Instrument Interface with

HP-IB

STUDENT WORKBOOK

data systems training

22999-90248

January 1980

PLEASE LOG ON

:USER 14

:AC, EPO, P * allocate a contiguous

:CL catalog list

in edit

?? gives a list of commands

:EX, SP * to get out of session

CONTENTS

HP Computer Museum
www.hpmuseum.net

For research and education purposes only.

COURSE SCHEDULE

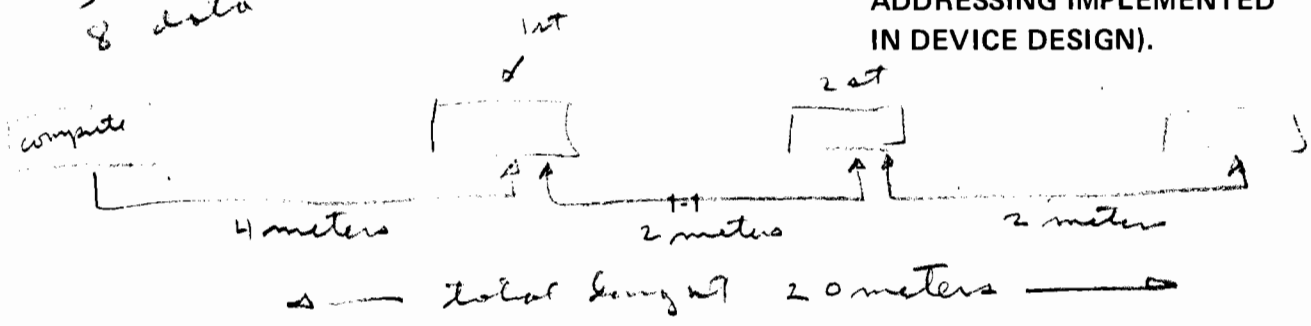
DAY 1	DAY 2	DAY 3	DAY 4
LESSONS 1, 2, 3	LESSON 4, 5	LESSON 6, 7	LESSON 8, 9
NOON	NOON	NOON	NOON
1, 2	2, 3, 4	4, 5, 6	7, 8

1

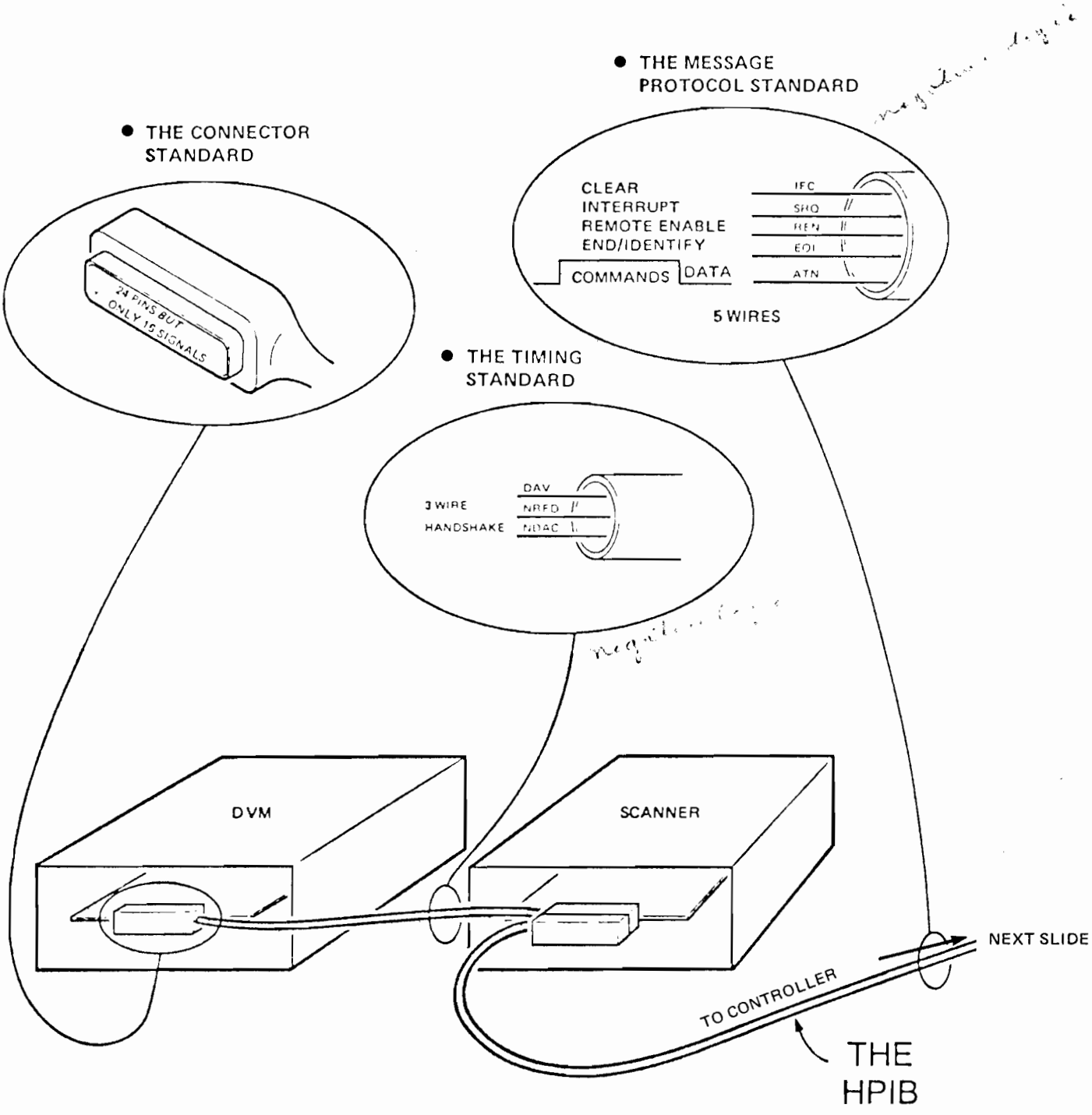
HP-IB OVERVIEW

HP INTERFACE BUS SPECIFICATIONS

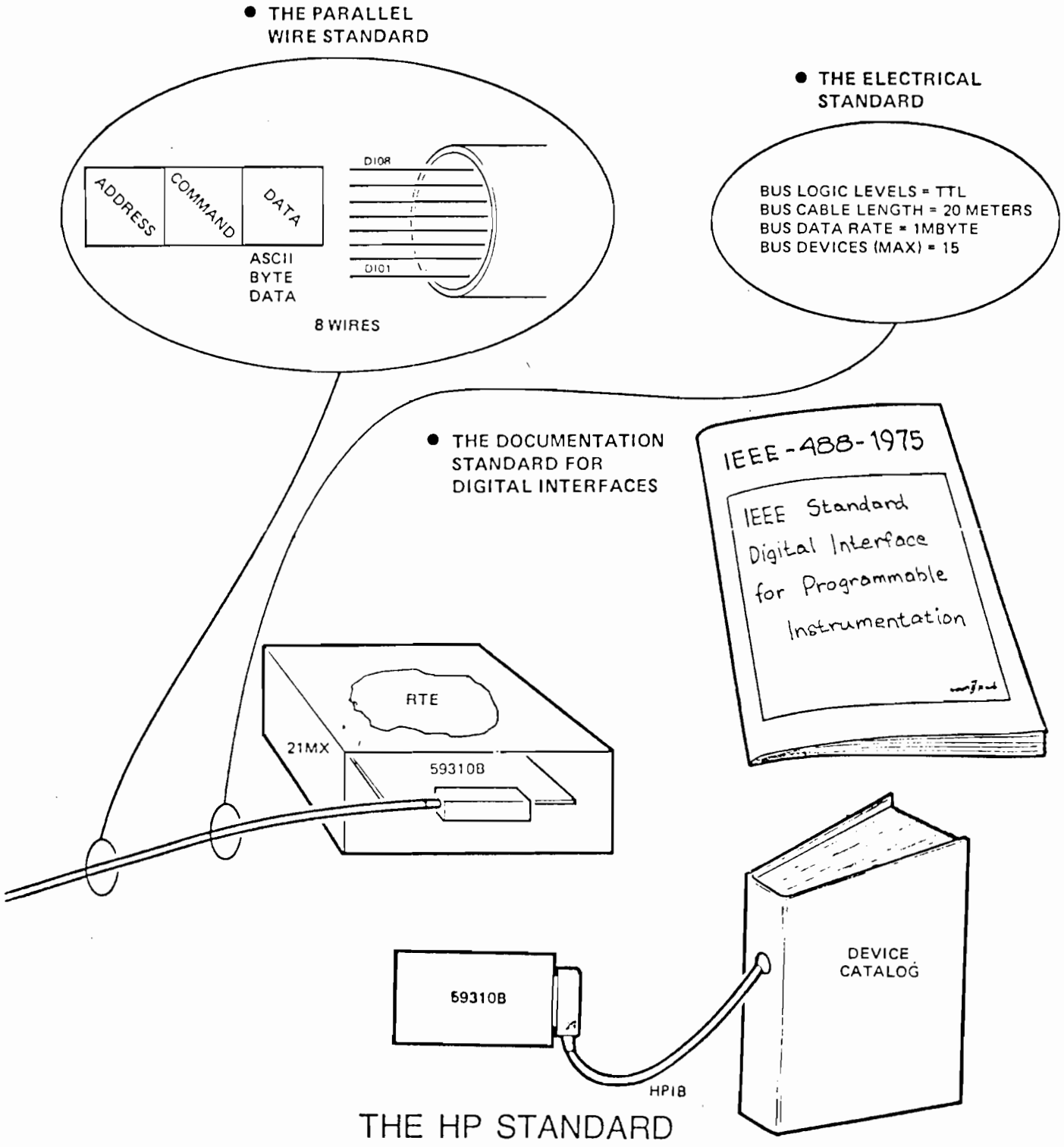
INTERCONNECTED DEVICES:	UP TO 15 MAXIMUM ON ONE BUS INCLUDING CONTROLLER.
INTERCONNECTION PATH:	LINEAR NETWORK OVER 20 METERS TOTAL LENGTH.
SIGNAL LINES:	SIXTEEN TOTAL: 8 DATA LINES AND 8 CONTROL LINES.
MESSAGE TRANSFER SCHEME:	BYTE-SERIAL BIT PARALLEL ASYNCHRONOUS USING 3-WIRE HANDSHAKE TECHNIQUE.
DATA RATE:	ONE MEGABYTE PER SECOND MAXIMUM OVER LIMITED DISTANCES: 250-500 KBYTES/SEC OVER FULL TRANSMISSION LENGTH W/ OLD INTERFACE, 900 KBYTES/SEC W/ NEW INTERFACE.
INTERFACE FUNCTIONS:	TEN TOTAL; FIVE PRIMARY COMMUNICATION FUNCTIONS AND FIVE SPECIAL-PURPOSE FUNCTIONS.
ADDRESS CAPABILITY:	31 PRIMARY TALK AND LISTEN ADDRESSES PER BUS; 31 SECONDARY ADDRESSES AVAILABLE PER PRIMARY ADDRESSED DEVICE (IF SECONDARY ADDRESSING IMPLEMENTED IN DEVICE DESIGN).



BIT PARALLEL/BYTE SERIAL

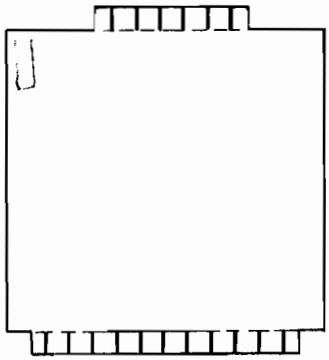


BYTE SERIAL



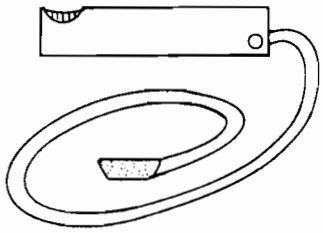
THE HP STANDARD

THE HP-IB KIT HARDWARE



HP 59310B-60101
DUPLEX I/O
COMPUTER
INTERFACE CARD

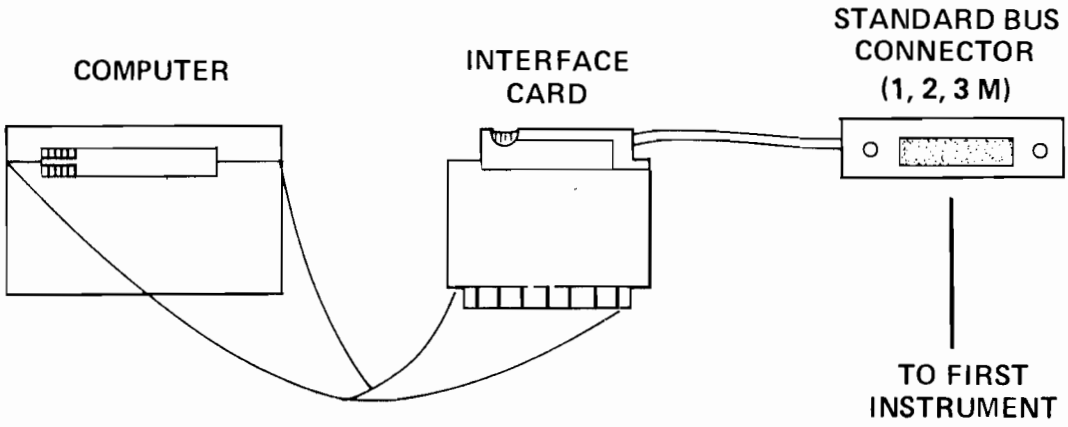
CONNECTING THE
COMPUTER UP TO
14 HP-IB COMPATIBLE
INSTRUMENTS



HP 59310-60002
3 METER
HOOD AND CABLE

CONNECTS COMPUTER
INTERFACE CARD
TO STANDARD BUS
CONNECTOR OF
FIRST INSTRUMENT

THE CONFIGURED SYSTEM



AVOID CONNECTING MORE THAN 2-3 CABLES AT ANY DEVICE TERMINAL SINCE ANY FORCE APPLIED TO THAT TERMINAL COULD CAUSE A LEVERAGE FORCE ON THE CONNECTORS RESULTING IN CONNECTOR DAMAGE. ONLY ONE CABLE PER DEVICE. (TWO CONNECTIONS PER DEVICE) IS REQUIRED FOR SUCCESSFUL HOOK-UP.

RTE HP-IB SOFTWARE

HP Part No.	File Name	Description
59310-16002	%1DV37	Driver DVR37 without Service Request (SRQ) capability
59310-16003	%2DV37	Driver DVR37 with SRQ capability
59310-12001	%IB4A	RTE HP-IB LIBRARY
92101-12002	%BAMLB	BASIC Memory Resident Library
92101-12003	%BASLB	BASIC Subroutine Library
59310-16005	%SRQ.P	SRQ/TRAP program for BASIC

BUS OPERATION

★ ALL ACTIVE CIRCUITRY IS CONTAINED WITHIN THE VARIOUS HP-IB DEVICES.

★ THE INTERCONNECTING CABLE CONTAINING 16 SIGNAL LINES IS ENTIRELY PASSIVE.

★ THE CABLE'S ROLE IS LIMITED TO THAT OF INTERCONNECTING ALL DEVICES TOGETHER IN PARALLEL.

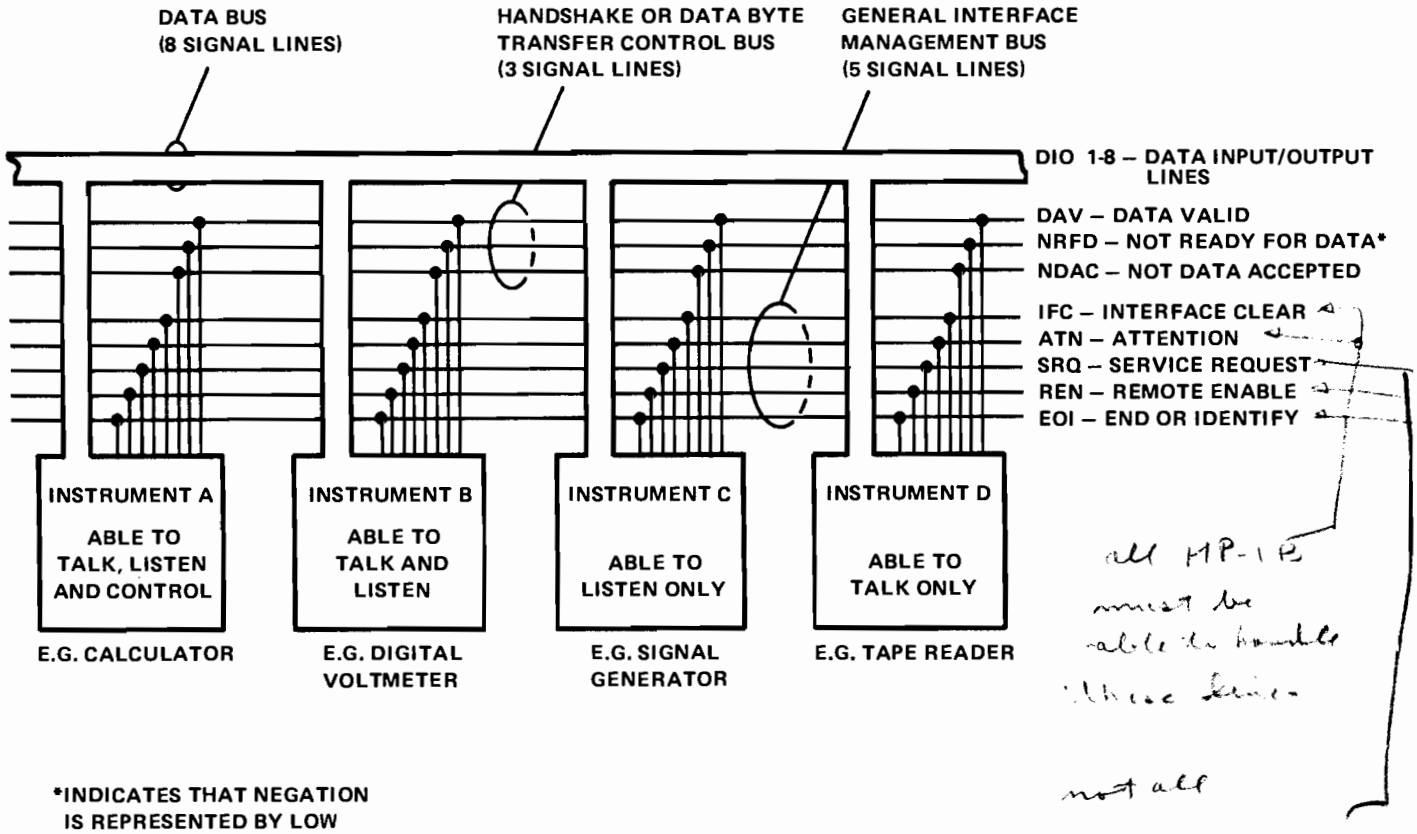
★ THIS PERMITS ANY ONE DEVICE TO TRANSFER DATA TO ONE OR MORE OTHER PARTICIPATING DEVICES.

★ THE INTEGRITY OF THE ENTIRE BUS SYSTEM DEPENDS UPON THE INDIVIDUAL CONFORMITY OF EACH DEVICE CONNECTED TO THE BUS.

2

THE BUS SYSTEM

THE BUS SYSTEM



*INDICATES THAT NEGATION IS REPRESENTED BY LOW STATE ON THESE TWO LINES ONLY.

ROLES OF DEVICES ON THE BUS

EVERY HP-IB BUS INSTRUMENT MUST FALL INTO AT LEAST ONE OF THE FOLLOWING CATEGORIES:

CONTROLLER – INSTRUMENT THAT HAS THE ABILITY TO CONTROL OTHER INSTRUMENTS ON THE BUS

TALKER – INSTRUMENT WITH THE ABILITY TO TRANSMIT DATA VIA THE BUS TO ONE OR MORE LISTENERS

LISTENER – INSTRUMENT WITH THE ABILITY TO RECEIVE MESSAGES TRANSMITTED BY A BUS TALKER

– OR –

A COMBINATION OF ALL THE ABOVE

SYSTEM CONTROLLER

- HAS CAPABILITY TO MANAGE OPERATION OF THE 5 BUS MANAGEMENT AND 3 DATA CONTROL LINES:
  - DESIGNATING WHICH DEVICES ARE TO SEND AND RECEIVE DATA
  - COMMAND SPECIFIC ACTIONS WITHIN OTHER DEVICES
- ONE DEVICE IS DESIGNATED DURING SYSTEM CONFIGURATION AS THE SYSTEM CONTROLLER TO GAIN ABSOLUTE CONTROL OF THE BUS.

EXCLUSIVE FUNCTIONS

INTERFACE CLEAR: ALLOWS THE BUS TO BE SET TO A KNOWN QUIESCENT STATE BY SETTING IFC TRUE. DEVICE REFERENCE MANUALS DESCRIBE THE STATE A PARTICULAR DEVICE WILL GO INTO.

REMOTE ENABLE: ALLOWS PROGRAMMABLE DEVICES TO BE SWITCHED FROM LOCAL TO REMOTE BY SETTING REN TRUE.

CONTROLLER

- MANAGES THE FLOW OF INFORMATION BETWEEN DEVICES AND CAN DO EVERYTHING A SYSTEM CONTROLLER CAN DO EXCEPT IFC AND REN:

  - ADDRESSING: ALLOWS ADDRESSING ONE INSTRUMENT AT A TIME AS A TALKER AND ONE OR MORE AS LISTENERS BY SETTING ATN TRUE.
  - SERVICE REQUEST: ALLOWS MONITORING OF SERVICE REQUEST LINE (SRQ) OR ENABLING IT TO CAUSE AN INTERRUPT TO DETERMINE IF A DEVICE ON THE BUS IS REQUESTING SERVICE.
  - SERIAL POLL: ALLOWS STATUS RECOGNITION FOR EACH DEVICE ON THE BUS. USUALLY USED TO DETERMINE WHICH DEVICE REQUESTED SERVICE.
  - PARALLEL POLL: ALLOWS DEVICE STATUS RECOGNITION VIA DIO LINES ON A BIT PER DEVICE (OR GROUP OF DEVICES) BASIS WHEN BOTH ATN AND EOI ARE TRUE. UP TO 8 DEVICES OR GROUPS OF DEVICES CAN BE EXAMINED.

