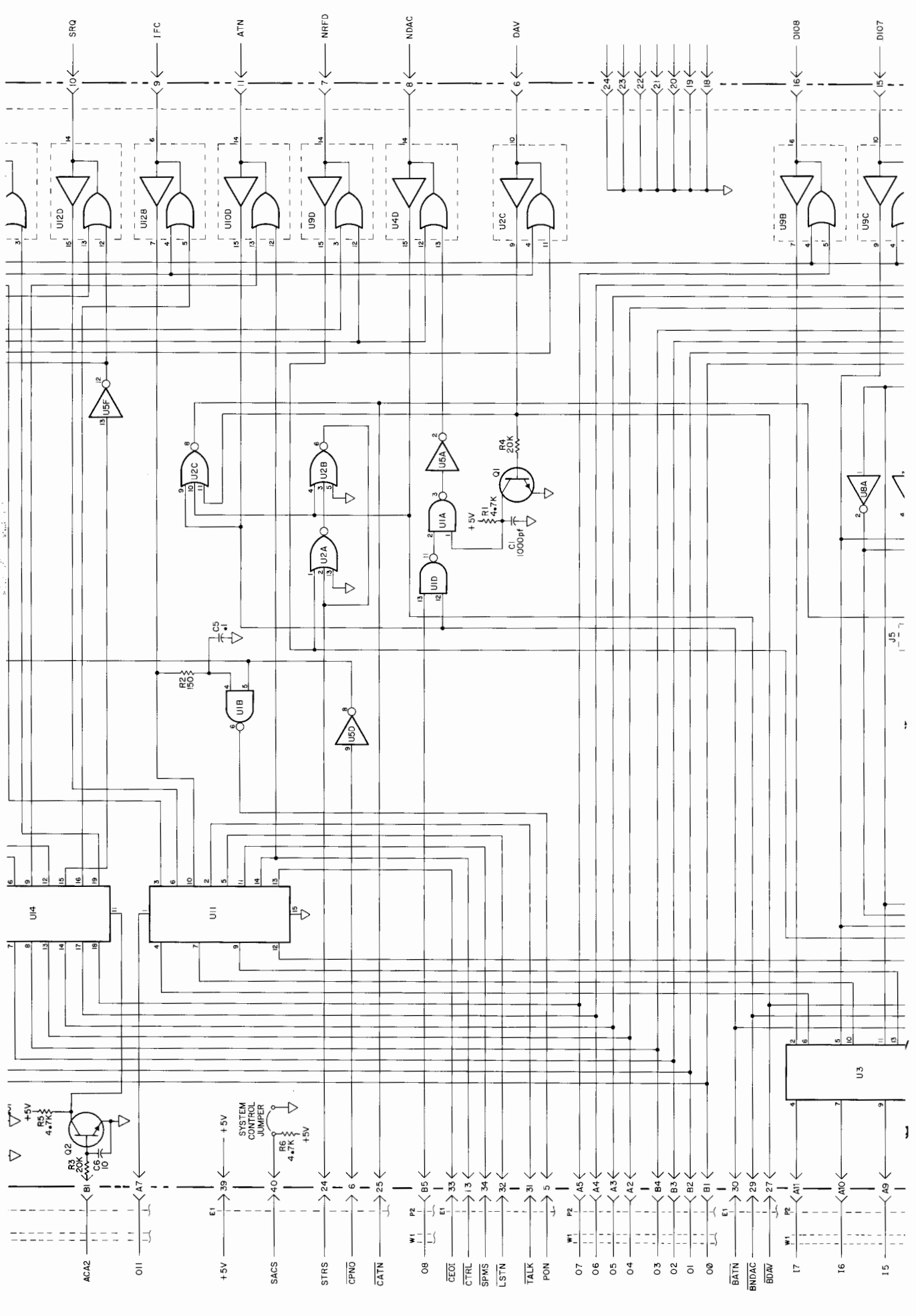


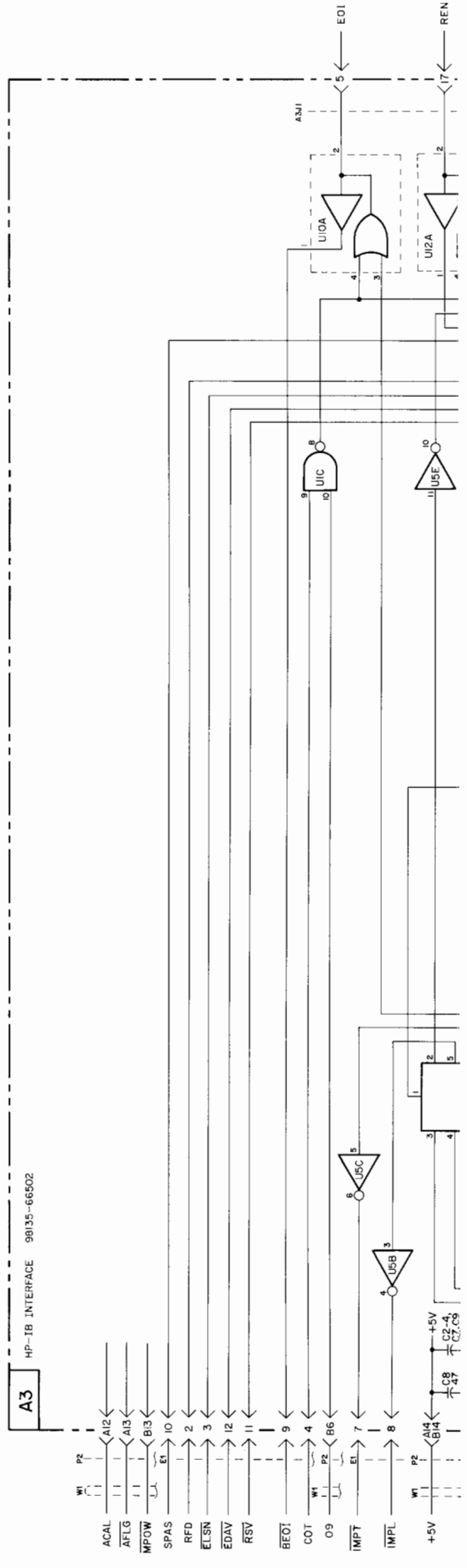
A3 I/O Board



A3 Schematic

98135-1-51080





Appendix

Binary Coding and Conversions

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

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

Binary-Decimal Conversions

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

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

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

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

Octal-Binary Conversions

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

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

For example, to convert binary 10110100011001 to octal:

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

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

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

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

































ASCII

Binary is often used as a code to represent not only numbers, but also alphanumeric characters such as "A" or "," or "?" or "x" or "2". One of the most common binary codes used is ASCII¹. ASCII is an eight-bit code, containing seven data bits and one parity bit. The BCD interface uses 7 bit ASCII without a parity bit. For example:

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

A complete list of ASCII characters and their equivalent binary and decimal representations is on page 56 and 57.

¹American Standard Code for Information Interchange.

ASCII Char.	EQUIVALENT FORMS			CMD Instruction Keys ¹	ASCII Char.	EQUIVALENT FORMS			CMD Instruction Keys ¹
	Binary	Octal	Dec			Binary	Octal	Dec	
NULL	00000000	000	0	S	' '	00100000	040	32	
SOH	00000001	001	1	S	!	00100001	041	33	S 
STX	00000010	002	2	S	"	00100010	042	34	S 
ETX	00000011	003	3	S	#	00100011	043	35	
EOT	00000100	004	4	S	\$	00100100	044	36	
ENQ	00000101	005	5	S	%	00100101	045	37	
ACK	00000110	006	6	S	&	00100110	046	38	S 
BELL	00000111	007	7	S	'	00100111	047	39	
BS	00001000	010	8	S	(00101000	050	40	
H _{TAB}	00001001	011	9	S)	00101001	051	41	
LF	00001010	012	10	S	*	00101010	052	42	
V _{TAB}	00001011	013	11	S	+	00101011	053	43	
FF	00001100	014	12	S	,	00101100	054	44	
CR	00001101	015	13	S	-	00101101	055	45	
SO	00001110	016	14	S	.	00101110	056	46	
SI	00001111	017	15	S	/	00101111	057	47	
DLE	00010000	020	16	S	∅	00110000	060	48	
DC ₁	00010001	021	17	S	1	00110001	061	49	
DC ₂	00010010	022	18	S	2	00110010	062	50	
DC ₃	00010011	023	19	S	3	00110011	063	51	
DC ₄	00010100	024	20	S	4	00110100	064	52	
NAK	00010101	025	21	S	5	00110101	065	53	
SYNC	00010110	026	22	S	6	00110110	066	54	
ETB	00010111	027	23	S	7	00110111	067	55	
CAN	00011000	030	24	S	8	00111000	070	56	
EM	00011001	031	25	S	9	00111001	071	57	
SUB	00011010	032	26	S	:	00111010	072	58	
ESC	00011011	033	27	S	;	00111011	073	59	S 
FS	00011100	034	28	S	<	00111100	074	60	
GS	00011101	035	29	S	=	00111101	075	61	
RS	00011110	036	30	S	>	00111110	076	62	
US	00011111	037	31	S	?	00111111	077	63	