TALKER

- ANY DEVICE CAPABLE OF SENDING OR TRANSMITTING INFORMATION ON THE BUS.

- ONLY ONE ACTIVE TALKER CAN PLACE INFORMATION ON THE BUS AT A GIVEN TIME.

ADDRESSABLE TALK:	DEVICE BECOMES ADDRESSED TO TALK WHEN A CONTROLLER SENDS IT A TALK ADDRESS. (SWITCH) SELECTABLE)
CONTINUOUS TALK:	DEVICE NEED NOT BE ADDRESSED AS A TALKER IN ORDER TO SEND DATA, (SWITCH SELECTABLE).
END OF RECORD:	EOI IS SET LOW WHEN ASCII LINE FEED (12_8) IS OUTPUT. THIS IS END OF DATA INDICATION.

driver needs a EOI - or be programmed to not see a EOI by program to send EOI, also if EOI is not used, at user time out

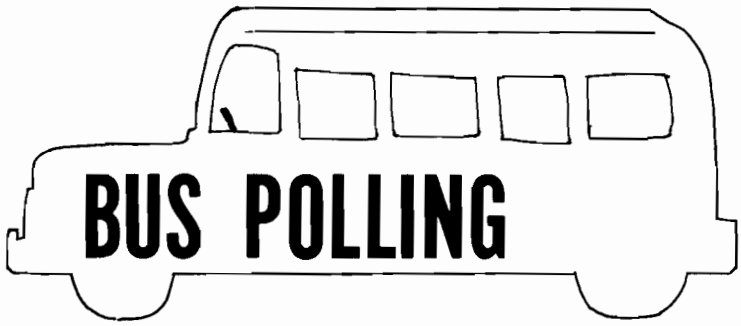
LISTENER

- ANY DEVICE CAPABLE OF RECEIVING OR ACCEPTING INFORMATION ON THE BUS.
- UP TO 14 ACTIVE LISTENERS AT A GIVEN TIME.

- ANY DEVICE CAPABLE OF BOTH SENDING AND RECEIVING INFORMATION ON THE BUS.

ADDRESSABLE LISTENERS ARE CONTROLLED IN THE SAME MANNER AS TALKERS.

TALKER - LISTENER



A POLL ENABLES THE COMPUTER TO LEARN STATUS OR CONDITION OF DEVICES ON THE BUS. EACH DEVICE MUST BE DESIGNED TO BE POLLED.

SERIAL POLL

THE CONTROLLER POLLS DEVICES ONE AT A TIME IN SEQUENCE. WHEN POLLED, A DEVICE TRANSMITS A SINGLE BYTE OF INFORMATION TO INDICATE THE STATUS OF THE DEVICE. TYPICAL STATUS MIGHT INDICATE DEVICE IS OVERLOADED (POWER SUPPLY), THE DEVICE OUTPUT HAS STABILIZED AT A LOW LEVEL (SIGNAL GENERATOR), OR THE DEVICE HAS REQUESTED SERVICE.

PARALLEL POLL

RETURN A ONE BIT PER DEVICE STATUS TO THE CONTROLLER OF UP TO 8 INSTRUMENTS OR GROUPS OF INSTRUMENTS.

DEVICE ADDRESSES

EACH DEVICE HAS ONE OR MORE 5-BIT ADDRESSES WHICH ALLOW IT TO. . .

TALK

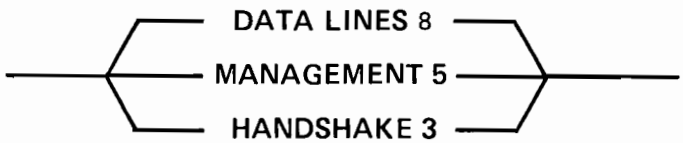
LISTEN

TALK AND/OR LISTEN

INSTRUMENT ADDRESS ASSIGNMENTS ARE SELECTED BY A SET OF DIP SWITCHES USUALLY LOCATED IN BACK OF THE INSTRUMENT.

HP-IB COMMUNICATIONS STRUCTURE

- 16 LINES ARE CONTAINED IN THE HP-IB CABLING



- 8 DATA LINES (DI01-DI08) CARRYING CODED INFORMATION TO AND FROM DEVICES

- 5 GENERAL BUS MANAGEMENT LINES ARE USED TO MANAGE AN ORDERLY FLOW OF INFORMATION ACROSS THE BUS

IFC, ATN
SRQ, REN
EOI

- 3 DATA TRANSFER CONTROL LINES USED TO EFFECT THE TRANSFER OF EACH BYTE OF CODED DATA OVER THE EIGHT DATA LINES – CALLED THE HANDSHAKE PROCESS

GENERAL BUS MANAGEMENT

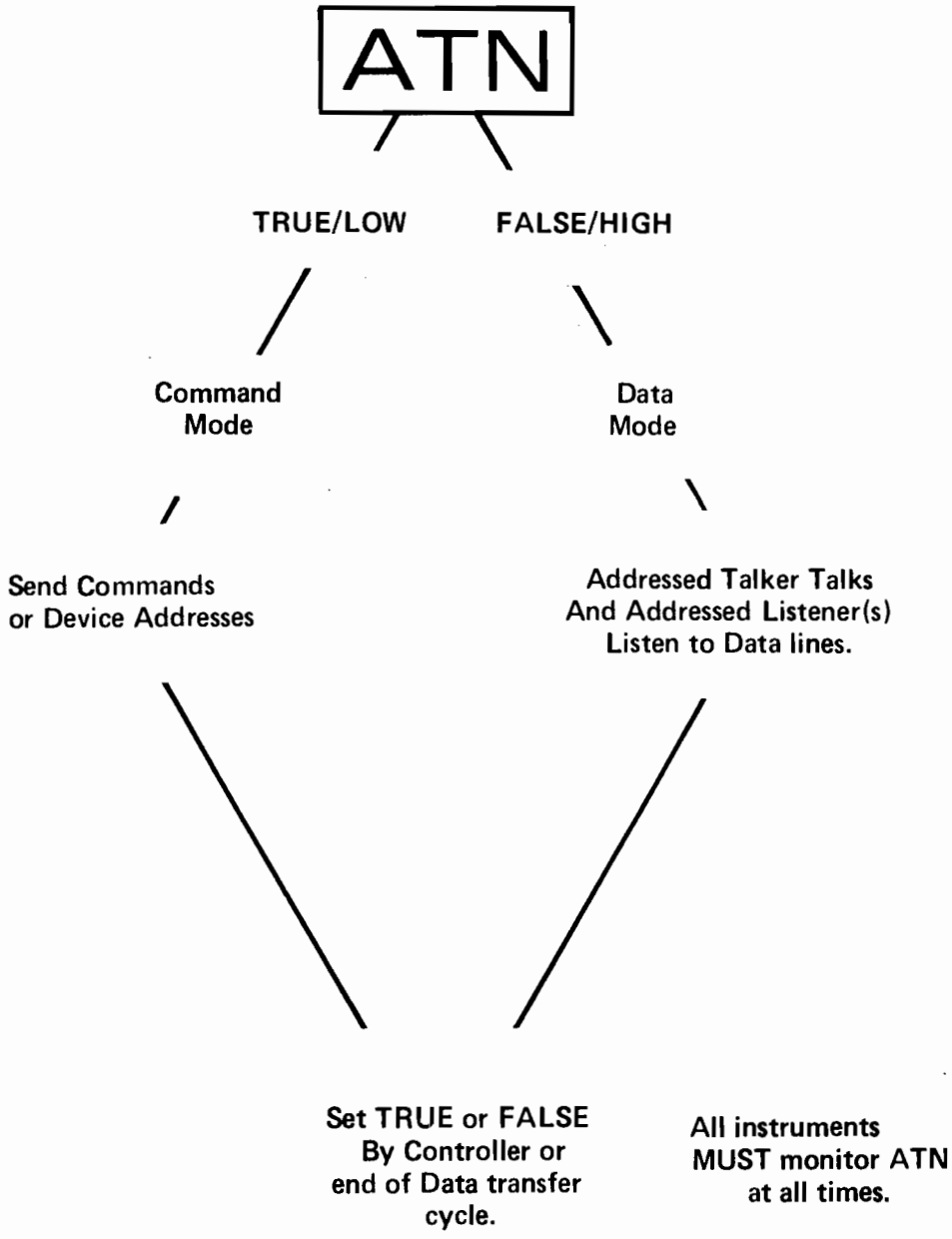
LINE MNEMONIC	DEFINITION
ATN	ATTENTION
*IFC	INTERFACE CLEAR
*REN	REMOTE ENABLE
SRQ	SERVICE REQUEST
EOI	END OR IDENTIFY

FUNCTION ON THE BUS

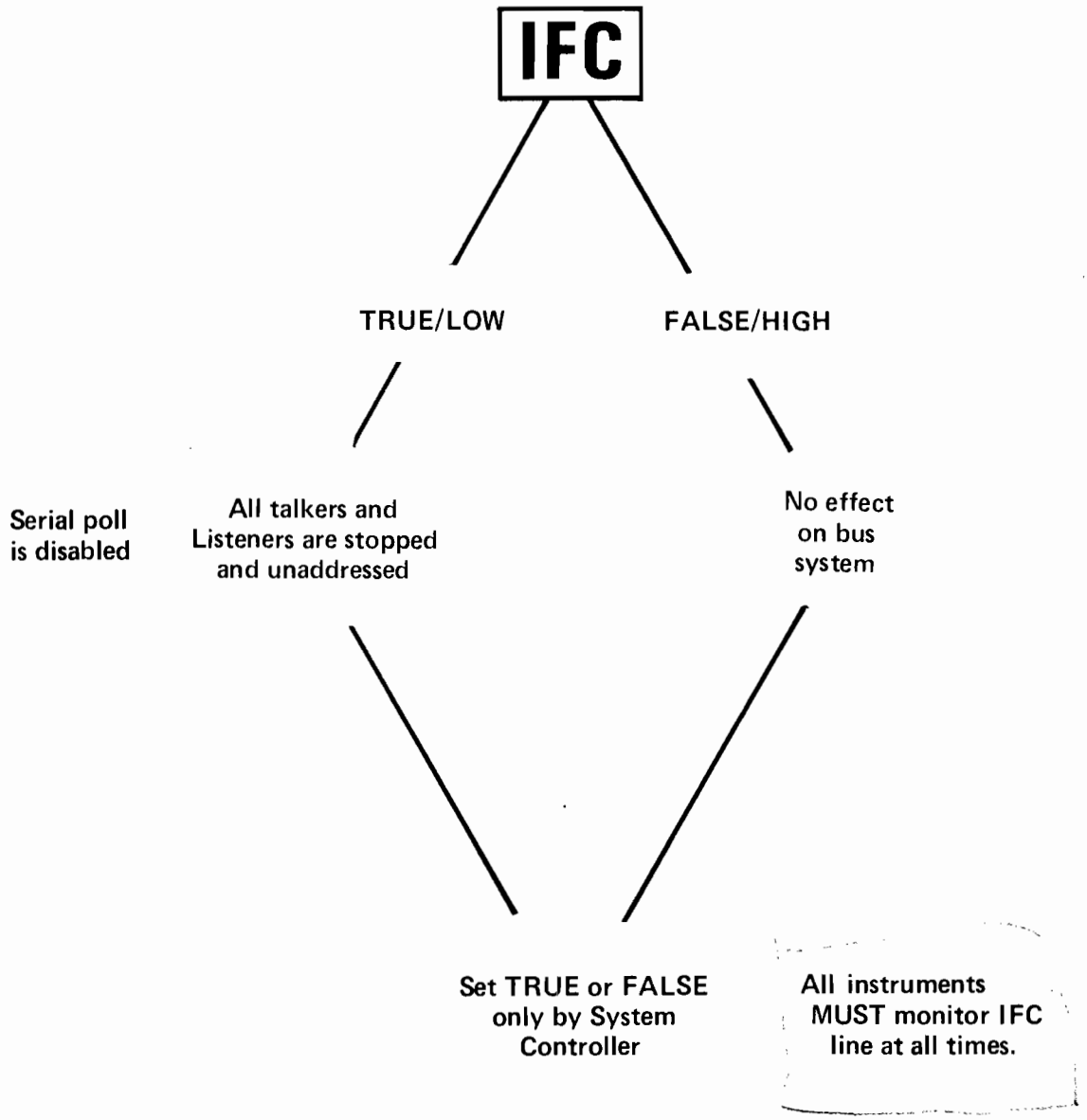
To manage an orderly flow of data across the bus lines.

***Managed only by system controller**

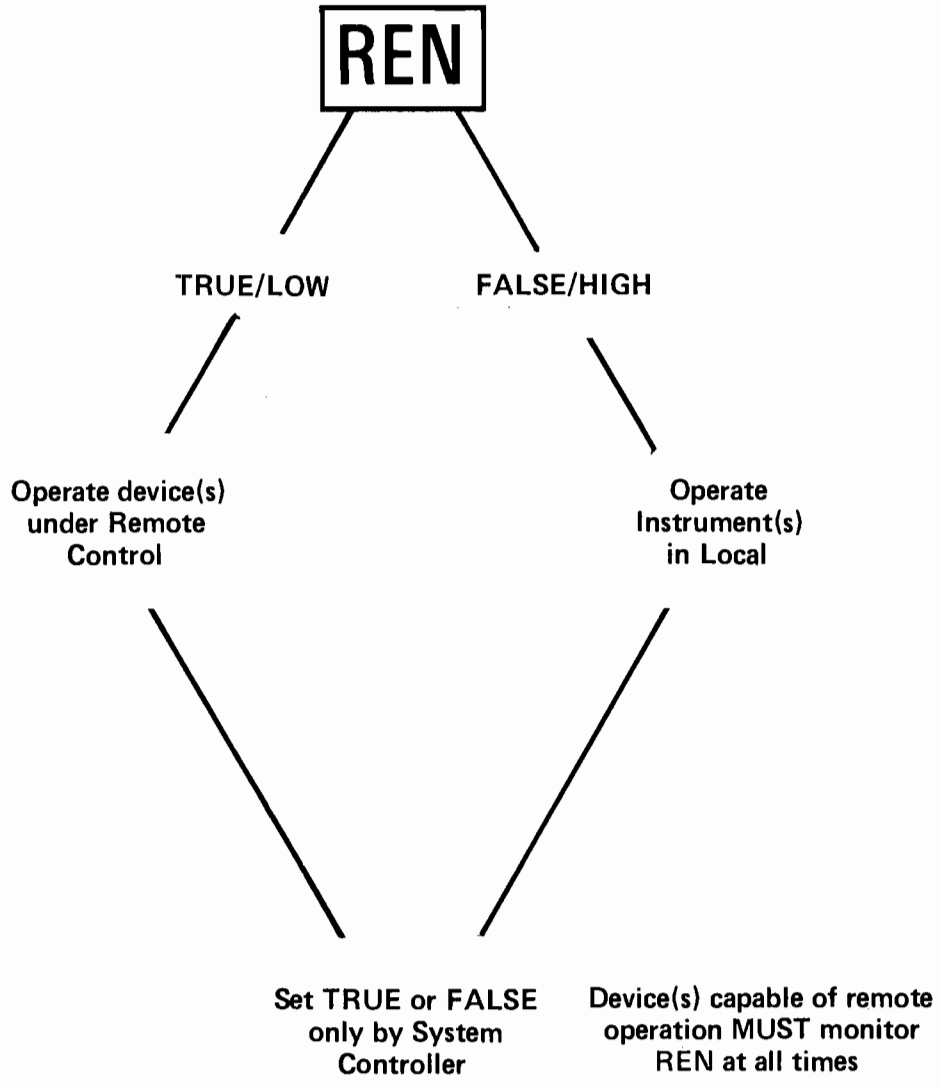
ATTENTION


ATN
TRUE/LOW
FALSE/HIGH
Command Mode
Data Mode
Send Commands or Device Addresses
Addressed Talker Talks And Addressed Listener(s) Listen to Data lines.
Set TRUE or FALSE By Controller or end of Data transfer cycle.
All instruments MUST monitor ATN at all times.

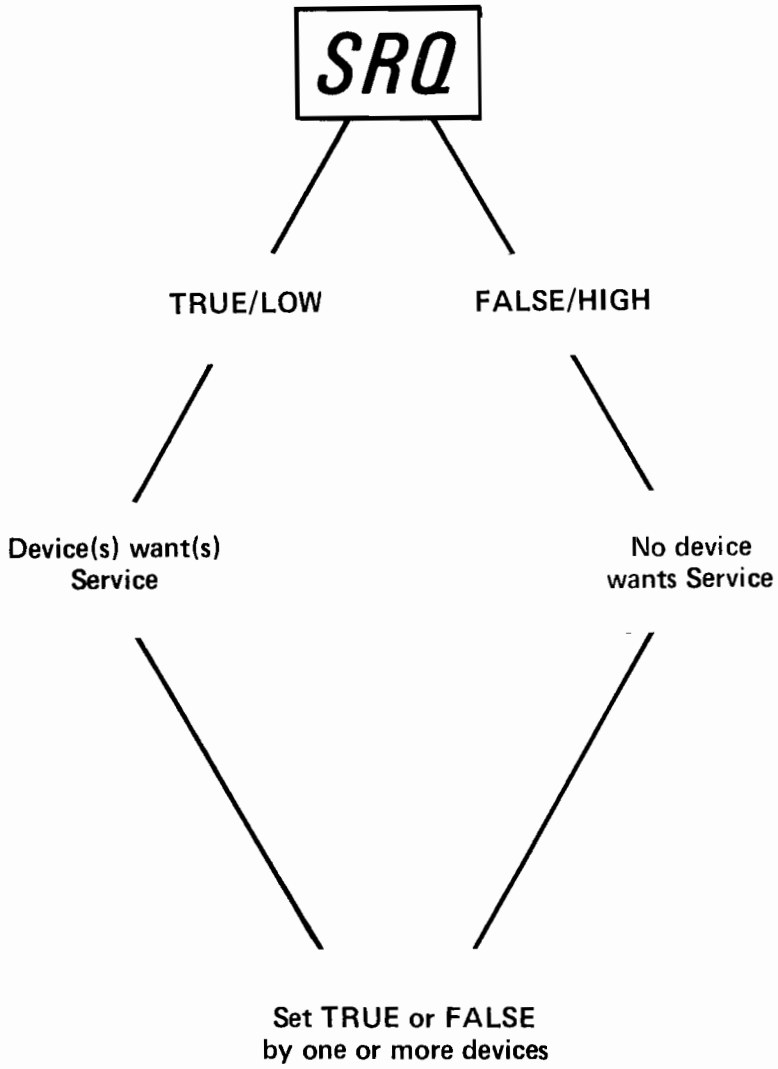
INTERFACE CLEAR



REMOTE ENABLE

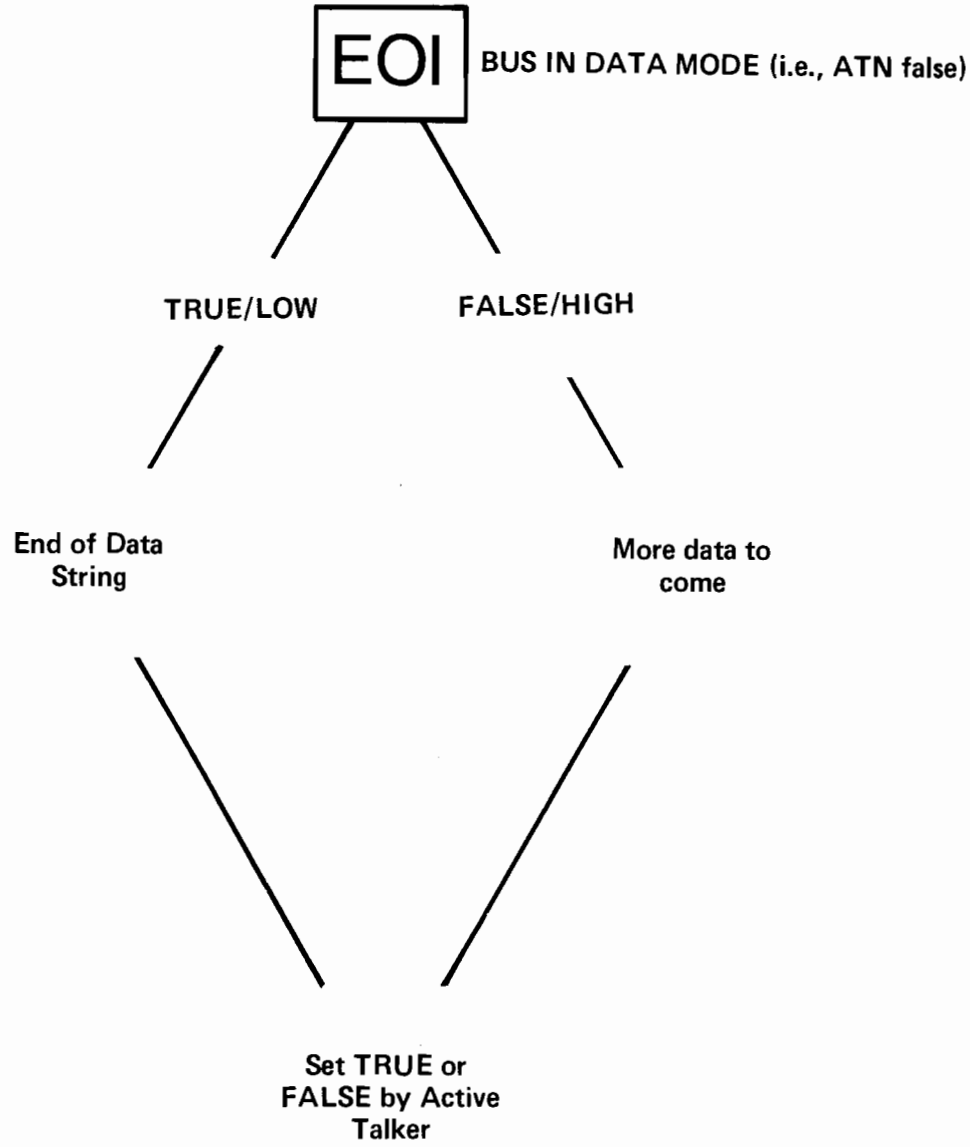


SERVICE REQUEST



SRQ may be set true by an instrument at any time except when IFC is true.

END OR IDENTIFY



EOI true when bus is in command mode (ATN true) will initiate a parallel poll.

set high after the last byte of transfer

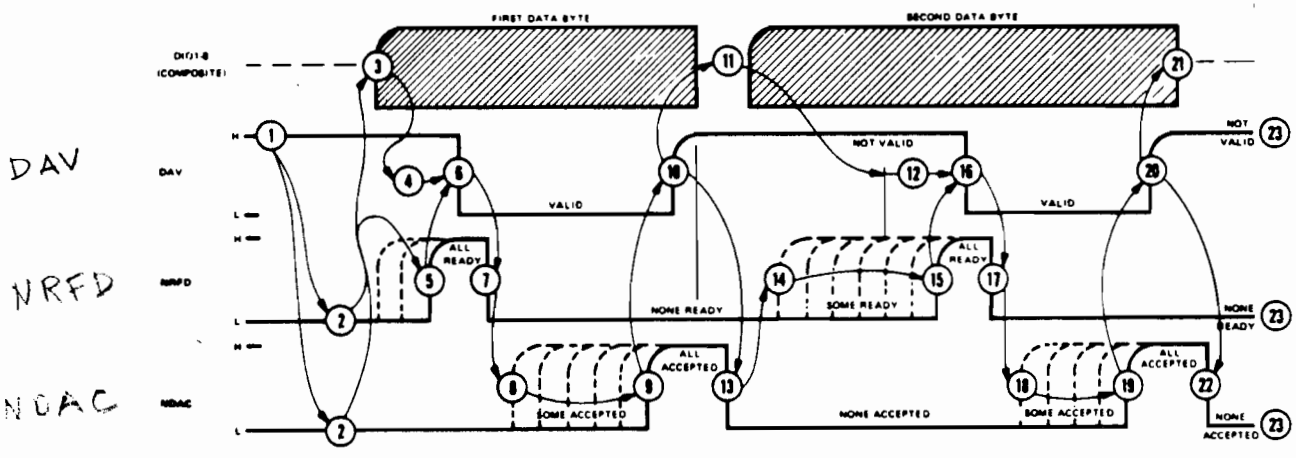
DATA BYTE TRANSFER CONTROL

LINE MNEMONIC	DEFINITION
NRFD	NOT READY FOR DATA
NDAC	NOT DATA ACCEPTED
DAV	DATA VALID

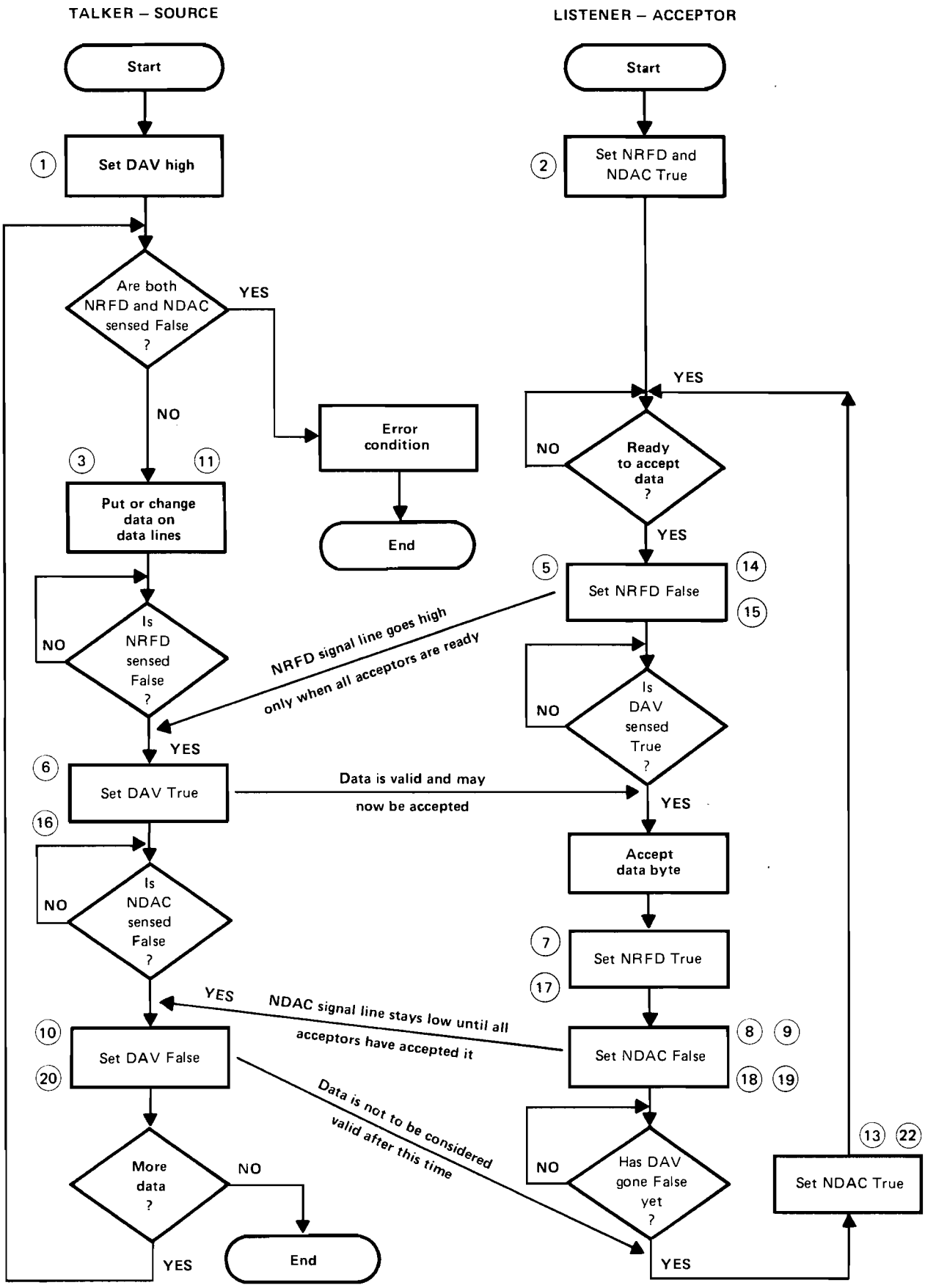
FUNCTION ON THE BUS

TO EFFECT THE TRANSFER OF EACH BYTE OF DATA OVER THE BUS FROM AN ADDRESSED TALKER TO ALL ADDRESSED LISTENERS.

MANAGED BY DEVICES ADDRESSED TO TALK AND/OR LISTEN WHILE IN PROCESS OF DATA TRANSFER



4 AND 12 = DELAY TIME FOR LINES TO SETTLE



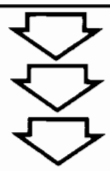
THE HANDSHAKE PROCESS

RELATIONSHIP BETWEEN ATN AND HANDSHAKE LINES

Mode	ATN	NRFD		NDAC		DAV	
		TRUE	FALSE	TRUE	FALSE	TRUE	FALSE
ADDRESS	TRUE	One or more units not ready for data	All units ready for data	One or more units have not accepted data	All units have accepted data	Controller has valid data on DIO lines	Controllers data is not valid
		1. Driven by all units except controller 2. Sensed by controller 3. All units set NRFD and NDAC to valid state within 200 nanoseconds after ATN goes LOW				1. Driven by controller 2. Sensed by listeners 3. See DAV above for timing	
DATA	FALSE	One or more listeners not ready for data	All addressed listeners ready for data	One or more listeners have not accepted data	All addressed listeners have accepted the data	The addressed talker has valid data on lines	The addressed talker data is not valid
		1. Driven by all units addressed to listen. 2. Sensed by the unit addressed to talk. 3. All units not addressed will not drive. 4. All addressed listeners set both NRFD and NDAC to valid within 200 nanoseconds after ATN goes HIGH.				1. Driven by the instruments addressed to TALK 2. Sensed by ALL instruments addressed to LISTEN 3. See DAV above for timing.	

TRUE = LOW

EXAMPLE 1: TYPICAL ROLES



1. THE CONTROLLER DICTATES THE ROLE OF EACH OF THE OTHER DEVICES BY SETTING THE ATTENTION LINE TRUE AND SENDING TALK OR LISTEN ADDRESSES ON THE DATA LINES.

2. WHEN THE ATTENTION LINE IS TRUE, ALL ADDRESSABLE DEVICES MUST LISTEN TO THE DATA LINES.

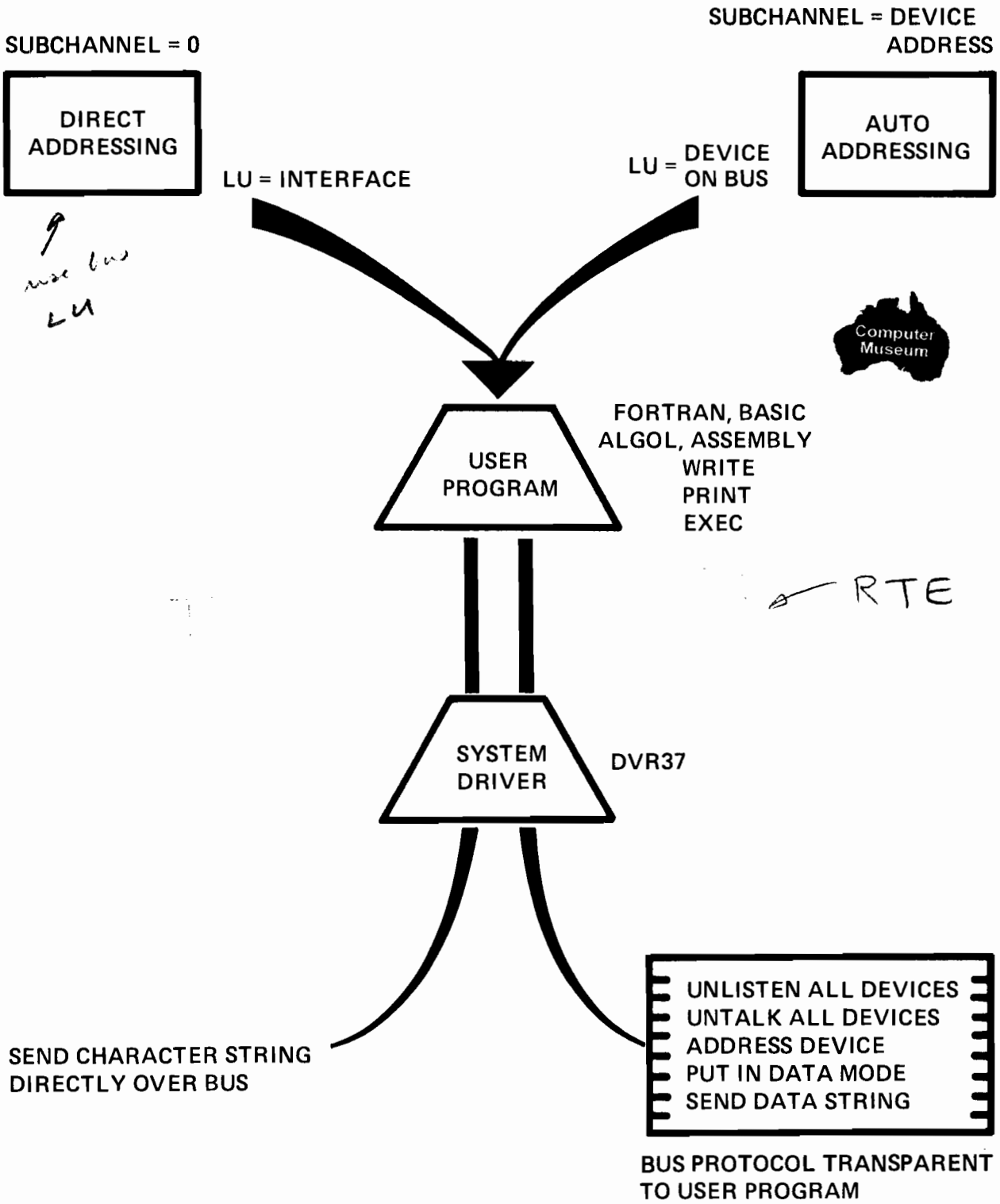
3. WHEN THE ATTENTION LINE IS HIGH OR FALSE, ONLY THOSE DEVICES THAT HAVE BEEN ADDRESSED WILL ACTIVELY SEND OR RECEIVE DATA WHILE ALL OTHERS IGNORE THE DATA LINES.

4. WHEREVER A TALK ADDRESS IS PUT ON THE DATA LINES WHILE THE ATTENTION IS TRUE, ALL OTHER TALKERS ARE AUTOMATICALLY UNADDRESSED.

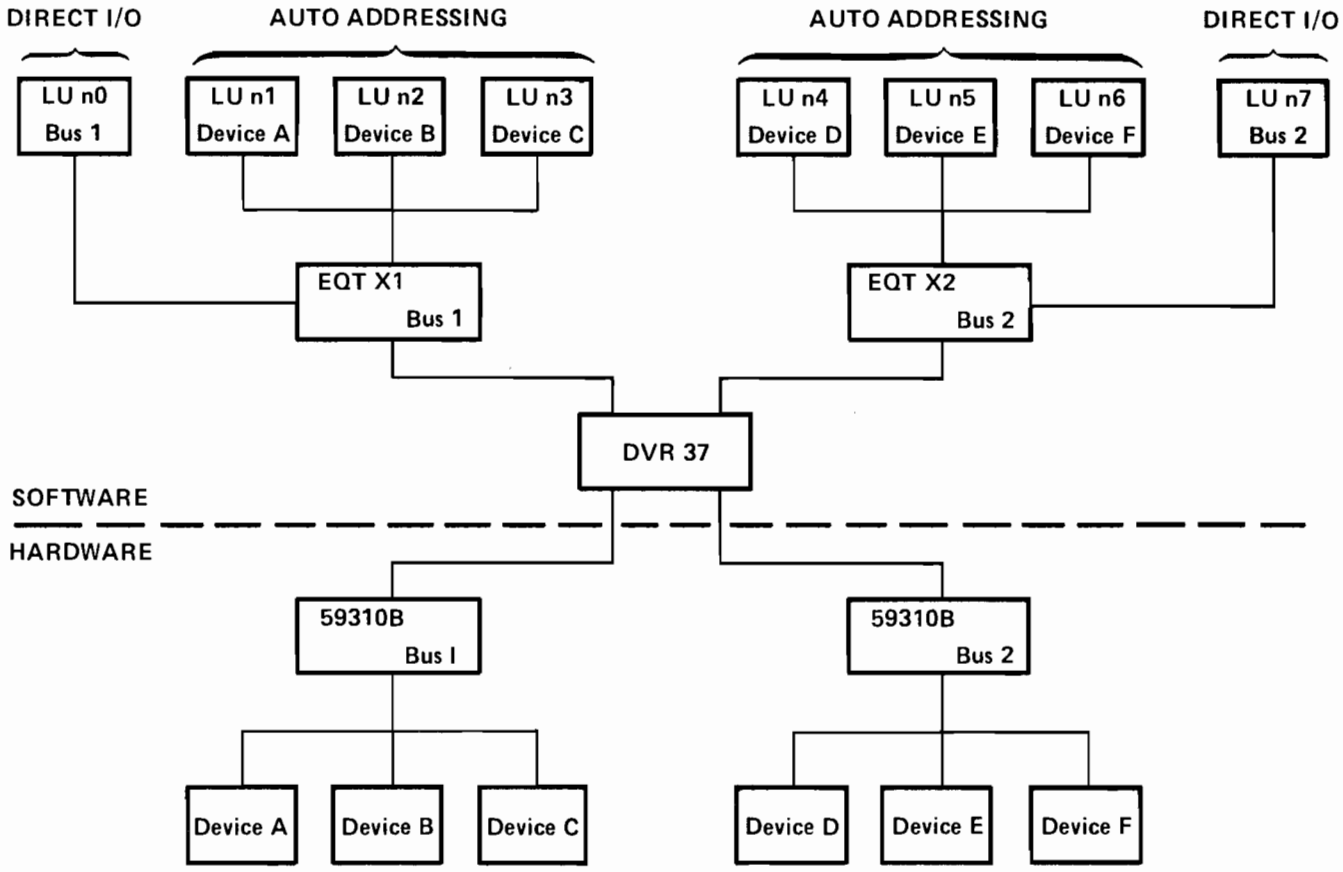
3

RTE I/O and HP-IB ADDRESSING

TWO METHODS OF PROGRAMMING

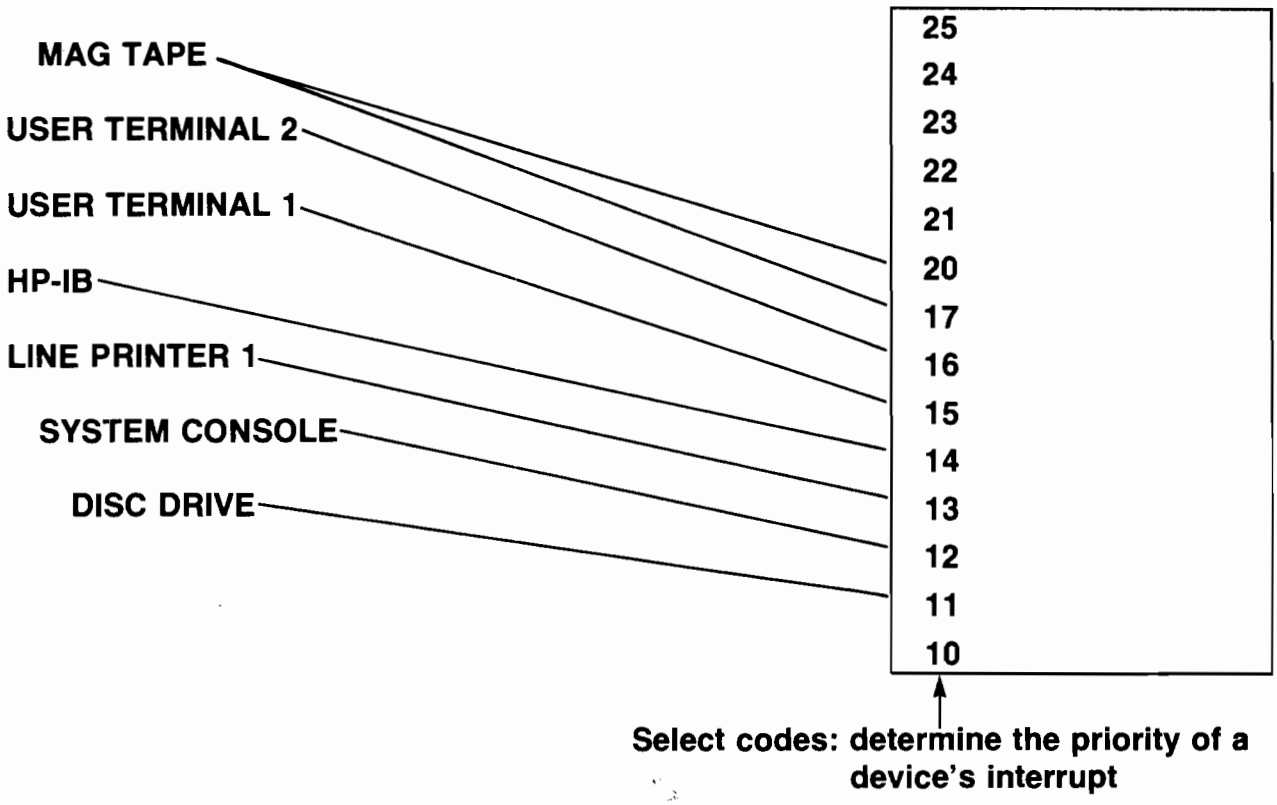


HP-IB INTERFACE STRUCTURE



RTE I/O STRUCTURE

Each peripheral device is connected by a cable to an interface card. The interface card is plugged into a numbered I/O SLOT in the back of the computer.



When an RTE system is generated, the select code assignments are incorporated into RTE's I/O structure.

EQT's

At generation time, each device is assigned an EQUIPMENT TABLE (EQT) number. This number represents an entry in RTE's EQUIPMENT TABLE (EQT).