ASCII Char.	EQUIVALENT FORMS			CMD Instruction Keys ¹	ASCII Char.	EQUIVALENT FORMS			CMD Instruction Keys ¹
	Binary	Octal	Dec			Binary	Octal	Dec	
@	01000000	100	64			01100000	140	96	T
A	01000001	101	65		a	01100001	141	97	T
B	01000010	102	66		b	01100010	142	98	T
C	01000011	103	67		c	01100011	143	99	T
D	01000100	104	68		d	01100100	144	100	T
E	01000101	105	69		e	01100101	145	101	T
F	01000110	106	70		f	01100110	146	102	T
G	01000111	107	71		g	01100111	147	103	T
H	01001000	110	72		h	01101000	150	104	T
I	01001001	111	73		i	01101001	151	105	T
J	01001010	112	74		j	01101010	152	106	T
K	01001011	113	75		k	01101011	153	107	T
L	01001100	114	76		l	01101100	154	108	T
M	01001101	115	77		m	01101101	155	109	T
N	01001110	116	78		n	01101110	156	110	T
O	01001111	117	79		o	01101111	157	111	T
P	01010000	120	80		p	01110000	160	112	T
Q	01010001	121	81		q	01110001	161	113	T
R	01010010	122	82		r	01110010	162	114	T
S	01010011	123	83		s	01110011	163	115	T
T	01010100	124	84		t	01110100	164	116	T
U	01010101	125	85		u	01110101	165	117	T
V	01010110	126	86		v	01110110	166	118	T
W	01010111	127	87		w	01110111	167	119	T
X	01011000	130	88		x	01111000	170	120	T
Y	01011001	131	89		y	01111001	171	121	T
Z	01011010	132	90		z	01111010	172	122	T
[01011011	133	91	S	{	01111011	173	123	T
\	01011100	134	92	S		01111100	174	124	T
]	01011101	135	93	S	}	01111101	175	125	T
^	01011110	136	94	S	~	01111110	176	126	T
_	01011111	137	95	S	DEL	01111111	177	127	T

¹The keys with a T or an S in front of them indicate either the T or S shifted state of the keyboard. Pressing sets the S shifted state and pressing sets the T shifted state. To reset the keyboard to the unshifted state, press the key that set the state again or terminate the CMD mode.

²The @ and space characters can only be sent in data messages (see "The Data Message" in Section 3).

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

Instrument	HP-IB Address ¹	
	Talker	Listener
98135A Interface	U	5
3490A Multimeter	V	6
9871A (Opt. 001) Printer	A	!
59309A Digital Clock	P	0

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

Now, using the ASCII table on page 56, convert the five least-significant bits of each character's

binary form to a decimal value:

Instrument	Address	
	Character	5-Bit Value
98135A Interface	U	21
	5	21
3490A Multimeter	V	22
	6	22
9871A Printer	A	1
	!	1
59309A Clock	P	16
	O	16

Notice that the 5 bit value for both the talker and listener characters is the same number.

These numbers are used as the address code parameters for interface operations. Each address code value must always contain two digits; if the 5 bit value is a one-digit number (e.g., 9), a leading zero must be used (e.g., 09).

A table of the available ASCII character addresses and their equivalent decimal codes is provided next.