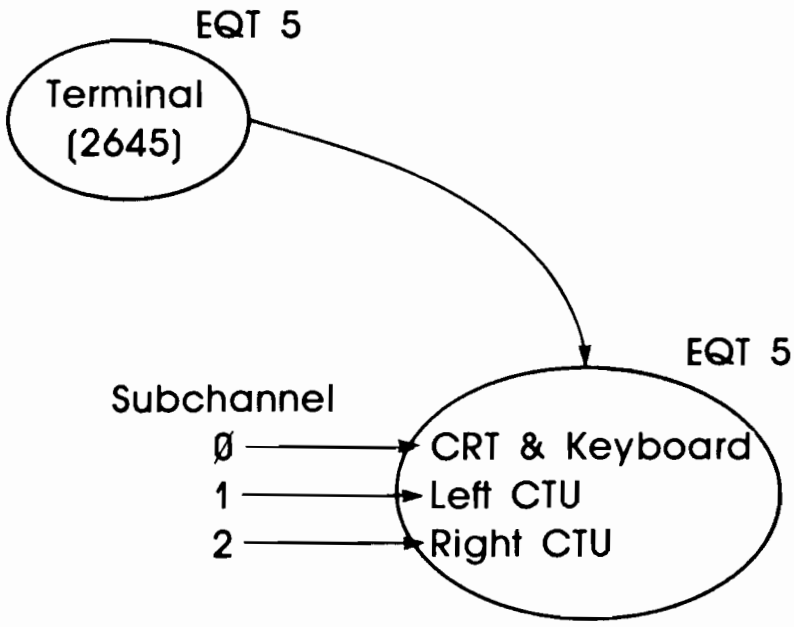
EQT

Eqt numbers	select code	driver	last subchannel addressed	
1	11	DVR32		(disc)
2	12	DVRØ5		(system console)
3	13	DVA12		(line printer 1)
4	14	DVR37		(HP-IB)
5	15	DVRØ5		(user terminal 1)
6	16	DVRØ5		(user terminal 2)
7	17	DVR23		(mag tape)

SUBCHANNELS

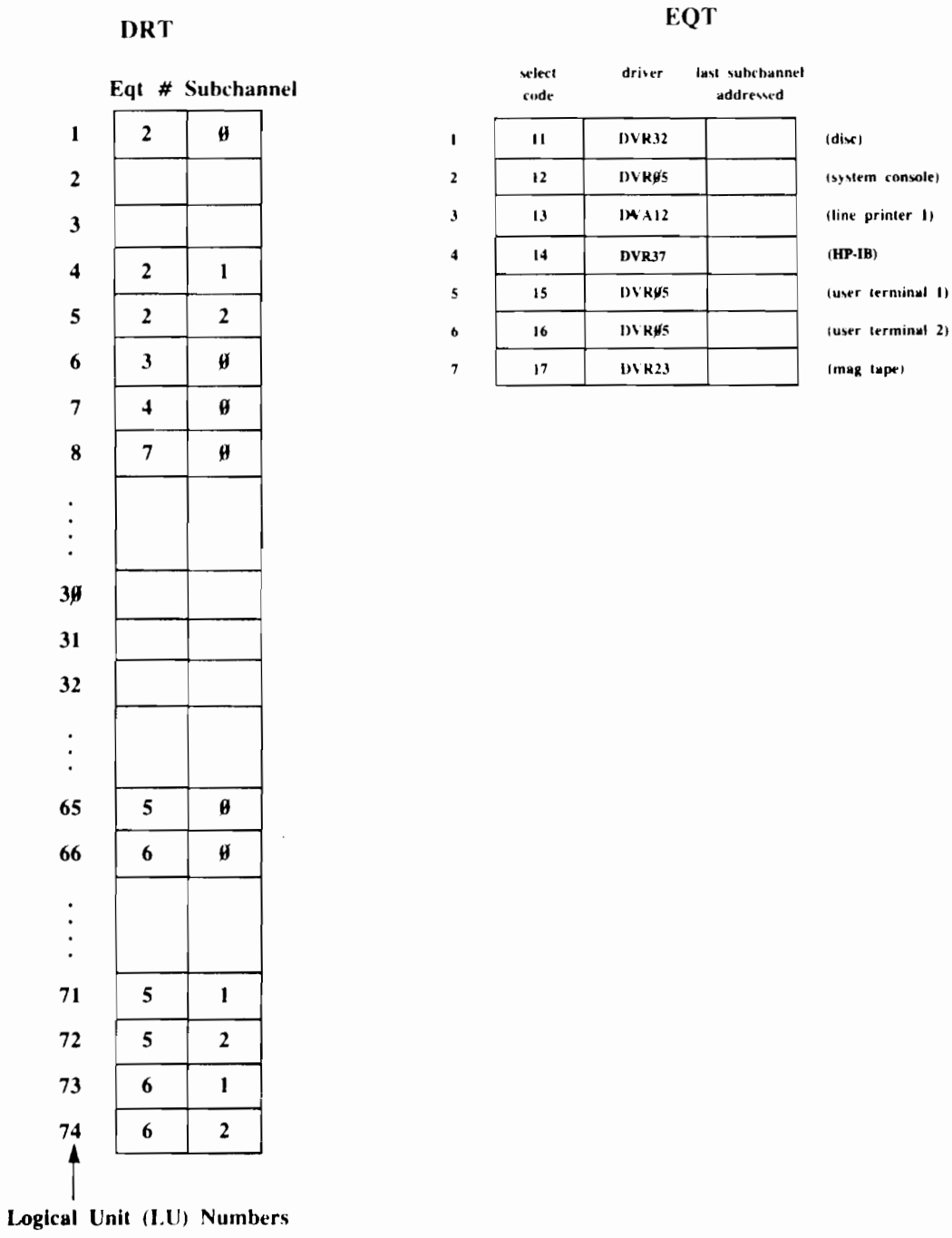
Some devices have several component parts. Each component part is identified by a SUBCHANNEL number.

for example,



SYSTEM LOGICAL UNIT (LU) NUMBERS

When the RTE system is generated, each EQT-SUBCHANNEL pair is assigned a SYSTEM LOGICAL UNIT (LU) number. This represents an entry in RTE's DEVICE REFERENCE TABLE (DRT).



SESSION LU NUMBERS

With SESSION MONITOR, users reference devices via the SESSION LU's set up when their accounts are defined.

For example, if KAREN.PROGDEV logs on at user terminal 1:

DRT

	Eqt #	Subchannel
1	2	ø
2	1	ø
3	1	4
4	2	1
5	2	2
6	3	ø
7	4	ø
8	7	ø
. . .		
30	1	1
31	1	2
32	1	3
. . .		
65	5	ø
66	6	ø
. . .		
71	5	1
72	5	2
73	6	1
74	6	2

Logical Unit (LU) Numbers

SCB

KAREN.PROGDEV
3ø

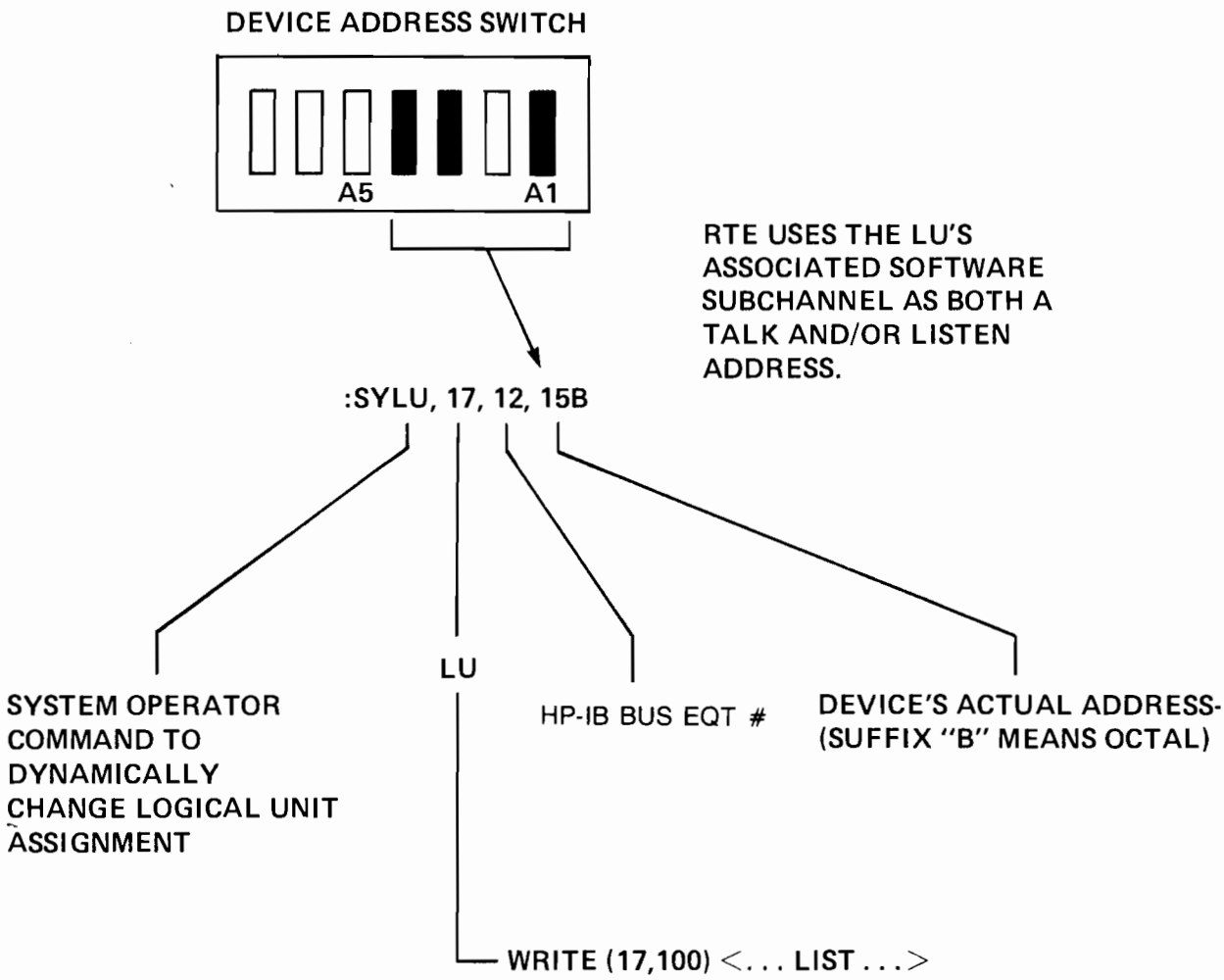
SST

System LU	Session LU
65	1
71	4
72	5
2	2
3	3
7	6
8	8

EQT

	select code	driver	last subchannel addressed	
1	11	DVR32		(disc)
2	12	DVRø5		(system console)
3	13	DVA12		(line printer 1)
4	14	DVR37		(HP-IB)
5	15	DVRø5		(user terminal 1)
6	16	DVRø5		(user terminal 2)
7	17	DVR23		(mag tape)

DEVICE ADDRESS ASSIGNMENT



The HP-IB driver constructs three different addresses from the 5 bit code on the back of each HP-IB device by adding two more bits to form a 7 bit address as shown on page 3-10. The settings of these two function bits determine whether a device is being addressed to talk or listen and whether the address is primary or secondary.

When the function bits are set to 10, the device is addressed to talk, i.e. to output data to the bus according to its primary function as described in the device manual. The 7 bit talk address used is primary to the device.

When the function bits are set at 01, the device is addressed to listen, i.e. to monitor the bus data lines for incoming data according to its primary function as described in the device manual. The 7 bit listen address used is the device's primary listen address.

Some HP-IB devices are designed so that internal registers and auxiliary I/O ports may be accessed. The bus addresses assigned to these internal locations are called secondary addresses. Secondary addressed are device dependent and are set internally. The user has no control over their assignments. Each individual device reference manual must be consulted to determine the particular device's secondary address assignments. Secondary addressing subroutines are used to read or write to/from these internal locations. The subroutines provide the correct command sequences given the device LU, data buffer, and secondary address. The HP-IB driver indicates secondary addressing to a device by setting the function bits of the device's 7 bit address to 11.

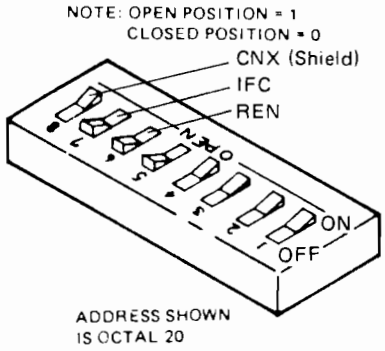
There are 31 possible primary addresses, 29 of which are available to the user to assign to HP-IB devices as desired. Address 0B is reserved for the HP-IB interface card in RTE-IV systems and address 37B is used as untalk/unlisten. In RTE-L, 36B is reserved for the interface card instead of 0B. Each primary addressed device may have up to 31 secondary addresses associated with it.

The 5 bit code used by the driver to construct addresses corresponds to subchannels of the bus EQT. for instance, a device with a 5 bit code equal to 3 set on the dip switches on its back panel is assigned the lu that corresponds to subchannel 3 of the bus EQT. LU's may be assigned as bus subchannels either at generation time or on-line using FMGR commands. For session monitor users the FMGR commands require a capability level of 60.

A few HP-IB devices are multi-addressable. They have only 4 dip switches on their back panels with the least significant bit being a "don't care". This allows one code to be used construct 2 talk and 2 listen addresses which also allows the device to be accessed via 2 LU's.

The AN401 Series Application Note package contains information on the appropriate FMGR commands to use when assigning LU's.

SETTING TALK AND LISTEN ADDRESSES



Bit Position:	b7	b6	b5	b4	b3	b2	b1
Talk Address:	*1*	*0*	A5	A4	A3	A2	A1
Listen Address:	*0*	*1*	A5	A4	A3	A2	A1
Universal Command	*0*	*0*					
Secondary Address	*1*	*1*					

Allowable Address Codes

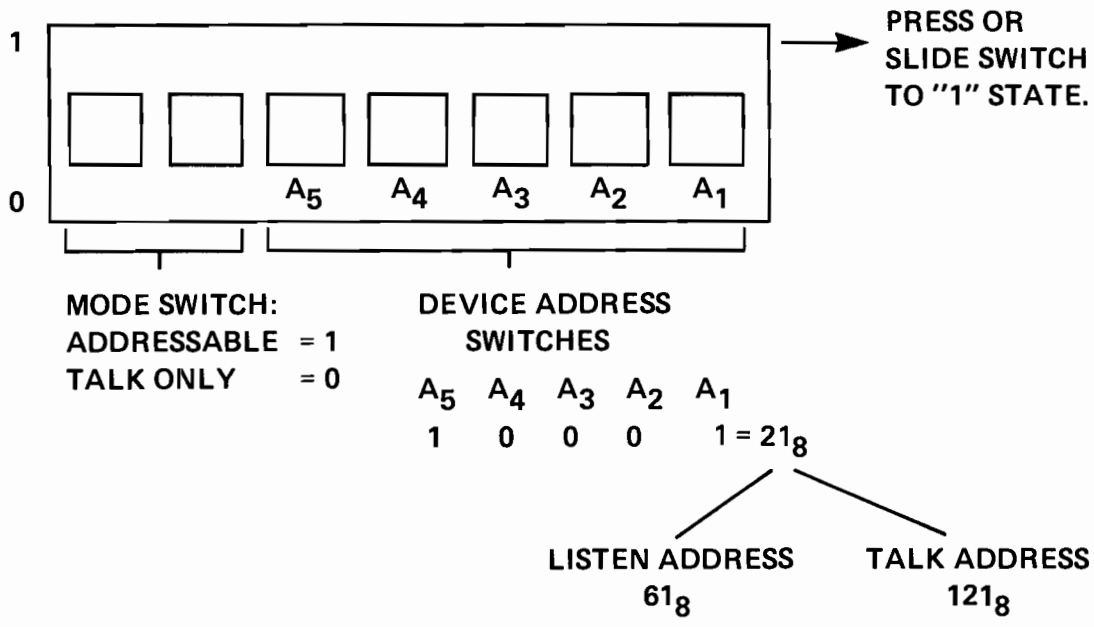
ADDRESS SWITCHES					TALK ADDRESS CHARACTER	LISTEN ADDRESS CHARACTER	OCTAL
A_5	A_4	A_3	A_2	A_1			
0	0	0	0	0	@	SP	0
0	0	0	0	1	A	!	1
0	0	0	1	0	B	"	2
0	0	0	1	1	C	#	3
0	0	1	0	0	D	$	4
0	0	1	0	1	E	%	5
0	0	1	1	0	F	&	6
0	0	1	1	1	G	'	7
0	1	0	0	0	H	(	10
0	1	0	0	1	I	)	11
0	1	0	1	0	J	*	12
0	1	0	1	1	K	+	13
0	1	1	0	0	L	,	14
0	1	1	0	1	M	-	15
0	1	1	1	0	N	.	16
0	1	1	1	1	O	/	17
1	0	0	0	0	P	0	20
1	0	0	0	1	Q	1	21
1	0	0	1	0	R	2	22
1	0	0	1	1	S	3	23
1	0	1	0	0	T	4	24
1	0	1	0	1	U	5	25
1	0	1	1	0	V	6	26
1	0	1	1	1	W	7	27
1	1	0	0	0	X	8	30
1	1	0	0	1	Y	9	31
1	1	0	1	0	Z	:	32
1	1	0	1	1	[	;	33
1	1	1	0	0	\	<	34
1	1	1	0	1	]	=	35
1	1	1	1	0	^	>	36

Address 37 is reserved for UNLISTEN or UNTALK.

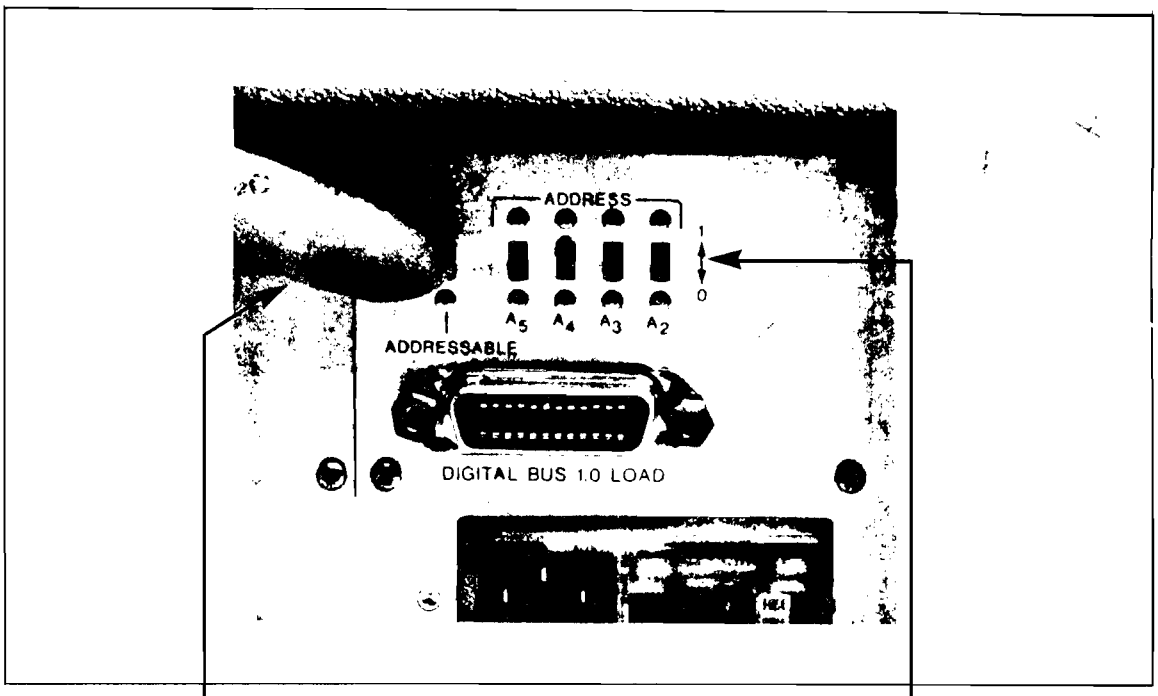
TYPICAL DEVICE ADDRESS SWITCH

HP 59309A DIGITAL CLOCK

TURN CLOCK UPSIDE DOWN AND VIEW ADDRESS SWITCH AS FOLLOWS:



TYPICAL ADDRESSABLE MODE SWITCH



IF IN "ADDRESSABLE" POSITION, THEN ANY ONE OF FOUR POSSIBLE ADDRESSES CAN BE USED.

MULTIPLE ADDRESS SWITCH IN WHICH ONE SETTING OF A_5-A_2 DETERMINES FOUR ADDRESSES - TWO TALK, TWO LISTEN. EXAMPLE:

A_5	A_4	A_3	A_2	
1	0	0	1	→ Listen = 62, 63
				→ Talk = 122, 123

4

REAL-TIME BASIC

LOADING A SOURCE PROGRAM INTO MEMORY

FUNCTION

TO LOAD ALL OR A PORTION OF A BASIC SOURCE PROGRAM OR A SEMI-COMPILED PROGRAM INTO MEMORY.

EXAMPLE 1 ➡ LOAD FROM PERIPHERAL DEVICE

```
> LOAD      ──────────► LOAD FROM INPUT DEVICE LU
> LOAD MT   ──────────► LOAD FROM TYPE Ø FILE
```

EXAMPLE 2 ➡ LOAD FROM DISC FILE

```
> LOAD TEST1      ──────► LOAD PROGRAM FROM FILENAME "TEST1".
> LOAD TEST3::25  ──────► LOAD PROGRAM FROM A SPECIFIC CARTRIDGE
```

EXAMPLE 3 ➡ SELECTIVE LOAD

```
> LOAD 30,300 PROG ──► SELECTIVELY LOAD STATEMENTS 30-300 FROM DISC
                       FILE
> LOAD 10,650 PR   ──► SELECTIVELY LOAD STATEMENTS 10-650 FROM TYPE
                       Ø FILE
```

RUNNING A BASIC PROGRAM

FUNCTION

TO LOAD AND/OR RUN A PROGRAM OR A PORTION OF A PROGRAM IN SOURCE OR SEMI-COMPILED FORM.

EXAMPLE 1 RUN PROGRAM IN MEMORY

```
> RUN  ──► EXECUTE EXISTING PROGRAM IN MEMORY
```

EXAMPLE 2 LOAD PROGRAM FROM DISC AND RUN

```
> RUN JULIAN  ──► LOADS JULIAN (IF NOT ALREADY IN MEMORY) AND
                  EXECUTES THE PROGRAM
```

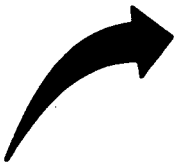
EXAMPLE 3 SELECTIVELY EXECUTE PROGRAM

```
> RUN 100,710 PROC2::35  ──► EXECUTES STATEMENTS 100-710
                  ▲
CAN BE A TYPE O FILE NAME
```

SAVING A SOURCE PROGRAM

FUNCTION

TO SAVE THE CURRENT PROGRAM (OR PORTION OF THE PROGRAM) IN MEMORY IN SOURCE FORM.



EXAMPLE 1 SAVE PROGRAM IN A DISC FILE

> SAVE FIL15:-15:110 ⟶ SAVE CURRENT PROGRAM IN MEMORY IN FILE "FIL15"

EXAMPLE 2 SAVE PORTION OF PROGRAM IN DISC FILE

> SAVE 15,230 SAM:AL ⟶ SAVE LINES 15-230 IN FILE "SAM"

EXAMPLE 3 SAVE PROGRAM ON PERIPHERAL DEVICE

> SAVE ⟶ SAVE PROGRAM ON ASSIGNED OUTPUT DEVICE LU
> SAVE MT ⟶ SAVE PROGRAM ON TYPE 0 FILE NAME

SAVE ~ the same as EC, &NAME

LISTING A SOURCE PROGRAM

FUNCTION

TO LIST ALL OR A PORTION OF A SOURCE PROGRAM IN MEMORY.