Table 5. Available Bus Addresses and Codes

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

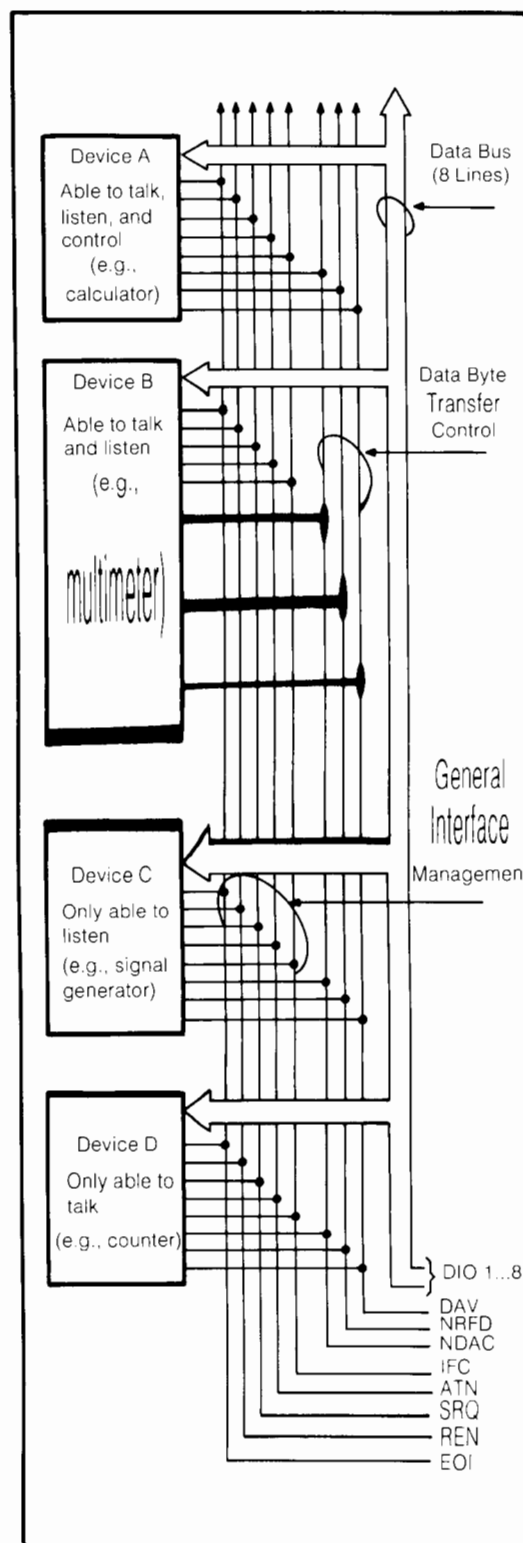
The HP Interface Bus is Hewlett-Packard's implementation of the IEEE standard 488-1975. You can order a copy of this standard from the IEEE Standards Office; 345 East 47th Street, New York, N.Y., 10017. A brief technical description of the HP-IB is given here.

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.

Eight of the lines (DI01-DI08) 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 (DAV, NRD, NDAC). The other five lines are for control of bus activity.

Devices connected to the bus may be talkers, listeners, or controllers. The controller dictates the role of the other devices by setting the ATN (attention) line true and sending talk or listen addresses on the data lines (DI01-DI08). 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.



HP-IB Signal 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. 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 control 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 other control lines are IFC, REN, SRQ, EOI.



IFC (interface clear) places the interface system in a known quiescent state.

REN (remote enable) is used with other coded messages to select either local or remote control of each device.

Any active device can set the SRQ (service request) line true. 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.

EOI (end or identify) 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. It uses a group of commands to direct the other instruments on the bus in carrying out their functions of talking and listening.

The commands serve several different purposes.