✓ EXAMPLE 1

LIST SOURCE PROGRAM

```
> LIST        ──────►  LIST CURRENT PROGRAM IN MEMORY ON TERMINAL LU.
> LIST 10,25  ──────►  LIST STATEMENTS 10-25.
```

✓ EXAMPLE 2

LIST SOURCE PROGRAM TO DISC FILE

```
> LIST LIST18::102  ──────►  LIST SOURCE PROGRAM TO FILE "LIST18"
```

✓ EXAMPLE 3

LIST TO TYPE 0 FILE

```
> LIST TTY7  ──────►  LIST PROGRAM TO TYPE 0 FILE "TTY7"
```

SAMPLE TERMINAL SESSION

FACTORIAL OF A NUMBER

THE FACTORIAL OF A NUMBER, SAY 6, IS DEFINED TO BE 1 x 2 x 3 x 4 x 5 x 6 = 720

BASIC SOLUTION

```
*RU,BASIC
BASIC READY
> LOAD FACT::- 13
BASIC READY
> LIST
   10   INPUT N
   20   IF N<0 GOTO 130
   30   IF N=0 THEN 110
   40   LET M=1
   50      FOR I=1 TO N
   60      LET L=I*M
   70      LET M=L
   80      NEXT I
   90   PRINT "FACTORIAL OF "N;"= ";M
  100      GOTO 10
  110      LET M=1
  120      GOTO 90
  130      END
> RUN
?3
FACTORIAL OF 3     =  6
?6
FACTORIAL OF 6     =  720
?8
FACTORIAL OF 8     =  40320.
?25
FACTORIAL OF 25    =  1.55112E+25
?35
FACTORIAL OF 35    =  1.70141E+38
?45
FACTORIAL OF 45    =  1.70141E+38
?- 1
BASIC READY
> BYE
```

LOAD PROGRAM FROM DISC.

LIST ON LU=1. BRANCH TABLE IS LOADED IF PROGRAM NOT SOURCE.

FOR AND NEXT LOOPS ARE INDENTED ON PRINTOUT.

EXECUTE PROGRAM

RUN-TIME RESULTS

DELETE AND REPLACE PROGRAM

DELETE

DELETE ALL OR A PORTION OF THE CURRENT PROGRAM IN MEMORY

EXAMPLES ⇨

```
> DEL 5,20    ──────►  DELETE LINES 5-20.
> DELETE 20   ──────►  DELETE LINE 20.
> 150         ──────►  DELETE LINE 150.
> DEL         ──────►  DELETE ENTIRE PROGRAM.
```

REPLACE

REPLACE THE NAMED FILE WITH ALL OR A PORTION OF THE CURRENT PROGRAM IN MEMORY.

EXAMPLES ⇨

```
> REPLACE                  ──────►  SAME AS "SAVE"
> REPLACE 10,500  ALPHA : 35 : -13
```

REPLACE FILE "ALPHA"
WITH LINES 10-500
OF PROGRAM IN
MEMORY

SOURCE STATEMENT EDITING

PURPOSE

ALLOWS ON-LINE CHARACTER EDITING OF BASIC SOURCE PROGRAM STATEMENTS. ALL EDIT CHARACTERS AND CONVENTIONS ARE THE SAME AS THE SYSTEM EDITOR.

EDIT CHARACTER	FUNCTION
I^C	INSERT NEW CHARACTER(S).
R^C	REPLACE EXISTING CHARACTER(S).
C^C	DELETE EXISTING CHARACTER(S).
T^C	TRUNCATE REMAINING CHARACTER(S).
/	DISPLAY NEXT STATEMENT.
/n	POSITION AND DISPLAY STATEMENT NUMBER n.
XI	CHANGE CONTROL CHARACTER TO I. (DEFAULT IS THE "/")
–	DELETE CURRENT STATEMENT.

USAGE

AN EDIT COMMAND IS RECOGNIZED BY THE FIRST CHARACTER BEING NON-NUMERIC.

EXAMPLE

```
>P//////A//*////)
```

EDIT PENDING STATEMENT

THE "/" PRESERVES EXISTING TEXT

MODIFY EXISTING TEXT

OBTAINING THE STATEMENT TO BE EDITED

PURPOSE

TO DISPLAY THE DESIRED STATEMENT TO BE EDITED, MAKING IT THE PENDING STATEMENT.

METHOD 1 USING THE "LIST" COMMAND.

```
> LIST 50,50          LIST STATEMENT 50.
  50 LET X=K-P+5.7
>
```

METHOD 2 USING THE "/n" EDIT COMMAND.

```
>50
  50 LET K=K-P+5.7
>
```

CHARACTER EDITING

EXAMPLE 1 → CORRECT AN ENTRY ERROR.

```
> 10        AB*10
MISSING ASSIGNMENT OPERATOR IN LINE    10
> P
    10      AB*10
MISSING ASSIGNMENT OPERATOR IN LINE    10
>P/////// I^C=
  10        A=B*10
>
```

EXAMPLE 2 → CORRECT STATEMENT ERROR UPON PROGRAM LOAD.

```
  > LOAD PROGX
UNDECIPHERABLE OPERAND IN LINE    200
    200     LET    A=-
  > P/////////// .047
  > MERGE PROGA          → MERGE REST OF PROGRAM
                           STARTING AT NEXT LINE #
  BASIC READY
  >
```

BRANCH AND MNEMONIC TABLES

FUNCTION

The Branch and Mnemonic tables provide the necessary links between subroutines in memory and the BASIC subroutine call. If you plan to use any subroutines or functions, then these tables must be loaded prior to entering BASIC statements.

To load Branch & Mnemonic tables:

>TABLES BTBL, MTBL

Where BTBL and MTBL are the names of the Branch & Mnemonic tables specified inthe table generator answer file.

See Appendix A.

5

PROGRAMMING THE BUS

PART I
DATA AND DEVICE CONTROL MESSAGES

HP-IB MESSAGE SUBROUTINES

A SET OF USER-CALLABLE SUBROUTINES THAT PROVIDE THE FOLLOWING HP-IB CAPABILITIES.

- DEVICE COMMUNICATION
- DEVICE CONTROL
- INTERRUPT AND STATUS
- SYSTEM CONTROL*
- BAIL OUT

**NOTE: PASS CONTROL IS AN HP-IB SYSTEM CONTROL CAPABILITY, BUT IS NOT SUPPORTED BY DVR37*

HP-IB MESSAGE SUBROUTINES

A SET OF USER-CALLABLE SUBROUTINES THAT PROVIDE THE FOLLOWING HP-IB CAPABILITIES.

- DEVICE COMMUNICATION
- DEVICE CONTROL
- INTERRUPT AND STATUS
- SYSTEM CONTROL*
- BAIL OUT

**NOTE:* *PASS CONTROL IS AN HP-IB SYSTEM CONTROL CAPABILITY, BUT IS NOT SUPPORTED BY DVR37*

HP-IB MESSAGE CATEGORIES AND FUNCTIONS

Category	Message	Message Function
Device Communication	Data	Transfers device dependent programming command strings and data between a talker and one or more listeners.
Device Control	Clear	Initializes one or more devices to the device dependent reset state.
	Remote	Enables remote mode for one or more devices, allowing devices to communicate with the bus.
	Trigger	Signals one or more devices to execute a device dependent action.
	Local	Enables local mode for one or more devices, disabling remote mode, allowing local panel control of the devices.
	Local Lockout	Enables device remote mode and prevents local panel control of device functions.
	Clear Lockout	Returns devices to local mode from either remote mode or local lockout mode.
Interrupt and Device Status	Require Service	One or more devices need interaction with the controller.
	Status Byte	Transfer a byte of status information to a listener. One bit indicates whether or not the talker is currently sending the require service message. The other seven indicate device dependent status.
	Status Bit	A single bit returned by a device or a group of devices as part of a composite byte indicating device status.
Passing Control	Pass Control	Pass bus controller function to a device that can control the bus. The HP1000 System does not support pass control to other bus devices.
Bail Out	Abort	Clear all bus activity for controller instructions.

DEVICE COMMUNICATION MESSAGES:

THE DATA MESSAGE

AUTO ADDRESSING:

STATEMENT	FUNCTION
WRITE	WRITE DATA TO THE BUS FROM **FORTRAN** PROGRAM.
PRINT	WRITE DATA TO THE BUS FROM **BASIC** PROGRAM.
READ	READ DATA FROM THE BUS FROM **FORTRAN** OR **BASIC** PROGRAM.

DIRECT ADDRESSING:

SUBROUTINE	FUNCTION
CMDW	WRITE COMMANDS AND DATA TO THE BUS.
CMDR	WRITE COMMANDS AND READ DATA FROM THE BUS.

SECONDARY ADDRESSING:

SECONDARY ADDRESSING ROUTINES ALLOW ACCESS TO REGISTERS, I/O PORTS, ETC. INTERNAL TO A DEVICE IF SECONDARY ADDRESSING IS IMPLEMENTED IN THE DEVICE DESIGN.

SUBROUTINE	FUNCTION
SECW	SECONDARY WRITE FOR FORTRAN PROGRAMS OR SECONDARY WRITE OF INTEGER DATA FROM BASIC
SECWR	SECONDARY WRITE OF REAL DATA FROM BASIC
SECR	SECONDARY READ FOR FORTRAN PROGRAMS OR SECONDARY READ OF INTEGER DATA FROM BASIC
SECRR	SECONDARY READ OF REAL DATA FROM BASIC

AUTO-ADDRESSING MODE

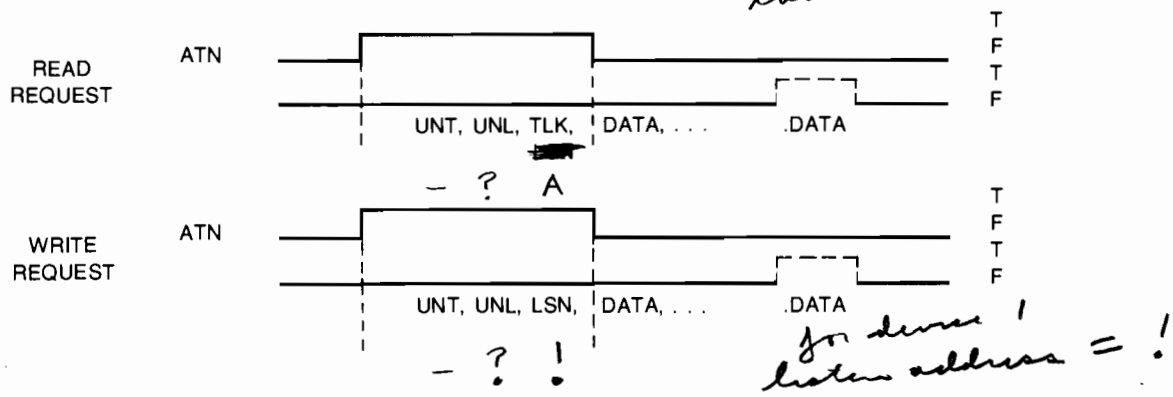
PURPOSE

TO PROVIDE A METHOD OF DEVICE COMMUNICATION ON THE BUS IN WHICH THE BUS PROTOCOL IS MADE TRANSPARENT TO THE USER PROGRAM.

AUTO ADDRESSING: HP-IB LINE PROTOCOL SUPPLIED BY DRIVER DVR37

SRQ - MAY OCCUR AT ANY TIME; HANDLED ACCORDING TO DEVICE CONFIGURATION WORD.

REN - UNAFFECTED



NOTE:

T.F. — LOGICAL STATE OF LINE: T = LOW, F = HIGH

DIRECT I/O PROGRAMMING

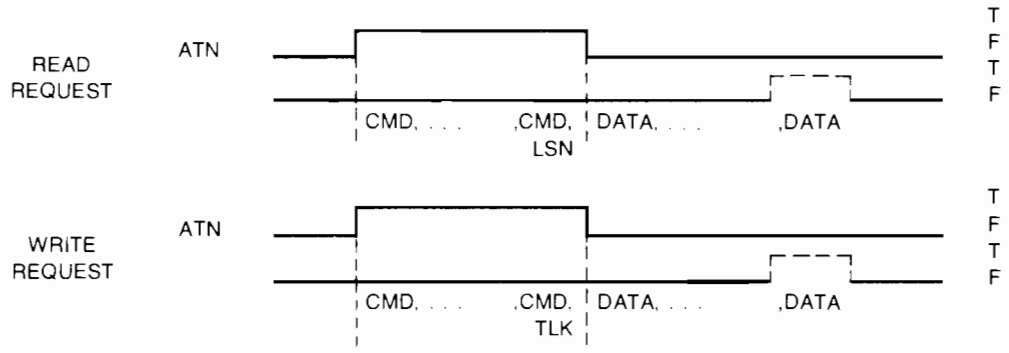
USER PROGRAM RESPONSIBLE FOR ALL BUS PROTOCOL

PURPOSE:

TO PROVIDE THE USER DIRECT ACCESS TO THE HP-IB, PERMITTING SPECIAL BUS AND DEVICE CONTROL. USER CAN USE DIRECT ADDRESSING WHEN HE WANTS TO SET UP MULTIPLE LISTENER ON THE BUS.

DIRECT I/O: HP-IB LINE PROTOCOL SUPPLIED BY USER

SRQ - MAY OCCUR AT ANY TIME; HANDLED ACCORDING TO DEVICE CONFIGURATION WORD.



NOTE:

T.F. — LOGICAL STATE OF LINE: T = LOW, F = HIGH

DEVICE COMMUNICATION

AUTO ADDRESSING

PURPOSE: TO SEND/RECEIVE DATA MESSAGES TO/FROM A DEVICE.

TO RECEIVE A DATA MESSAGE FROM A DEVICE:

20 = LU # of device

BASIC	FORTRAN
50 L = 20	L = 20
100 READ #L; V1	READ (L, *) V1
OR	OR
10 DIM A$(10)	L = 20
50 L = 20	READ (L, 100) V1
100 READ #L; A$	100 FORMAT (E10.4)

FOR AUTO-ADDRESSING, THE FOLLOWING BUS TRAFFIC ALWAYS OCCURS:

- __(UNTALK)
- ? (UNLISTEN)
- (TALK ADDRESS OF DEVICE) WHICH WILL CAUSE THE DEVICE TO TRANSMIT DATA.

SENDING DATA MESSAGES

TO SEND A DATA MESSAGE TO A DEVICE:

BASIC

```
 50 L = 25
100 PRINT #L; "F1R1A0"
```

OR

```
 10 DIM A$(6)
 20 A$ = "F1R1A0"
 50 L =25
100 PRINT #L; A$
```

FORTRAN

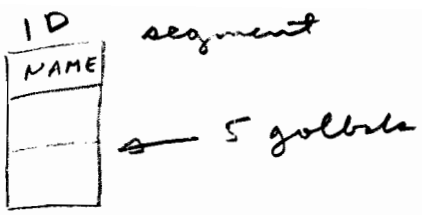
```
      L = 25
      WRITE (L, 100)
100 FORMAT ("F1R1A0")
```

FOR AUTO-ADDRESSING, THE FOLLOWING BUS TRAFFIC ALWAYS OCCURS:

— __(UNTALK)

— ? (UNLISTEN)

— (LISTEN ADDRESS OF DEVICE) AFTER WHICH THE CONTROLLER SENDS DATA



FORTRAN INPUT/OUTPUT EXAMPLE

PROBLEM

WRITE A PROGRAM TO DISPLAY THE CURRENT TIME FROM THE DIGITAL CLOCK IN THE DISPLAY AND STOP AT THE END OF THE CURRENT MINUTE PERIOD.

SOLUTION

```
0001   FTN,L
0002   C      RUN PARAMETERS ARE : CONSOLE LU.
0003   C
0004          PROGRAM BUS02
0005          INTEGER TIME(6),INITL(6),IPRAM(5)
0006          CALL RMPAR(IPRAM)
0007          LUTTY=1
0008          IF(IPRAM(1).NE.0) LUTTY=IPRAM(1)
0009          WRITE(LUTTY,100)
0010   100    FORMAT("ENTER : CLOCK LU, DISPLAY LU _")
0011          READ(LUTTY,*)ICLU,IDLU
0012          READ(ICLU,101)INITL
0013   101    FORMAT(6A2)
0014   10     READ(ICLU,101)TIME
0015          IF(TIME(6).EQ.2H00) GO TO 99
0016   30     WRITE(IDLU,101)TIME
0017          GOTO 10
0018   99     WRITE(IDLU,101) TIME
0019          WRITE(LUTTY,102)
0020   102    FORMAT("END OF CURRENT MINUTE")
0021          END


**  NO ERRORS**     PROGRAM = 00160      COMMON = 00000
```

RMPAR (IPRAM) remembers parameters from run statements gives 5 words

if nothing is put in the run statement then RMPAR(1) is global 1 (the LU that the program is from)

BASIC INPUT/OUTPUT EXAMPLE

PROBLEM: DISPLAY DIGITAL CLOCK TIME ON DISPLAY UNTIL STOP TIME REACHED.

```
10  DIM A$(10), B$(10)
20  PRINT "ENTER: STOP TIME";
30  INPUT A$
40  READ #14; B$
50  IF B$ = A$ GO TO 80
60  PRINT #15; B$
70  GO TO 40
80  PRINT "STOP TIME REACHED"
90  END
```

14 — CLOCK LOGICAL UNIT #

15 — DISPLAY LOGICAL UNIT #

BASIC DEVICE PROGRAM EXAMPLE

PROBLEM

PROGRAM A DIGITAL MULTIMETER (HP 3455A) TO THE 1 VOLT RANGE, AC FUNCTION AND TAKE ONE READING

DVM LU = 11

SOLUTION:

```
10  DIM A$(8)
20  A$ = "F2R2H1T3"
30  PRINT #11; A$
40  PRINT #11; "T3"
50  READ #11; R1
60  END
```

This ASCII string is sent as a command sequence to the multimeter while bus is in data mode

F — SELECT FUNCTION
2 — SELECT AC MEASUREMENT
R — RANGE
2 — 1 VOLT
H — HIGH RESOLUTION
1 — ON
T — TRIGGER SOURCE
3 — HOLD/MANUAL

REFER TO DEVICE PROGRAMMING REFERENCE SHEETS, INDIVIDUAL DEVICE REFERENCE MANUALS, OR AN-401 SERIES APPLICATION NOTES FOR DEVICE PROGRAMMING CODES.

DIRECT I/O
CMDR, CMDW SUBROUTINE

PURPOSE: ALLOW BASIC AND FORTRAN PROGRAMS TO SEND CHARACTERS OVER THE BUS IN COMMAND OR DATA MODE AND RECEIVE CHARACTERS IN DATA MODE.

TO READ BUS DATA:

BASIC	**FORTRAN**
CMDR(A,B$,C$)	CALL CMDR(IA,IB,IC)

TO WRITE BUS DATA:

BASIC	**FORTRAN**
CMDW(A,B$,C$)	CALL CMDW(IA,IB,IC)

A,IA — BUS LU#.
B$, IB — COMMAND BUFFER
C$,IC — DATA BUFFER

NOTE: IN FORTRAN THE FIRST WORD OF ARRAYS IB AND IC MUST CONTAIN THE NUMBER OF CHARACTERS IN THE ARRAY.