1. Addresses, or talk and listen commands, select the instruments that will transmit and accept data. They are all multiline messages.
2. Universal commands cause every instrument equipped to do so to perform a specific interface operation. They include multiline messages and three uniline commands, interface clear (IFC), remote enable (REN), and attention (ATN).
3. Addressed commands are similar to universal commands, except that they affect only those devices that are addressed and are all multiline commands. An instrument responds to an addressed command, however, only after an address has already told it to be a talker or listener.
4. Secondary commands are multiline 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 below, five data bits are available for addresses, so a total of 31 addresses is 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
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
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
X	1	1	1	1	1	1	1	Ignored

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

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

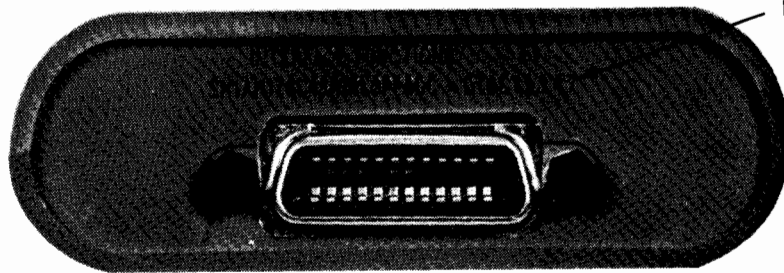
Since interface functions are the physical agency through which bus messages are implemented, each device must implement one or more functions to enable it to send or receive a given bus message.

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 instruments, such as the 98135A Interface, list the functions implemented directly on the device, as shown in the next photo.

Functions Used by the Bus Messages

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 *.



Functions Implemented

98135 Interface Connector

Bus Functions Available with the 98135A Interface

Function		Implementation
SH1	(Source Handshake)	Complete Capability.
AH1	(Acceptor Handshake)	Complete Capability.
T6	(Talker)	Basic Talker. Serial Poll. Unaddress if my listen address (MLA).
L4	(Listener)	Basic Listener. Unaddress if my talk address (MTA).
SR1	(Service Request)	Complete Capability.
RL0	(Remote-Local)	Not Implemented.
PP0	(Parallel Poll)	Not Implemented.
DC*	(Device Clear)	Complete Capability ¹ .
DT0	(Device Trigger)	Not Implemented.
C1,2,3,4,7	(Controller)	System Controller. Send Interface Clear (IFC). Send Remote Enable (REN). Respond to Service Request (SRQ). Send Interface Message. Receive Control. Take Control Synchronously.

¹The calculator is not affected by a Clear message when it is either the system controller or the device sending the Clear message.

<p>Output Instructions</p> <p>WRITE X (WRTX)</p> <p>CALL ALPHA 5 C</p>		Byte No.	+(Address code) or -(Register No.)	Data Value	Sends the contents of the X register to the bus or stores it in calculator memory as ASCII coded characters with CR/LF.
<p>FIELD</p> <p>CALL ALPHA 5 L</p>				Field Width (1 thru 255)	Specifies the total number of characters sent by WRTX. Value is sent right-justified with leading spaces filling the excess field width.
<p>WRITE BYTE (WBYTE)</p> <p>CALL ALPHA 5 B</p>		Byte No.	+(Address Code) or -(Register No.)	Decimal Equivalent (0 thru 255)	Sends the binary equivalent of the value in X to either the bus or to a particular data register byte location.
<p>OUTPUT (OUTPT)</p> <p>CALL ALPHA 5 A</p>	Address Code	Number ± of Bytes	Byte No.	Register No.	Sends a string of binary bytes from calculator register storage to the bus.
<p>COMMAND (CMD)</p> <p>CALL ALPHA 5 CALL ALPHA RUN STOP</p> <p>ASCII Character Keys CALL ALPHA</p>			Address Code	Data Value	Sends a string of ASCII characters to the devices specified by the address code parameter in Y.
<p>Input Instructions</p> <p>READ X</p> <p>CALL ALPHA 5 H</p>			Byte No.	+(Address Code) or -(Register No.)	Inputs ASCII coded data values from the bus or recalls a value from the calculator memory and enters the value in X.
<p>READ BYTE (RBYTE)</p> <p>CALL ALPHA 5 G</p>			Byte No.	+(Address Code) or -(Register No.)	Inputs a binary byte from the bus or recalls a specified byte from calculator memory. The decimal equivalent is entered into X.
<p>INPUT</p> <p>CALL ALPHA 5 F</p>	Address Code	Number ± of Bytes	Byte No.	Register No.	Inputs a string of binary bytes from the bus and stores them directly into calculator data registers.
<p>Character Storage Instructions</p> <p>α → STR</p> <p>CALL ALPHA 5 I</p>	Program Step No.	Byte No.	Register No.	Data Value	Stores the binary code of the characters contained by specified steps as bytes in data registers.
<p>STR → α</p> <p>CALL ALPHA 5 D</p>	Program Step No.	Number ± of Bytes	Byte No.	Register No.	Transfers a string of program-step characters from data registers to program memory.
<p>Logical Operator Instructions</p> <p>AND</p> <p>CALL ALPHA 5 M</p> <p>OR</p> <p>CALL ALPHA 5 N</p>			Decimal Value (0 thru 255)	Decimal Value (0 thru 255)	Combines the binary equivalents of the decimal values in X and Y in an AND logic operation and returns the decimal equivalent of the result in X.
<p>Program Instructions</p> <p>DUMP PROGRAM (DUPGM)</p> <p>CALL ALPHA 5 E</p>			Address Code	Program Step No.	Sends the contents of the program memory in machine code to the bus.
<p>LOAD PROGRAM (LDPGM)</p> <p>CALL ALPHA 5 J</p>			Address Code	Program Step No.	Inputs program steps from the bus until an END step has been loaded.