FIVE WAYS TO USE THE CMDR AND CMDW SUBROUTINES:

TYPE OF I/O	CALL FROM USER PROGRAM
COMMAND OUTPUT ONLY	CMDW(BUSLU,CMND,0) CMDR(BUSLU,CMND,0)
COMMAND OUTPUT, DATA INPUT	CMDR(BUSLU,CMND,DATA)
COMMAND OUTPUT, DATA OUTPUT	CMDW(BUSLU,CMND,DATA)
DATA INPUT ONLY	CMDR(BUSLU,0,DATA)
DATA OUTPUT ONLY	CMDW(BUSLU,0,DATA)

BUSLU = LOGICAL UNIT # OF BUS

CMND = COMMAND BUFFER SPECIFIED IN USER PROGRAM

DATA = DATA BUFFER SPECIFIED IN USER PROGRAM

```
10  REM  *** COMMAND OUTPUT ONLY *****
20  REM
30  DIM C$[20],D$[20]
40  LET B1=17
50  REM  *** UNTLK, UNLSN, CLOCK TALK/DISPLAY LISTEN
60  LET C$="_?Q&"
70  CALL CMDW(B1,C$,0)
80    FOR I=1 TO 5000
90    NEXT I
100  REM  *** UNTLK,UNLSN
110  LET C$="_?"
120  CALL CMDW(B1,C$,0)
130  END

10  REM  *** COMMAND OUTPUT, DATA INPUT
20  REM
30  DIM C$[20],D$[20]
40  LET D$="00000000000000000000"
50  LET B1=17
60  REM  *** UNTLK, UNLSN, CLOCK TALK
70  LET C$="_?Q"
80  CALL CMDR(B1,C$,D$)
90  PRINT
100  PRINT D$
110  END

10  REM  *** COMMAND OUTPUT, DATA OUTPUT
20  REM
30  DIM C$[20],D$[20]
40  LET B1=17
50  REM  *** UNTLK, UNLSN, CLOCK LISTEN
60  LET C$="_?1"
70  LET D$="PRDDDT"
80  CALL CMDW(B1,C$,D$)
90  END

10  REM  *** DATA INPUT ONLY
20  REM
30  DIM C$[20],D$[20]
40  LET C$="00000000000000000000"
50  LET D$=C$
60  LET B1=17
70  REM  *** UNTLK, UNLSN, CLOCK TALK
80  LET C$="_?Q"
90  CALL CMDW(B1,C$,0)
100  REM
110  CALL CMDR(B1,0,D$)
120  PRINT
130  PRINT D$
140  END

10  REM  *** DATA OUTPUT ONLY
20  REM
30  DIM C$[20],D$[20]
40  LET B1=17
50  REM  *** UNTLK, UNLSN, DISPLAY LISTEN
60  LET C$="_?&"
70  CALL CMDW(B1,C$,0)
80  REM
90  LET D$="12345678 E-1"
100  CALL CMDW(B1,0,D$)
110  END
```

```
0001  FTN,L
0002  C      COMMAND OUTPUT ONLY
0003  C
0004         PROGRAM COO
0005         DIMENSION ICMND(20)
0006         INTEGER BLU
0007         BLU=17
0008         ICMND(1)=4
0009  C  UNTLK, UNLSN, CLOCK TALK/DISPLAY LISTEN
0010         ICMND(2)=2H_?
0011         ICMND(3)=2HQ&
0012         CALL CMDW(BLU,ICMND,0)
0013         DO 10 IX=1,10
0014         DO 10 I=1,32767
0015   10    CONTINUE
0016         ICMND(1)=2
0017         ICMND(2)=2H_?
0018         CALL CMDW(BLU,ICMND,0)
0019         END
```

```
**  NO ERRORS**     PROGRAM = 00109       COMMON = 00000
```

```
0001  FTN,L
0002  C      COMMAND OUTPUT, DATA INPUT
0003  C
0004         PROGRAM CODI
0005         DIMENSION ICMND(20),IDATA(20),ITEMP(6)
0006         EQUIVALENCE (IDATA(2),ITEMP)
0007         INTEGER BLU
0008         BLU=17
0009         ICMND(1)=3
0010  C  UNTLK, UNLSN, CLOCK TALK
0011         ICMND(2)=2H_?
0012         ICMND(3)=2HQ
0013         IDATA(1)=12
0014         CALL CMDR(BLU,ICMND,IDATA)
0015  C
0016  C  IDATA(1) WILL STILL CONTAIN THE #OF CHARACTERS AFTER THE CALL
0017  C
0018         WRITE(1,101) ITEMP
0019  101    FORMAT(6A2,/)
0020         END
```

```
**  NO ERRORS**     PROGRAM = 00110       COMMON = 00000
```

```
        PAGE  0001          FTN4 COMPILER: HP24177  (SEPT. 1974)

0001  FTN,L
0002  C      COMMAND OUTPUT, DATA OUTPUT
0003  C
0004         PROGRAM CODO
0005         DIMENSION ICMND(20),IDATA(20)
0006         INTEGER BLU
0007         BLU=17
0008         ICMND(1)=1
0009  C   COMMAND CLOCK TO LISTEN
0010         ICMND(2)=2H1
0011         IDATA(1)=6
0012  C   STOP, RESET, ADVANCE DAYS BY 3, START
0013         IDATA(2)=2HPR
0014         IDATA(3)=2HDD
0015         IDATA(4)=2HDT
0016         CALL CMDW(BLU,ICMND,IDATA)
0017         END


**  NO ERRORS**     PROGRAM = 00109       COMMON = 00000



0001  FTN,L
0002  C      DATA INPUT ONLY
0003  C
0004         PROGRAM DIO
0005         DIMENSION ICMND(20),IDATA(20),ITEMP(12)
0006         INTEGER BLU
0007         EQUIVALENCE (IDATA(2),ITEMP)
0008         BLU=17
0009         ICMND(1)=3
0010  C   UNTLK, UNLSN, CLOCK TALK
0011         ICMND(2)=2H_?
0012         ICMND(3)=2HQ
0013         CALL CMDW(BLU,ICMND,0)
0014  C
0015         IDATA(1)=12
0016         CALL CMDR(BLU,0,IDATA)
0017         WRITE(1,100) ITEMP
0018  100    FORMAT(6A2,/)
0019         END


**  NO ERRORS**     PROGRAM = 00116       COMMON = 00000
```

```
0001  FTN,L
0002  C      DATA OUTPUT ONLY
0003  C
0004         PROGRAM DOO
0005         DIMENSION ICMND(20),IDATA(20),ITEMP(12)
0006         INTEGER BLU
0007         EQUIVALENCE (IDATA(2),ITEMP)
0008         BLU=17
0009         ICMND(1)=3
0010  C  UNTLK, UNLSN, DISPLAY LISTEN
0011         ICMND(2)=2H_?
0012         ICMND(3)=2H&
0013         CALL CMDW(BLU,ICMND,0)
0014  C
0015         IDATA(1)=12
0016         IDATA(2)=2H12
0017         IDATA(3)=2H34
0018         IDATA(4)=2H56
0019         IDATA(5)=2H78
0020         IDATA(6)=2H E
0021         IDATA(7)=2H-1
0022         CALL CMDW(BLU,0,IDATA)
0023         END
```

** NO ERRORS** PROGRAM = 00147 COMMON = 00000

SECONDARY ADDRESSING SECR,SECRR,SECW,SECWR SUBROUTINES

PURPOSE: ALLOW BASIC AND FORTRAN PROGRAMS TO SEND DATA TO AND RECEIVE DATA FROM SECONDARY ADDRESS LOCATIONS OF BUS DEVICES.

TO READ DATA:

BASIC

SECR(D,S,C,L) READS INTEGER DATA

SECRR(D,S,C,L) READS REAL DATA

FORTRAN

CALL SECR(ID,IS,IDAT,IL)

TO WRITE DATA:

BASIC

SECW(D,S,C,L) WRITES INTEGER DATA

SECWR(D,S,C,L) WRITES REAL DATA

FORTRAN

CALL SECW(ID,IS,IDAT,IL)

D,ID — DEVICE LOGICAL UNIT NUMBER

S,IS — SECONDARY ADDRESS TO BE ACCESSED (0-30)

C,IDAT — ARRAY TO CONTAIN INPUT DATA OR DATA TO BE WRITTEN TO SECONDARY LOCATION

L,IL — LENGTH OF DATA TO BE TRANSMITTED, POSITIVE WORDS OR NEGATIVE BYTES

NOTE: IN FORTRAN, THE FIRST WORD OF THE IDAT ARRAY MUST CONTAIN THE NUMBER OF CHARACTERS IN THE ARRAY.

THE FOLLOWING BUS TRAFFIC OCCURS FOR SECONDARY ADDRESSING:

- _? UNTALK, UNLISTEN
- BUS LISTEN ADDRESS
- DEVICE TALK OR LISTEN ADDRESS
- SECONDARY ADDRESS

SECONDARY ADDRESSING I/O EXAMPLE USING HP 2240 MEASUREMENT & CONTROL PROCESSOR

```
FTN4,L
      PROGRAM LOG
      INTEGER CCODE,DATA(100),IREAD(101),IERR(4)
      EQUIVALENCE (CCODE,IREAD(1)),(DATA(1),IREAD(2))
      LU2240=10

C THIS IS A PROGRAM WHICH PROGRAMS THE 2240 TO WAIT FOR A DIGITAL POINT
C TO GO TRUE, THEN TAKE 100 ANALOG READINGS.
C THE PROGRAM THEN WRITES THE SET OF READINGS TO TAPE.
C
C SECONDARY ADDRESSING IS USED IN TWO WAYS
C
C 1) TO POLL THE 2240 TO SEE IF THE READINGS HAVE BEEN TAKEN YET
C    BY READING SECONDARY 1.
C
C 2) TO GET ERROR STATUS FROM THE 2240 IF A NONZERO CONDITION CODE
C    WAS RETURNED. (INDICATING AN ERROR IN TAKING THE READINGS)

      DO 25 I=1,10
      WRITE(10,1000)
C THIS STRING TELLS THE 2240 TO WAIT FOR THE DIGITAL INPUT, THEN
C TAKE 100 READINGS, ONE EVERY 100 MILLISECONDS.
1000  FORMAT(' WT,1,1,1;WB,100;RP,100;AI,1,1,1;NX!')
C
C THIS EXEC CALL CAUSES PROGRAM LOG TO SUSPEND FOR 10 SECONDS
10    CALL EXEC(12,0,2,0,-10)

C READ SECONDARY 1 FOR SUMMARY STATUS
      CALL SECR(LU2240,1,ISTAT,1)
C GO TO 10 (SUSPEND) IF RESULTS ARE NOT READY
      IF(IAND(ISTAT,4).EQ.0) GO TO 10
C RESULT READY, READ IT!   (NOTICE EQUIVALENCE STATEMENT)
      READ(LU2240,*)IREAD
      IF(CCODE.NE.0) GO TO 100
C WRITE DATA TO TAPE
       WRITE(8)DATA
C WRITE END OF FILE
25    CONTINUE
      CALL EXEC(3,9)
      STOP
C
C THIS IS THE ERROR HANDLING SECTION
C
100   CALL SECR(LU2240,2,IERR,4)
      WRITE(1,1001)IERR
1001  FORMAT(' ERROR DURING DATA SAMPLING',//,
     1 ' SUMMARY STATUS        =',I5,/,
     2 ' ERROR CODE            =',I5,/,
     3 ' CURRENT COMMAND       =',I5,/,
     4 ' ADDITIONAL ERROR CODE =',I5)
      STOP
      END
```

DEVICE CONTROL MESSAGES

Message	Subroutine	Subroutine Function
Clear	CLEAR	Issues selected device clear command SDC to one or more devices, or Issues universal device clear command DCL to all devices on the bus.
Remote	RMOTE	Sets remote enable line REN true, enabling remote operation for all devices responding to the REN control line.
Trigger	TRIGR	Issues group execute trigger command GET to one or more devices.
Local	GTL	Issues go to local command GTL to addressed devices on the bus.
Local Lockout	LLO	Issues local lockout command LLO to all devices on the bus.
Clear Lockout	LOCL	Sets remote enable line REN false, removing all devices from local lockout mode and returning them to local control.

CLEAR SUBROUTINE

PURPOSE: TO RESET/INITIALIZE DEVICES TO PREDEFINED STATE

TO CLEAR ONE DEVICE (AUTO-ADDRESSING)

BASIC	FORTRAN
100 CLEAR(D,1)	CALL CLEAR(IDLU,1)

BUS TRAFFIC: ? (UNLISTEN), LISTEN ADDR OF DEVICE, SDC (SELECTED DEVICE CLEAR)

TO CLEAR A GROUP OF DEVICES (DIRECT ADDRESSING)

BASIC	FORTRAN
10 CMDW(B,A$,0)	CALL CMDW(IBLU,IA,0)
.	.
.	.
.	.
100 CLEAR(B,1)	CALL CLEAR(IBLU,1)

BUS TRAFFIC: SDC

TO CLEAR ALL DEVICES ON BUS

BASIC	FORTRAN
100 CLEAR(B,2)	CALL CLEAR(IBLU,2)

BUS TRAFFIC: DCL (UNIVERSAL DEVICE CLEAR)

D, IDLU ARE DEVICE LU#'S — B, IBLU ARE THE BUS LU#
A$ OR IA CONTAINS LISTEN ADDRESSES OF DEVICES TO BE CLEARED

RMOTE SUBROUTINE

PURPOSE: SWITCH DEVICES FROM LOCAL FRONT PANEL CONTROL TO REMOTE CONTROL OF THE USER PROGRAM.

TO REMOTE ONE DEVICE: (AUTO-ADDRESSING)

BASIC	FORTRAN
100 RMOTE(D)	CALL RMOTE(IDLU)

BUS TRAFFIC: ASSERT REN, ? (UNL), LISTEN ADDR OF DEVICE

TO REMOTE A GROUP OF DEVICES: (DIRECT ADDRESSING)

BASIC	FORTRAN
10 CMDW(B,A$,0)	CALL CMDW(IBLU,IA,0)
.	.
.	.
.	.
100 RMOTE(B)	CALL RMOTE(IBLU)

BUS TRAFFIC: ASSERT REN

D, IDLU ARE DEVICE LU #'S — B, IBLU ARE THE BUS LU #
A$ OR IA CONTAIN LISTEN ADDRESSES OF DEVICES TO BE REMOTED.

TRIGR SUBROUTINE

PURPOSE: INITIATE A DEVICE-DEPENDENT ACTION WITHIN A DEVICE OR GROUP OF DEVICES.

TO TRIGGER ONE DEVICE (AUTO-ADDRESSING)

BASIC	FORTRAN
100 TRIGR(D)	CALL TRIGR(IDLU)

BUS TRAFFIC: ? (UNL), LISTEN ADDR OF DEVICE, GROUP EXECUTE TRIGGER (GET) COMMAND

TO TRIGGER MULTIPLE DEVICES (DIRECT ADDRESSING)

BASIC	FORTRAN
10 CMDW(B,A$,0)	CALL CMDW(IBLU,IA,0)
•	•
•	•
100 TRIGR(B)	CALL TRIGR(IBLU)

BUS TRAFFIC: GET

THIS FORM OF THE TRIGR CALL TRIGGERS ANY DEVICES PREVIOUSLY ADDRESSED TO LISTEN

D, IDLU ARE DEVICE LU#'S — B, IBLU ARE THE BUS LU#
A$ OR IA CONTAIN LISTEN ADDRESSES OF DEVICES TO BE TRIGGERED.

GTL SUBROUTINE

PURPOSE: RETURN DEVICE(S) TO LOCAL FRONT PANEL CONTROL

TO SEND GTL TO ONE DEVICE (AUTO-ADDRESSING)

BASIC	FORTRAN
100 GTL(D)	CALL GTL(IDLU)

BUS TRAFFIC: ? (UNL), LISTEN ADDR OF DEVICE, GO TO LOCAL (GTL) COMMAND

TO SEND GTL TO MULTIPLE DEVICES (DIRECT ADDRESSING)

BASIC	FORTRAN
10 CMDW(B,A$,0)	CALL CMDW(IBLU,IA,0)
.	.
.	.
.	.
100 GTL(B)	CALL GTL(IBLU)

BUS TRAFFIC: GTL COMMAND

D, IDLU ARE DEVICE LU #'S — B, IBLU ARE THE BUS LU #
A$ OR IA CONTAIN LISTEN ADDRESSES OF DEVICES TO BE LOCALED.

LLO SUBROUTINE

PURPOSE: PREVENT AN OPERATOR FROM RETURNING A DEVICE TO LOCAL FRONT PANEL CONTROL.

BASIC	FORTRAN
100 LLO(B)	CALL LLO(IBLU)

BUS TRAFFIC: LOCAL LOCKOUT (LLO)COMMAND

B, IBLU ARE THE BUS LU#

THE LLO SUBROUTINE MUST BE CALLED WITH THE BUS LU #

NOTE: AS LONG AS THE LLO MESSAGE IS IN EFFECT, NO DEVICE CAN BE RETURNED TO FRONT PANEL CONTROL EXCEPT BY SENDING LOCL MESSAGE.

LOCL SUBROUTINE

PURPOSE: CLEAR LOCAL LOCKOUT AND RETURN DEVICES TO LOCAL FRONT PANEL CONTROL.

BASIC	FORTRAN
100 LOCL(B)	CALL LOCL(IBLU)

BUS TRAFFIC: REN LINE REMOVED

B, IBLU ARE THE BUS LU#

THE LOCL SUBROUTINE MUST BE CALLED WITH THE BUS LU#.

ADDRESSING MULTIPLE DEVICES

PROBLEM:

(1) SET THE HP 3455A DVM TO TAKE A 2 WIRE RESISTANCE MEASUREMENT.

(2) ADDRESS THE VOLTMETER AS A TALKER, THE NUMERIC DISPLAY AND THERMAL PRINTER AS LISTENERS, AND ACCEPT THE RESULT FROM THE VOLTMETER AT THE CONTROLLER.

SOLUTION:

10	DIM C$(5),D$(10)	Define strings.
	CALL CLEAR	
20	RMOTE(27)	Send remote message to voltmeter.
30	PRINT #27;"F3R5M3T3"	Send command string to voltmeter.
40	TRIGR(27)	Trigger DVM.
50	LET C$ = "_?D%&"	Define string containing untalk and unlisten commands, talk address of voltmeter, listen addresses of numeric display and thermal printer.
60	CMDW(20,C$,0)	Send device addresses in command mode.
70	READ #20;V	Read voltmeter measurement.
80	END	PRINT V

- D = TALK ADDRESS OF VOLTMETER
- # = LISTEN ADDRESS OF VOLTMETER.
- % = LISTEN ADDRESS OF NUMERIC DISPLAY.
- & = LISTEN ADDRESS OF THERMAL PRINTER.
- 20 = BUS LOGICAL UNIT NUMBER.
- 27 = VOLTMETER LOGICAL UNIT NUMBER.

USING BUS MESSAGES WITH MULTIPLE DEVICES

PROBLEM: SEND THE CLEAR, REMOTE AND TRIGGER MESSAGES TO THE DVM AND THE COUNTER.

DIRECT ADDRESSING

BASIC

```
10  A$ = "___ ? DE"

20  CMDW(B,A$,0)
30  CLEAR(B,1)
40  RMOTE(B)
50  TRIGR(B)
        .
        .
        .
100 END
```

FORTRAN

```
        .
        .
        .
      DIMENSION IA(3)
      IA(1) = 4
      IA(2) = 2H___?
      IA(3) = 2HDE
      CALL CMDW(B,IA,0)
      CALL CLEAR(B,1)
      CALL RMOTE(B)
      CALL TRIGR(B)
              .
              .
              .
      END
```

AUTO ADDRESSING

```
1. FOR I = 50 TO 51
2. CLEAR (I,1)
3. RMOTE (I)

4. TRIGR (I)
5. NEXT I
        .
        .
        .
100 END
```

```
      DO 60 I = 50, 51
         CALL CLEAR (I,1)
         CALL REMOTE(I)
         CALL TRIGR (I)
60    CONTINUE
              .
              .
              .
         END
```

B — THE BUS LOGICAL UNIT #
D,E — THE LISTEN ADDRESS OF THE DVM AND THE COUNTER.
50, 51 — THE LU # OF THE DVM AND THE COUNTER.

6

PROGRAMMING THE BUS

PART II
DEVICE STATUS AND BAILOUT MESSAGES

DEVICE STATUS MESSAGES

Message	Subroutine	Subroutine Function
Require Service	SRQ	Activates or disables a service request alarm program for a device: system software handles SRQ automatically.
	SRQSN	For BASIC language programs only: activates a service request trap entry.
Status Byte	STATS	Returns a byte of status information from a specific device, or Returns a byte of status information from a specific HP-IB interface card.
Status Bit	PPOLL	Enables a device to respond to a parallel poll and configures its response, or Disables a device or a group of devices from responding to a parallel poll, or Resets all parallel poll devices to a predetermined condition.
	PSTAT	Returns up to eight bits of information from up to eight responding devices or groups that have been previously parallel poll enabled.

SRQ, SRQSN SUBROUTINES

PURPOSE: CONDUCT A "SERIAL POLL" AND BRANCH THE PROGRAM TO A SERVICE SUBROUTINE UPON SERVICE REQUEST.

TO BRANCH TO A FORTRAN PROGRAM:

BASIC	FORTRAN
SRQ(A,B,"PROGNAME")	CALL SRQ(IA,IB,IPROG)

A, IA — THE DEVICE LU #.

"PROGNAME" — THE NAME OF THE SERVICE PROGRAM.

IPROG — AN ARRAY CONTAINING THE SERVICE PROGRAM NAME.

B,IB — ARBITRARY PARAMETER TO BE PASSED TO SERVICE PROGRAM

THIS CALL SETS UP THE SERVICE PROGRAM NAME IN THE SPECIFIED DEVICE'S EQT EXTENSION (WORDS 2-4) AND SETS THE S-BIT IN WORD 2 OF THE EXTENSION AREA. THE DRIVER DOES A SERIAL POLL TO DETERMINE THE DEVICE REQUESTING SERVICE THEN RETRIEVES THE SERVICE PROGRAM NAME FROM THE DEVICE'S EQT EXTENSION AND SCHEDULES IT IF THE S-BIT IS SET.

DIMENSION IPROG(3)

DATA IPROG /3HPROGA/

SRQ, SRQSN (continued)

TO BRANCH TO A BASIC SUBROUTINE:

BASIC

100 SRQSN(A,B)

B — THE TRAP # (THE TRAP STATEMENT ASSOCIATES A PARTICULAR TRAP # WITH A BASIC SUBROUTINE)

TO DISABLE A REQUIRE SERVICE MESSAGE RESPONSE:

BASIC	FORTRAN
SRQ(A,17,0)	CALL SRQ(IA,17)

A, IA — THE DEVICE LU#.

EXAMPLE PROGRAM WITH ASSOCIATED SRQ PROGRAM

```
FTN4,L
      PROGRAM SRQA
C
C THIS PROGRAM SETS UP A TIMING RATE FOR AN HP 59308A TIMING
C GENERATOR AND SETS UP SRQ TO SCHEDULE PROGRAM "NUMER" WHENEVER
C AN INTERRUPT IS RECEIVED FROM THE GENERATOR.
C
      DIMENSION IPROG(4),IPRAM(5)
      COMMON IFLG
C
C CALL RMPAR TO ASCERTAIN THE TERMINAL BEING USED.
C
      CALL RMPAR(IPRAM)
      ILU=IPRAM(1)
      IF(ILU.EQ.0)ILU=1
      IFLG=0
C
C THE PROGRAM HAS BEEN MADE INTERACTIVE SO THAT WE MAY INPUT THE
C RATE AT WHICH THE TIMING GENERATOR IS TO GENERATE INTERRUPTS.
C
      WRITE(ILU,10)
10    FORMAT("/SRQ: ENTER INTERRUPT TIME: _")
C
C THE TIMING GENERATOR ASSUMES INTERRUPTS IN MICROSECONDS;
C FOR EXAMPLE, IF WE SPECIFY '100' THE GENERATOR WILL INTERRUPT
C EVERY SECOND.
C
      READ(ILU,*)ITM
C
C SET UP THE TIMING GENERATOR (AUTO ADDRESSED LU# 48).
C
      WRITE(48,20)ITM
20    FORMAT(I3,"E4PSR")
C
C NEXT, SET UP THE DRIVER TO SCHEDULE 'NUMER' ON INTERRUPT
C
      IPROG(1)=5
      IPROG(2)=2HNU
      IPROG(3)=2HME
      IPROG(4)=2HR
      CALL SRQ(48,16,IPROG)
C
      END
FTN4,L
      PROGRAM NUMER
C
C THIS IS THE PROGRAM THAT IS SCHEDULED ON INTERRUPT.
C
      COMMON IFLG
C
C WRITE A WORD FROM COMMON TO AUTO-ADDRESSED LU# 13,
C AN HP 59304A NUMERIC DISPLAY.
C
      WRITE(13,1)IFLG
C
C ALSO WRITE THE WORD FROM COMMON TO THE SYSTEM CONSOLE.
C
      WRITE(1,1)IFLG
C
C INCREMENT THE WORD IN COMMON
C
      IFLG=IFLG+1
1     FORMAT(I6)
      END
      END$
```

THE SERVICE PROGRAM CAN OBTAIN THREE PARAMETEWRS BY CALLING "RMPAR", AS FOLLOWS:

```
FTN4,L
      PROGRAM EVENT
      DIMENSION IPRAM(5)        Allocate five-word array.

CALL  RMPAR(IPRAM)              The call to RMPAR must be the
            .                   first executable statement in
            .                   the program.
            .                   IPRAM(1) = status byte
            .                   IPRAM(2) = device 5 bit address code
                                           ( subchannel number )
      remainder of program      IPRAM(3)= equipment table address
                                           ( EQT )
                                IPRAM(4) = arbitrary parameter passed
                                           to service program through
                                           SRQ routine
```

CA

EXAMPLE OF USING THE SRQSN AND THE TRAP STATEMENT:

```
20   A=10
30   SRQSN(A,1)
       .
       .
       .
100  TRAP 1 GOSUB 900
       .
900  PRINT "SRQ ON DEVICE LU # ",A," "
901  RETURN
```

This associates the device to trap number 1.

Trap number 1 will cause a branch to statement 900.

STATS SUBROUTINE

PURPOSE: OBTAIN A STATUS BYTE FROM A DEVICE, OR A STATUS BYTE FROM THE HP-IB INTERFACE CARD.

TO OBTAIN DEVICE STATUS

BASIC	FORTRAN
100 STATS(D,S)	CALL STATS(IDLU,ISTAT)

BUS TRAFFIC: _ (UNT), ? (UNL), SERIAL POLL ENABLE (SPE) COMMAND, LISTEN ADDR OF DEVICE, SERIAL POLL DISABLE (SPD) COMMAND

D, IDLU ARE THE DEVICE LU #.

THE STATUS WORD RETURNED BY THE DEVICE IS STORED IN S OR ISTAT AND HAS THE FOLLOWING FORMAT:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	X	S	X	X	X	X	X	X

S = SERVICE REQUEST (1 = YES/0 = NO)

X REPRESENTS DEVICE DEPENDENT INFORMATION

REFER TO THE SPECIFIC DEVICE REFERENCE MANUAL FOR DEVICE DEPENDENT STATUS CODES.

THE STATUS ROUTINE RETRIEVES A BYTE OF STATS FROM WORD 5 OF THE EQT EXTENSION FIXED AREA.

SEE CHAPTER 9 FOR EQT AND EQT EXTENSION ENTRY FORMATS.

READING THE STATUS BYTE IN A FORTRAN PROGRAM

```
FTN4,L
      PROGRAM TOTAL, 3
      DIMENSION A(10)

      IDVM=20
      ILU=1

      CALL RMOTE(IDVM)                    Remote enable the DVM.
      WRITE(IDVM,20)                      Program the DVM.
20    FORMAT("F1R1M3A0D0H0T3")

      DO 24 I=1,10                        Take ten measurements.

      WRITE(IDVM,21)                      Trigger the DVM.
21    FORMAT("T3")

      CALL STATS(IDVM,ISTAT)              Get current status of DVM.
      IF(ISTAT.NE.0) GO TO 26             Status okay?

      READ(IDVM,23) A(I)                  Return result to the controller.
23    FORMAT(E13.6)

24    CONTINUE
      WRITE(ILU,25) A                     Print the measurements
25    FORMAT(10(2X,E13.6))                on the terminal
      GO TO 28

26    WRITE(ILU,27)ISTAT                  Error routine.
27    FORMAT (/" TOTAL: DEVICE ERROR: "K3" ")
                                          Show status byte in octal format.
28    END
      END$
```

PPOLL SUBROUTINE

PURPOSE: 1) PROGRAM A DEVICE(S) TO RESPOND TO A PARALLEL POLL ON A SPECIFIC DIO LINE (PARALLEL POLL ENABLE) OR
2) PROGRAM A DEVICE(S) NOT TO RESPOND TO PARALLEL POLL (PARALLEL POLL DISABLE) OR
3) UNCONFIGURE ALL DEVICES (PARALLEL POLL UNCONFIGURE)

PARALLEL POLL ENABLE FOR ONE DEVICE

BASIC	FORTRAN
100 PPOLL(D,1,A)	CALL PPOLL(IDLU,1,IASGN)

A, IASGN = POSITIVE OR NEGATIVE INTEGER IN THE RANGE OF ONE TO EIGHT REPRESENTING THE DIO LINE ON WHICH TO RESPOND TO A PARALLEL POLL.

A POSITIVE INTEGER INDICATES A ZERO RESPONSE AND A NEGATIVE INTEGER INDICATES A ONE RESPONSE TO A PARALLEL POLL IF THE DEVICE IS REQUESTING SERVICE.

D, IDLU ARE THE DEVICE LU

BUS TRAFFIC: ? (UNL), LISTEN ADDR. OF DEVICE, PARALLEL POLL CONFIGURE (PPC), PARALLEL POLL ENABLE (PPE).

PPOLL SUBROUTINE
(CONT)

PARALLEL POLL ENABLE FOR MULTIPLE DEVICES

BASIC	FORTRAN
100 PPOLL(B,1,A)	CALL PPOLL(IBLU,1,IASGN)

BUS TRAFFIC: PPC, PPE COMMANDS

B, IBLU ARE THE BUS LU#

THE ABOVE CALL PARALLEL POLL ENABLES A GROUP OF DEVICES ON THE INDICATED BUS. STATUS IS RETURNED ON A 1 BIT PER BUS BASIS FOR UP TO 8 BUSES.

PARALLEL POLL DISABLE FOR ONE DEVICE

BASIC	FORTRAN
100 PPOLL(D,2,0)	CALL PPOLL(IDLU,2)

BUS TRAFFIC: ? (UNL), LISTEN ADDR OF DEVICE, PPC, PPD (PARALLEL POLL DISABLE)

D, IDLU ARE THE DEVICE LU #

PPOLL SUBROUTINE
(AGAIN)

PARALLEL POLL DISABLE FOR MULTIPLE DEVICES

BASIC	FORTRAN
10 CMDW(B,A$,0)	CALL CMDW(IBLU,IA,0)
•	•
•	•
•	•
100 PPOLL(B,2,0)	CALL PPOLL(IBLU,2)

BUS TRAFFIC: PPC, PPD

B, IBLU ARE THE BUS LU #

A$ OR IA CONTAIN LISTEN ADDRESSES OF DEVICES TO BE DISABLED FROM RESPONDING TO A PARALLEL POLL. DIO ASSIGNMENTS ARE NOT RESET.

PARALLEL POLL UNCONFIGURE

BASIC	FORTRAN
100 PPOLL(B,3,0)	CALL PPOLL(IBLU,3)

BUS TRAFFIC: PARALLEL POLL UNCONFIGURE (PPU) COMMAND

B, IBLU ARE THE BUS LU #

THIS SUBROUTINE IS CALLED ONLY WITH THE BUS LU # AND RESETS THE DIO LINE ASSIGNMENTS SO NO DEVICES WILL RESPOND TO A PARALLEL POLL. DIO LINE ASSIGNMENTS VIA PARALLEL POLL ENABLE MUST TAKE PLACE BEFORE A PARALLEL POLL WILL RETURN ANY DEVICE'S STATUS BIT.

PSTAT SUBROUTINE

PURPOSE: INITIATE A PARALLEL POLL OPERATION. ANY DEVICE PREVIOUSLY CONFIGURED BY PPOLL WILL RESPOND ON A DIO LINE.

BASIC	FORTRAN
100 PSTAT(B,S)	CALL PSTAT(IBLU,ISTAT)

BUS TRAFFIC: ATN • EOI

B, IBLU ARE THE BUS LU #

S, ISTAT CONTAINS THE STATUS OF THE BUS DIO LINES IN BITS 0-7. BIT 0 REPRESENTS DIO 1, BIT 1 REPRESENTS DIO 2, ETC.

TRANSFER METHODS

TWO WAYS TO TRANSPORT INFORMATION ON THE BUS:

DMA	INTERRUPT
1) TRANSFER DATA DIRECT TO MEMORY.	1) TRANSFER DATA THROUGH CPU.
2) GENERATE ONLY TWO INTERRUPTS: j3.141590E+00CRLF	2) INTERRUPT AFTER EACH INPUT WORD: i+3i.1i41i59iOEi+0i0CRiLFi

WHEN TO USE THE INTERRUPT METHOD:

1) WHEN IT IS LIKELY THAT BOTH DMA CHANNELS WILL BE TIED UP FOR SIGNIFICANT AMOUNTS OF TIME.

2) FOR LOW SPEED DEVICES.

3) FOR MEDIUM AND HIGH SPEED DEVICES WITH SHORT BUFFERS.

WHEN TO USE THE DMA METHOD:

1) WHEN THERE IS LITTLE CHANCE THAT BOTH DMA CHANNELS WILL BE TIED UP BY OTHER DEVICES.

2) FOR MEDIUM AND HIGH SPEED DEVICES WITH LONG BUFFERS.

METHOD OF I/O

1. INTERRUPT BETWEEN WORDS.

2. SKIP ON FLAG SET.

3. DMA.

- UNLESS DMA IS SPECIFIED, THE METHOD 1 and 2 ARE USED.

- THE DRIVER ALWAYS CHECKS FOR A FLAG "SET" WHEN IT COMPLETES PROCESSING THE LAST WORD RECEIVED/TRANSMITTED. IF THE FLAG IS SET, THE DRIVER CONTINUES TO THE NEXT WORD WITHOUT EXITING FROM THE DRIVER.

- IF D BIT IS SET IN THE BUS EQT, DMA IS USED FOR ALL INSTRUMENTS ON THE BUS. OTHERWISE, ONLY THE INSTRUMENTS WHICH ARE CONFIGURED TO DO SO USE DMA (CONFIGURATION WORD BIT 13 = 1).

- UNLESS THERE ARE MORE THAN 3 BYTES TO OUTPUT, DMA IS NOT USED FOR ANY TYPE OF I/O.

- CONTROL REQUESTS WILL NOT USE DMA UNLESS D BIT IS SET IN THE EQT.

THERE ARE ONLY TWO DMA CHANNELS AVAILABLE ON RTE-IV SYSTEMS. THE BUS SHOULD NOT BE CONFIGURED FOR DMA IF OTHER DEVICES IN THE SYSTEM USE DMA EXCLUSIVELY (DISCS ETC.) SINCE THEY MAY TIE UP THE DMA CHANNELS FOR LONG PERIODS OF TIME CAUSING THE BUS TO HANG UP.

RTE-L OFFERS DMA/CARD SO THE BUS I/O CARD AUTOMATICALLY TRANSFERS DATA VIA DMA AND DOESN'T HAVE TO WAIT FOR A DMA CHANNEL.

CNFG SUBROUTINE

PURPOSE: STREAMLINE HP-IB PERFORMANCE, ADAPT TO SPECIFIC DEVICE REQUIREMENTS

TO CONFIGURE A DEVICE

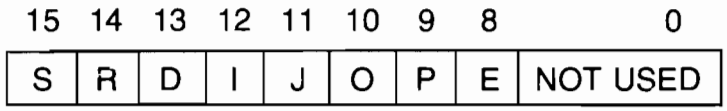
BASIC	FORTRAN
100 CNFG(D,1,C)	CALL CNFG(IDLU,1,ICONF)

BUS TRAFFIC: NONE

D, IDLU ARE THE DEVICE LU #

C, ICONF ARE THE DEVICE CONFIGURATION WORD

CONFIGURATION WORD FORMAT:



0 = DISABLE,
1 = ENABLE

BIT	DEFAULT	MEANING
15	0	DISABLE/ENABLE DRIVER TO ABORT CURRENT ACTIVE I/O REQUEST IN ORDER TO SERVICE ON SRQ INTERRUPT.
14	0	DISABLE/ENABLE DRIVER TO ATTEMPT TO RESTART I/O REQUEST WHICH WAS ABORTED. THIS BIT ONLY FUNCTIONAL IF S = 1.
13	0	DISABLE/ENABLE DRIVER TO USE DMA FOR I/O REQUESTS.
12	1	DON'T REQUIRE/REQUIRE EOR FROM DEVICE FOR END OF TRANSMISSION.
11	1	EXPECT EOR TO OCCUR AFTER/WITH LAST DATA BYTE.
10	1	DON'T ISSUE/ISSUE EOR TO DEVICE AT END OF TRANSMISSION.
9	1	ISSUE EOR AFTER/WITH LAST DATA BYTE.
8	0	ALLOW/DISALLOW OCCURRENCE OF ERROR TO ABORT CURRENT PROGRAM.

CNFG SUBROUTINE
(CONT)

TO UNCONFIGURE A DEVICE (RESET WORD TO DEFAULT FORMAT)

BASIC	FORTRAN
100 CNFG(D,2,0)	CALL CNFG(IDLU,2)

TO CONFIGURE THE BUS

BASIC	FORTRAN
100 CNFG(B,1,C)	CALL CNFG(IBLU,1,ICONF)

C, ICONF ARE THE CONFIGURATION WORD (SAME FORMAT AS DEVICE CONFIGURATION WORD)

NOTE: THE BUS CONFIGURATION IS USED WITH DIRECT I/O REQUESTS

TO UNCONFIGURE THE BUS (RESET WORD TO DEFAULT FORMAT)

BASIC	FORTRAN
100 CNFG(B,2,0)	CALL CNFG(IBLU,2)

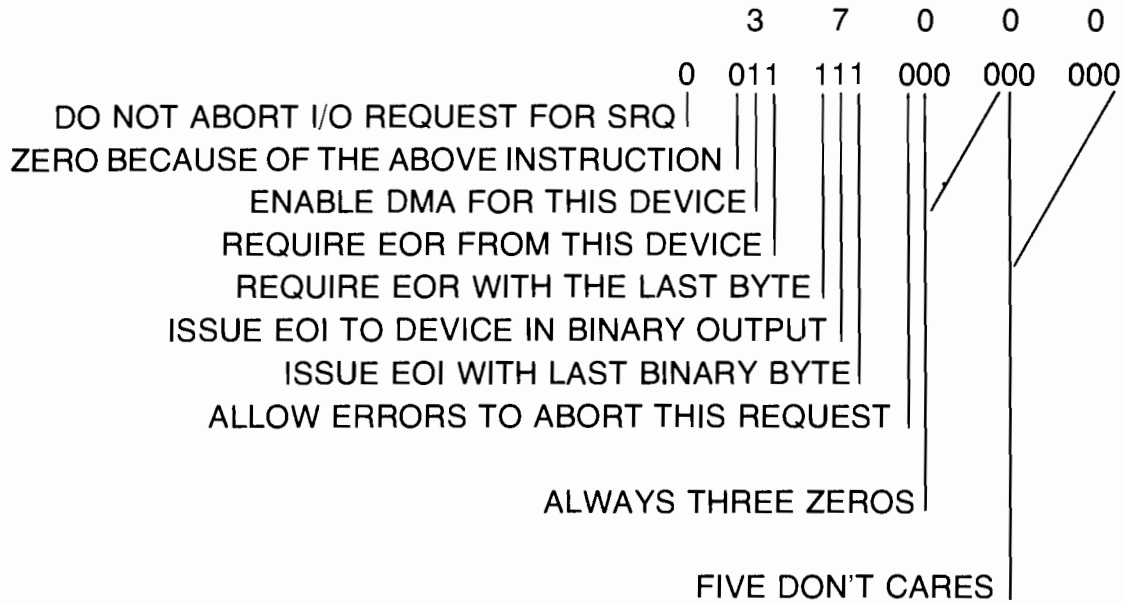
D, IDLU ARE THE DEVICE LU #
B, IBLU ARE THE BUS LU #

EXAMPLE OF SETTING THE CONFIGURATION WORD:

FTN4,L

```
      PROGRAM CNFGR
        .
        .
        .
      CALL CNFG(26,1,37000B)
        .
        .
        .
      END
```

THE CONFIGURATION WORD, OCTAL 37000, ISSUES THE FOLLOWING INSTRUCTIONS TO THE DRIVER:



THE SAME CALL IN BASIC:

```
10 CNFG(26,1,15872)
```

NOTE: $37000_8 = 15872_{10}$

THE BAIL OUT MESSAGE

Message	Subroutine	Subroutine Function
Abort	ABRT	Issues interface clear command IFC to an HP-IB interface card, or Issues interface clear command IFC and universal device clear command DCL to an HP-IB interface card. Unaddresses all devices on the bus, by issuing untalk and unlisten commands UNT and UNL to the bus.

ABRT SUBROUTINE

PURPOSE: TERMINATE CURRENT BUS OPERATION, RESET BUS AND DEVICES TO KNOWN STATE, OR CLEAR ALL TALKERS AND LISTENERS

TO RESET BUS TO KNOWN STATE

BASIC	FORTRAN
100 ABRT(B,1)	CALL ABRT(IBLU,1)

BUS TRAFFIC: IFC

TO RESET BUS AND DEVICES TO KNOWN STATE

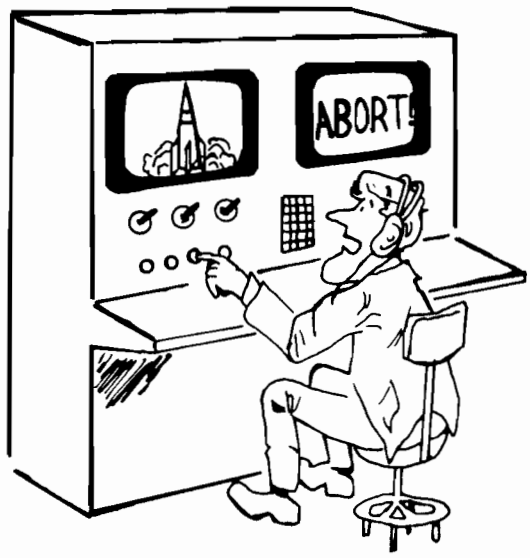
BASIC	FORTRAN
100 ABRT(B,2)	CALL ABRT(IBLU,2)

BUS TRAFFIC: IFC, DEVICE CLEAR (DCL) COMMAND

ABRT

SUBROUTINE

(CONT)



TO CLEAR TALKERS AND LISTENERS

BASIC	FORTRAN
100 ABRT(B,3)	CALL ABRT(IBLU,3)

BUS TRAFFIC: – (UNT), ? (UNL)

FOR ALL THREE FORMS OF THE ABRT CALL,

B OR IBLU IS THE BUS LU #

HANDLING BUS ERRORS

IBERR SUBROUTINE

PURPOSE: RETURN ERROR MESSAGE FROM A DEVICE OR THE HP-IB BUS TO THE USER PROGRAM.

BASIC	FORTRAN
I = IBERR(A)	IERR = IBERR(IA)

A, IA — DEVICE OR BUS LU#.

THE ERROR CODE IS DEFINED AS FOLLOWS:

0 = NO ERROR

1 = Device time out, or transmission error detected.

2 = IFC detected during I/O request

3 = Require service message has aborted data message.

4 = Specified service program does not exist.

5 = Illegal I/O request.

6 = System configuration error. EQT extension area full, no new device may be added on-line.

HANDLING ERRORS INVOLVES THREE STEPS:

(1) Bus must be unbuffered. ← can be found not from EQT

(2) The E bit, in the device configuration word, must be equal to 1.

(3) Error code must be checked after each data message.

ERROR HANDLING EXAMPLE:

The following program will configure the configuration word, issue a command string to logical unit number 25, and check for errors, then continue.

```
FTN4,L
      PROGRAM LU25                    S=1, R=0, E=1 in the
      CALL CNFG(25,1,117400B)         configuration word.

50    WRITE(25,500)                   Output data message to
500   FORMAT("F3R5M3T3")              the device.

      IERR=IBERR(25)                  Get error code.

      IF(IERR.EQ.0) GO TO 100         If error code=0,
                                      normal continuation of program.

      IF(IERR.EQ.3) GO TO 50          If SRQ abort, resend data
                                      message.

      WRITE(1,501)                    Print error message.
501   FORMAT(/" LU25: ERROR")
      STOP                            Terminate.

100   CONTINUE                        Continue program
      .
      .
      .
```

The driver defaults the E-bit of the configuration word to 0 causing I/O errors to abort programs. In other words, the default state is for the system to handle I/O errors.

The IBERR routine obtains error status from word 7 of the device's EQT extension area.

See chapter 9 for EQT and EQT extension entry formats.