Summary of the Interface Instructions (Continued)

Bus Control Instructions STATUS (STAT) (CALL ALPHA) S (O)					Returns two bytes in X and Y. The binary equivalent of the bytes are defined to specify the status of the various interface lines.
SERIAL POLL (SERPL) (CALL ALPHA) S (K)				Address Code	Inputs a status byte from a specified device on the bus and enters the decimal equivalent of the byte in X.
COMMAND (CMD) (CALL ALPHA) S (CALL ALPHA) (Bus Message Keys) (CALL ALPHA)			Address Code	Data Value	Provides the capability of sending the various bus messages. The bus message key sequences are shown below.

Message	Key Sequence
The Trigger Message	(F) (H)
The Device Clear Message	(F) (K)
The Selected Device Clear Message	(F) (F)
The Remote Message	(F) (B)
The Local Message	(F) (G)
The Local Lockout Message	(F) (L)
The Clear Lockout and Local Message	(F) (C)
The Pass Control Message	(F) (I)
The Abort Message	(F) (A)
The Require Service Message	(F) (D)
Cancel the Require Service Message	(F) (E)



Subject Index

a

Abort Message	34
Address Code Jumpers	41,42
Address Code Parameter	7,11
Address, Listener	7,58
Address, Talker	7,58
AND Instruction	30
ASCII Code	55
ASCII Table	56
Automatic Addressing	11

b

Binary Codes	54
Binary Instructions:	
WRITE BINARY	15
READ BINARY	22
Bus Address Characters	4
Bus Messages, Overview	31

Bus Messages:

Control	31
Data	12
Bus Signal Lines	4
Byte, Definition	10
Byte Number Parameter	10

c

Cable, Interconnecting	2
Cable Length Restrictions	2
Calculator Option 002	1
Character Storage	25
Clear Messages	32
Clear Lockout and Set Local Message	34
Code Conversion	54
Command Instruction:	
Bus Control Messages	31
Data Message	18
Component Locators:	
A1 Board	48
A2 Board	50
A3 Board	52
Connecting the Interface	2
Control Logic	44
Control Messages	6,31
Controller Function	45
Controller Jumper	42

d

Data, Definition	10
Data Input Instructions	20
Data Message	6,12
Data Message Instructions	13
Data Output Instructions	14
Data Transfer Rate	5
Definition of Terms	10
Device Clear Message	33
Disassembly Procedure, A2 and A3	41
DUMP PROGRAM Instruction (DUPGM)	31

e

Equipment Supplied	1
Error Messages	72

f

FIELD Instruction	15
-------------------	----

h

Handshake Logic	44
HP-IB Functions, Theory of Operation	45
HP-IB, Overview	5
HP-IB, Technical Description	60

i

IEEE Standard 488-1975	1,60
Input Instruction	22
Input Instructions, Data	20
Input Multiplexer	44
Installing the Interface	2
Instruction Summary	68
Interconnecting Cables	2
Interface Functions:	
General	63
98135A	64,65

j

Jumper, Address	41,42
Jumper, System Controller	41,42

k

Key Overlay 9

l

Listener Address 7,58
 Listener, Definition 7
 Listener, Function 45
 LOAD PROGRAM Instruction (LDPGM) . 31
 Local Lockout Message 33
 Local Message 33
 Logic Levels 4
 Logical Operators:
 AND Instruction 30
 OR Instruction 30

m

Maximum Number of Devices 3
 Memory Usage 4
 Mounting Fasteners 3

n

Number of Bytes Parameter 11

o

OR Instruction 30
 OUTPUT Instruction 15

p

Parameters, Definitions 10
 Parts, Replaceable 46
 Pass Control Example Program 38
 Pass Control Message 34
 Power Requirements 5
 Program Character Storage 25
 $\alpha \rightarrow$ STR Instruction 25
 STR $\rightarrow \alpha$ Instruction 26
 Program Step Parameter 13
 Program Instructions:
 Dump Program Instruction 31
 LOAD PROGRAM Instruction 31

r

READ BYTE Instruction 22
 READ X Instruction 20
 Recalling Data 20,22
 Register Number Parameter 10
 Remote Message 33
 Remote Programming Example 19
 Replaceable Parts 46
 Require Service Message 35
 ROM, Theory of Operation 44

s

Sales and Service Offices 66
 Schematics:
 A1 Board 49
 A2 Board 51
 A3 Board 53
 Select Code 4
 Selected Device Clear Message 33
 Sending Service Requests 37
 Serial Polling 36
 SERIAL POLL Instruction 37
 Service Requests 34,35,36
 Status Byte, Interface 35
 Status Byte Message 36
 STATUS Instruction 35
 Storing Data 14,15
 Summary of Interface Instructions 68

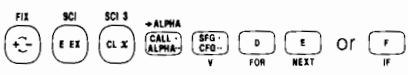


t

Talker Address 7,58
 Talker, Definition 5
 Talker Function 45
 Technical Specifications 5
 Temperature, Operating Range 5
 Theory of Operation 43
 Trigger Message 32

w

Wire Connections 47
 WRITE BYTE Instruction 15
 WRITE X Instruction 14

Error Messages

* OVERFLOW	Number or result exceeds calculating range.
* SQRT OF NEG #	
* DIVISION BY ZERO	
* LOG OF # <=0	
* NO I/O DEVICE	Peripheral device or interface not connected.
ILLEGAL ADDRESS	Improper step address or storage register specified.
ILLEGAL ARGUMENT	Improper value for operation (e.g., improper scale or axis parameters).
MEMORY OVERFLOW	Program instruction, storage register assignment, or program loaded from tape exceeds available memory.
GOSUB OVERFLOW	More than seven subroutines (including special functions) nested at a time.
KEY NOT DEFINED	Special function just called is not defined.
IMPROPER SYNTAX	Incorrect use of  or 
* CHECKSUM ERROR	Program or data loaded into calculator not identical to that in file; this usually indicates a dirty tape head or a worn tape.
* VERIFY FAILED	Program or data in file not identical to that in calculator.
WRONG FILE TYPE	Attempting to load an empty, extra, or binary file; recording on an extra file.
END OF TAPE	End of tape reached during MARK operation. Also indicates a broken or defective tape; if the tape does not appear to be broken, (advance it using the drive wheel), replace the cartridge, press  , and continue.
PROTECTED TAPE	The cartridge RECORD slide is positioned to prevent MARK and RECORD operations.
SECURED MEMORY	Attempting to list, edit, or record a secured program.
MISSING FOR STMT	
LABLE NOT FOUND	
FILE NOT FOUND	
CARTRIDGE OUT	
MISSING GOSUB	
NUMBER TOO BIG	The useable range of an instruction has been exceeded.
FIELD TOO SMALL	The current field is too small to contain the value being output.
NOT ADDRESSED	The calculator is not active controller and has not been addressed to talk or listen.
NOT CONTROLLER	The calculator is not system controller and a command that only the system controller can send has been executed. The command is not sent to the bus.

*These messages are suppressable; see "Flags" in Section 3 of the calculator operating manual.

HEWLETT  PACKARD



PART NO. 98135-90000
MICROFICHE NO. 98135-99000

PRINTED IN U.S.A.
May 3, 1979