DATA CONVERSION BASIC FORMATTED

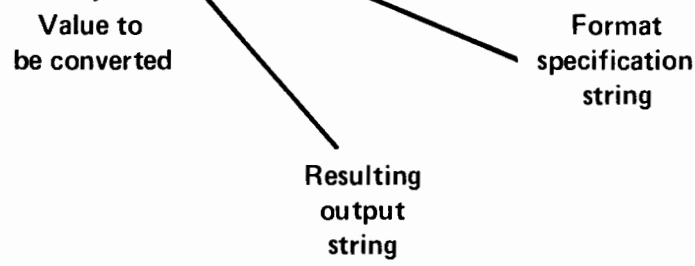
FUNCTION

TO PROVIDE BINARY TO ASCII AND ASCII TO BINARY FORMATTED CONVERSION IN BASIC

GENERAL FORMATS

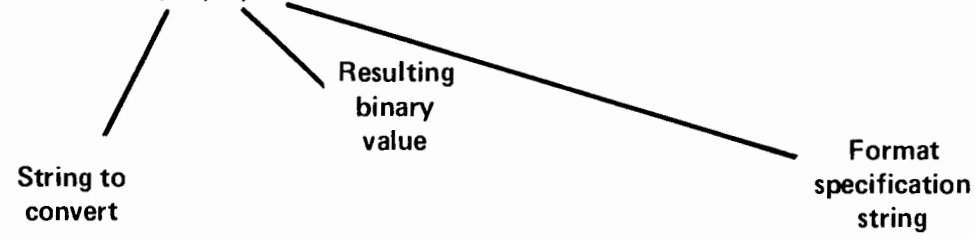
BINARY TO ASCII:

```
10 CALL DCODE (V, A$, B$)
```



ASCII TO BINARY:

```
10 CALL DCODE (A$, V, B$)
```



BINARY-TO-ASCII CONVERSION

EXAMPLE 1:

PREDEFINED STRINGS AND F-FORMAT CONVERSIONS; (∧ = BLANK).

```
10 DIM A$(7),B$(6)                    <define string length>
20 LET A$ = "xxx.xxx"                 <initialize string content>
30 LET B$ = "(F7.3)"                  <define format specification>
40 DCODE (V,A$,B$)                    <perform conversion>
```

variable (V)	string result (A$)
1.234	∧∧1.234
12.34	∧12.340
123.456	123.456
1234.567	1234.57
-1.3579	∧-1.358
12345600	$$$$$$$

EXAMPLE 2:

F-FORMAT CONVERSION TO PRODUCE AN INTEGER STRING.

```
10 DIM A$(12)                         <define string length>
20 LET A$ = "INTEGER=xxxx"            <initialize string content>
30 DCODE (V,A$(9,12),"(F4.0)")        <perform conversion>
```

variable (V)	string result (A$)
1234.0	INTEGER=1234
-765.432	INTEGER=-765

ASCII-TO-BINARY CONVERSION

EXAMPLE 3: ASCII-TO-BINARY CONVERSION BY F-FORMAT; (∧ = BLANK).

```
10 DIM A$(40),B$(6)          <define string length>
20 READ #12; A$              <input string via LU 12>
30 LET B$= "(F7.3)"          <define format specification>
40 DCODE (A$,V, B$)          <perform conversion>
```

ASCII string (A$)	result (V)
123.456	123.456
123.4∧∧	123.4
∧123.4 ∧	123.4
∧∧123.4	123.4
-00.123	-.123

EXAMPLE 4: ASCII-TO-BINARY CONVERSION BY THE E-FORMAT.

```
10 DIM A$(40)                <define string length>
20 READ #12; A$              <input string via LU 12>
30 DCODE (A$(5,16), V,"(E12.6)")   <perform conversion>
```

ASCII string (A$)	result (V)
DCV -.123456E+01	-1.23456
DCV +.123E+03	123.0
ABCD1.379E+00	1.379

BASIC DEVICE/PROGRAMMING EXAMPLE:

PROBLEM

PROGRAM HP 3495A SCANNER TO BE SET AT ANY REQUIRED CHANNEL.

SOLUTION

```
100 DIM $[6]
100 LET A$="C0000E"
```

Initialize string. Put clear and execute commands in the proper positions, and set channel numbers to zero.

```
120   INPUT C1,C2
```

Accept channel numbers.

```
130 IF C1>9 THEN 170
```

Skip if channel 1 has two digits.

```
140   IF C1=0 THEN 180
```

Skip if channel 1 = 0.

```
150   CALL DCODE(C1,A$[3,3],"(F1.0)")
160   GOTO 180
```

Put one-digit channel number in string position three.

```
170   CALL DCODE(C1,A$[2,3],"(F2.0)")
```

Put two-digit channel number in positions two and three.

```
180   IF C2>9 THEN 220
```

Skip if channel 2 has two digits.

```
190   IF C2=0 THEN 230
```

Skip if channel 2 = 0.

```
200   CALL DCODE(C2,A$[5,5],"(F1.0)")
```

Put one-digit channel number in string position five.

```
220 CALL DCODE(C2,A$[4,5],"(F2.0)")
```

Put two-digit channel number in positions four and five.

```
230   PRINT #23;A$
240   END
```

Output data message to channel scanner.

DATA CONVERSION IN FORTRAN

ASCII to Binary Conversion in FORTRAN

The following calling sequence makes an ASCII to binary conversion using subroutine CODE:

	`CALL CODE`	This statement must directly precede the `READ` statement.
	`READ (IBUF1,100) V1`	`IBUF1` contains the ASCII-coded string to be converted. `V1` is the variable to contain the converted value.
`100`	`FORMAT (F8.3)`	The format statement specifies the numeric format of the input data.

Binary to ASCII Conversion in FORTRAN

The following shows the FORTRAN calling sequence for binary to ASCII conversion:

	`CALL CODE`	This statement must directly precede the `WRITE` statement.
	`WRITE (V2,200) IBUF2`	`V2` contains variable to be converted; `IBUF2` will contain the converted string.
`200`	`FORMAT (E10.4)`	The format statement that specifies the numeric format of the output data.

ENCODE (C, F, buffer) LIST ← BIN → ASCII
DECODE (C, F, buffer) LIST ← ASCII → BIN
C = max # of characters in internal record
F = format statement label
buffer = location of internal record
LIST

FTN4+

EXAMPLE: ASCII to Binary Conversion in FORTRAN

```
FTN4,L
      PROGRAM VOLTM
      DIMENSION INBUF(7)                  Define buffer to receive
                                          measurement string.

      EQUIVALENCE(INBUF(3),MEAS)          MEAS is at the beginning of the
                                          numeric part of the string.

      CALL EXEC(1,27,INBUF,-14)           Do unformatted read from
                                          voltmeter, logical unit #27.
          .
          .                               Perform the remainder of the
          .                               time-critical measurements.

      CALL CODE                           Call conversion subroutine.

      READ (MEAS,100) VAL                 MEAS is where numeric part
                                          of string starts; VAL will
                                          contain converted value.

100   FORMAT (E10.0)                      Format statement by which
                                          conversion occurs.

      WRITE (10,120) VAL                  Display data on console.
120   FORMAT (F10.6)
      END
```

USING FILE MANAGER COMMANDS IN PLACE OF BUS SUBROUTINES

In most cases, configuration is a one-time job and can be performed in a simple manner which need not be repeated unless the operating system is restarted. Although switching the bus to remote usually needs to be performed, device configuration is often adequately done automatically by the operating system and defaults to the settings specified at system generation.

Device configuration can be set by a FORTRAN, BASIC, etc., user program, but the RTE File Manager also works nicely for configuring new instruments on HP-IB especially during device checkout. The commands needed are right at the user's fingertips; no preparation such as program writing, compilation, or relocation is necessary.

Basically, four commands are needed which are commonly used by RTE programmers (as shown below):

Using File Manager commands with HP-IB

HP-IB MESSAGES	FILE MANAGER COMMANDS
ABRT(<IBLU>,1)	:CN,<IBLU>,0,0
ABRT(<IBLU>,2)	:CN,<IBLU>,0,1
ABRT(<IBLU>,3)	NA
CLEAR(<IDLU>,1)	:CN,<IDLU>,0
CLEAR(<IBLU>,1)	NA
GTL(<IDLU>)	NA
GTL(<IBLU>)	NA
LLO(<IBLU>)	NA
LOCL(<IBLU>)	:CN,<IBLU>,17B
PRINT #<IDLU>	:LL,<IDLU>
WRITE (<IDLU>,fmt)	:AN,<ASCII COMMAND>
READ # <IDLU>	:DU,<IDLU>,namr,
READ (<IDLU>,fmt)	
PPOLL(<IDLU>,1,assign)	NA (special handling by driver)
PPOLL(<IBLU>,1,assign)	NA (special handling by driver)
PPOLL(<IBLU>,2[,0])	NA (special handling by driver)
PPOLL(<IBLU>,3[,0])	NA (special handling by driver)
PSTAT(<IBLU>,<STATUS>)	NA (special handling by driver)
RMOTE(<IDLU>)	:CN,<IDLU>,16B
RMOTE(<IBLU>)	:CN,<IBLU>,16B
STATS(<IDLU>,<STATUS>)	NA (Status is not retrievable)
TRIGR (<IDLU>)	NA (special handling by driver)
TRIGR (<IBLU>)	NA (special handling by driver)
CMDR(<IBLU>,<add>,<data>)	NA(double buffer request not allowed)
CMDW(<IBLU>,<add>,<data>)	NA (double buffer request not allowed
CNFG(<IDLU>,1,<WORD>)	:CN,<IDLU>,25B,<WORD>B
CNFG(<IBLU>,1,<WORD>)	:CN,<IBLU>,25B,<WORD>B
CNFG(<IDLU>,2[,0])	:CN,<IDLU>,27B,0
CNFG(<IBLU>,2[,0])	:CN,<IBLU>,27B,0
NA	:CN,<IDLU>,11B,-1 or :CN,<IDLU>,TO
SRQ(IDLU,16,IPROG)	NA (specially handled by driver)
SRQ(IDLU,17)	:CN,IDLU,21B

```
:CN --- Control non-disc device
:LL --- Change logical unit of list device
:AN --- Send ASCII message to list device[3]
:DU --- Send ASCII message from source to
        destination
```

File Manager commands for HP-IB

The table summarizes the list of commands and how each corresponds to the common set of HP-IB messages. In most configuration situations, the "CN" command can be used to perform the set up required.

Setting the Device to Remote

Almost all HP-IB devices need to be set to remote before HP-IB programming can take place. The file manager request,

```
:CN,IDLU,16B
     or
:CN,IBLU,16B
```

will perform the operation.[4] In most cases two conditions are required for a device to be in remote, so don't be alarmed if the remote light doesn't appear immediately after the request. First, the hardware "REN" line must be asserted. (This happens when the request is made.) Second, the device must receive its talk or listen address. Some devices must also be switched to data mode before the remote light will appear.

The bus logical unit (IBLU with device address zero) should be used to remote disable the bus.[4]

```
:CN,IBLU,17B
```

Configuring the Device

The number of devices which may be connected on a bus is determined at system generation time.[5] The Bus Status Utility Program[1] shows how many were allocated, how many have been used, and the number of spaces remaining. Once an LU assignment has been made and the device has been referenced in a request, one device space is said to have been allocated. (It can be determined from the utility program if an LU assignment was made, but no reference request has been attempted.)

Once a device space is allocated, five words are reserved for that device in the HP-IB driver (EQT) area in memory.

1. One word for device configuration.
2. Three words for the program name of a program to be scheduled on a service request.
3. One word for the device status received from the serial poll sequence.

Once a device space has been allocated, it will be deallocated only if specifically instructed to do so. The File Manager request to deallocate a device is:

```
:CN,IDLU,27B
```

Notice that once a device space has been allocated, the LU should not be reassigned to zero or to another device until a request has been made to deallocate the device space (as above). Once the LU reassignment is made, the EQT mapping is lost and can only be retrieved by reassigning an LU to the device. The device space deallocation can then be performed, thereby deallocating the device. (Note, that the Bus Status and Configuration Utility automatically cleans up unwanted device space if this mistake is made.[1]

---

[3]Note: The "AN" file manager command inserts a blank before the message. Care should be taken to see that the HP-IB device ignores blanks.

[4]Because the remote disable request is not device specific, the bus logical unit number (subchannel 0) must be used. The device logical unit number may be used with the remote enable request as a convenience to the programmer (when the bus LU is unknown).

[5]IEEE-488 indicates a maximum of 14 devices plus 1 system controller/controller.

7

PROGRAMMING THE BUS USING EXEC CALLS

EXAMPLE

CALL EXEC (1, LU+400B, IMAGE(1))

IOWD

IOWD = LU

FUNCTION = Bits

LU

EXEC INPUT/OUTPUT REQUEST:

CALL EXEC (ICODE,ICNWD,IDBFR,IDLNG,ICBFR,ICLNG)

where:

ICODE = Function Code
- 1 = Read
- 2 = Write
- 3 = CONTROL

ICNWD = Control Word composed of subfunction and logical unit

Format: 0 00Z 0X0 00M LLL LLL

- L = Device or Bus Logical Unit #, 1-63 (decimal)
- M = Data Format, 0=ASCII/1=Binary
- X = Transparent Mode, 0=Disabled/1=Enable
- Z = Buffer, 0=Single/1=Double

Therefore, the following conventions:

- 00LU = ASCII data record format. (EOR supplied by driver, bus interface card ASCII logic disabled)
- 01LU = Fixed length binary record format. (EOR supplied by driver, bus interface card ASCII logic disabled)
- 21LU = Transparent Mode Binary Format. (EOR not supplied by driver, bus interface card ASCII logic disabled)

IDBFR = Address of first word of data buffer

IDLNG = Data buffer length
- = n if specified as 16-bit words
- = −n if specified as 8-bytes

ICBFR = Address of first word of control buffer

ICLNG = Control buffer length.
- = n if specified as 16-bit words (16,383 words max.)
- = −n if specified as 8-bit bytes

LU — DEVICE LU# IN AUTO-ADDRESSING.
LU — BUS LU# IN DIRECT-ADDRESSING.

CONTROL REQUESTS

REQUEST	EXEC CALL
SDC	CALL EXEC(3,LU)
REMOTE ENABLE	CALL EXEC(3,1600B+LU)
REMOTE DISABLE	CALL EXEC(3,1700B+LU)
EOR	CALL EXEC(3,100B+LU)
SRQ	CALL EXEC(3,2000B+LU,IPROG)
DISABLE SRQ	CALL EXEC(3,2100B+LU)
CONFIGUR DRV. WORD	CALL EXEC(3,2500B+LU,IPRA)

NOTE: EXEC (3,~~2100B~~+LU) IS NOT CALLABLE FROM FORTRAN IN ORDER TO ESTABLISH A SERVICE PROGRAM.

DATA TRANSFER MODE

- **ASCII DATA RECORD**

  THE MOST USED RECORD FORMAT

  ASCII CHARACTER STRING TERMINATED BY A CARRIAGE RETURN, LINE FEED (CR/LF)

- **BINARY RECORD**

  FIXED NUMBER OF WORDS IN A RECORD

  RECORD TERMINATED BY EOI SIGNAL

- **TRANSPARENT MODE ASCII**

  RECORD MAY CONTAIN CONTROL CHARACTERS (IFC, REN, REN OFF, ATN ON, ATN OFF, EOR)

  ASCII CARD LOGIC ENABLED

  DVR37 DOES NOT SUPPLY CR/LF

- **TRANSPARENT MODE BINARY**

  RECORD WITH NO EOR

HOW TO SELECT DATA MODE AND CONFIGURATION WORD

- A DEVICE TRANSMITS/EXPECTS ASCII DATA FOLLOWED BY A LINE FEED.

  STANDARD I/O CALLS, NO CONFIGURATION (USE DEFAULT).

- A DEVICE TRANSMITS A FIXED NUMBER OF WORDS/BYTES BUT NO EOI.

  USE THE BUFFER LIMIT FOR TERMINATION. MAKE SURE THE BUFFER SIZE SPECIFIED IN READ STATEMENTS IS EXACT.

- A DEVICE EXPECTS VARIABLE LENGTH RECORDS TERMINATED BY EOI SIGNAL RIGHT AFTER THE VALID DATA.

  SET THE CONFFIGURATION WORD BIT 10 = 1 AND BIT 9 = 0. USE BINARY MODE WRITE STATEMENTS.

- A DEVICE TRANSMITS DATA WITHOUT A LINE FEED OR EOI SIGNAL.

  TRY TO CALCULATE TRANSMISSION TIME FROM THE DEVICE TO CPU AND SET THE TIME OUT VALUE. SET THE CONFIGURATION WORD, BIT 12 = 0.

RECORD TERMINATION

- END-OF-RECORD (EOR) LINE FEED OR EOI SIGNAL
- BUFFER END
- I/O TIME OUT (INPUT ONLY)

AUTO-ADDRESSING: DATA TO AND FROM A DEVICE

```
0001   FTN,L
0002   C  RUN PARAMENTERS ARE: CONSOL LU, CLOCK LU, DISPLAY LU.
0003         PROGRAM BUS04
0004         DIMENSION IBUFR(6),INPUT(6),IPRAM(5)
0005   C  INPUT & OUTPUT BUFFERS INITIALIZED.
0006         DATA IBUFR/2H12,2H34,2H56,2H78,2H90,2HE1/,INPUT/5*2H  /
0007         CALL RMPAR(IPRAM)
0008         LUTTY=1
0009         LUCLK=19
0010         LUDSP=20
0011         IF(IPRAM(1).NE.0) LUTTY=IPRAM(1)
0012         IF(IPRAM(2).NE.0) LUCLK=IPRAM(2)
0013         IF(IPRAM(2).NE.0) LUDSP=IPRAM(3)
0014   C  TRANSMIT OUTPUT BUFFER TO DISPLAY IN ASCII MODE.
0015         CALL EXEC(2,LUDSP,IBUFR,-12)
0016   C  INPUT FROM THE DIGITAL CLOCK.
0017         CALL EXEC(1,LUCLK,INPUT,-12)
0018   C  PRINT INPUT BUFFER.
0019         WRITE(LUTTY,100) INPUT
0020   100   FORMAT(6A2)
0021         PAUSE 1
0022   C  REPEAT OPERATION USING STANDARD 'READ/WRITE' STATEMENTS
0023         IBUFR(6)=2HE2
0024         WRITE(LUDSP,101) IBUFR
0025   101   FORMAT(6A2)
0026         READ(LUCLK,101) INPUT
0027         WRITE(LUTTY,101) INPUT
0028         END


**  NO ERRORS**     PROGRAM = 00159       COMMON = 00000
```

EXEC CALLS OR UTILITY SUBROUTINES:

USE EXEC. CALLS ONLY WHEN:

(1) SPEED OF EXECUTION IS IMPORTANT.

(2) THERE IS NOT ENOUGH MEMORY TO LOAD ALL THE UTILITY SUBROUTINES.

IN ALL OTHER CASES, AND MOST OF THE TIME, USE THE UTILITY SUBROUTINES:

(1)SAVE TIME IN PREPARING THE APPLICATION PROGRAM.

(2) MUCH EASIER AND CONVENIENT TO USE.

(3) MINIMIZE PROGRAM ERROR OCCURENCES.

IN THE END, ALL THE MESSAGE ROUTINES SEEN SO FAR USE EXEC CALLS TO PERFORM THEIR FUNCTIONS. EXEC CALLS ARE NOT AS CONVENIENT FOR THE USER BUT OFFER FASTER EXECUTION TIME AND REQUIRE LESS MEMORY THAN THE BUS MESSAGE ROUTINES.

8

DEVICE SUBROUTINES

DEVICE SUBROUTINES

PURPOSE:

TO SIMPLIFY THE DEVICE PROGRAMMING TASK AND REDUCE THE AMOUNT OF PROGRAM SPACE REQUIRED.

WHEN TO USE (TYPICALLY):

- WHEN A DEVICE IS USED MANY TIMES IN A PROGRAM EACH TIME WITH DIFFERENT PARAMETERS.
- WHEN A SPECIAL DEVICE REQUIRES A COMPLICATED PROGRAMMING PROCEDURE (SEE 8660A EXAMPLE).
- WHEN A DEVICE REQUIRES A MULTI-STEP PROCEDURE TO ENABLE IT TO PERFORM A FUNCTION (SEE 3455A EXAMPLE).
- WHEN A DEVICE IS BEING USED WITH OTHER DEVICES TO PERFORM AN OPERATION (SEE 59306A EXAMPLE).

HOW TO WRITE DEVICE SUBROUTINES

THE FOLLOWING GENERAL PROCEDURE APPLIES:

- DEFINE THE PROBLEM TO BE SOLVED WITH THE DEVICE SUBROUTINE.
- DETERMINE THE DEVICE PROGRAMMING CODES; REFER TO THE DEVICE PROGRAMMING/OPERATING MANUAL.
- ENSURE THAT THE DEVICE CONFIGURATION WORD IS PROPERLY CONFIGURED; DMA REQUIRE, OR NOT, SET E BIT, ETC.
- SET UP ERROR MESSAGE ROUTINE (HP-IB MUST BE UNBUFFERED).
- USE THE HP-IB MESSAGES WHERE POSSIBLE TO SIMPLIFY THE DEVICE SUBROUTINE.
- USE EXEC CALLS ONLY WHEN SPEED OF EXECUTION IS IMPORTANT OR YOU DO NOT HAVE ENOUGH MEMORY TO LOAD ALL THE HP-IB MESSAGES.

DEVICE SUBROUTINE

EXAMPLES

```
* LIBARY
LI, %DVM
LI, %LOCKM   } upto 10 LI
LI, %ERR
  :
EN
```

OR

```
RU, MERGE
OUTPUT NAME  $ HPIBL.SUBS
INPUT NAME   %DVM
             %LOCKM
```


then *LIBARY
LI, HPIBL

HP 8660A/B SYNTHESIZER, DEVICE SUBROUTINE

DESCRIPTION:

THE HP 8660A/B SYNTHESIZER REQUIRES A DATA MESSAGE IN REVERSE ORDER, WITH 10 SIGNIFICANT DIGITS. FOR EXAMPLE, TO SET THE 8660 to 21.5 MHZ, THE FOLLOWING ASCII STRING SHOULD BE SENT: 0000051200.

PROBLEM:

WRITE A DEVICE SUBROUTINE TO SET THE FREQUENCY OF THE 8660 TO ANY FREQUENCY BETWEEN 1 Hz AND 500 MHZ.

```
&MC    T=00003 IS ON CR00037 USING 00004 BLKS R=0000

0001  FTN4,L
0002        SUBROUTINE RFF(ISLU,FRQ)
0003  C
0004        COMMON IP(5)
0005        INTEGER IBUF1(5),IBUF2(5)
0006  C    THIS ROUTINE SETS UP THE FREQUENCY FOR THE 8660.  THE FREQUENCY
0007  C    IS SENT TO THIS ROUTINE AS A DOUBLE PRECISION VALUE SINCE TEN DIGITS
0008  C    ARE REQUIRED.  THE ROUTINE RECEIVES THE LU OF THE 8660 AND THE FREQ,
0009  C    IT REVERSES THE ORDER OF THE DIGITS, AND SENDS THE VALUE TO THE 8660.
0010  C    THE FREQUENCY SHOULD BE IN MHZ.
0011        DOUBLE PRECISION FRQ,A
0012        A=FRQ*1E6
0013        CALL CODE
0014        WRITE(IBUF1,101) A
0015  101   FORMAT(1I10)
0016        DO 88 I=1,5
0017        IL=IAND(IBUF1(I)/400B,377B)
0018        IF(IL.EQ.040B) IL=060B
0019        IH=IAND(IBUF1(I),377B)
0020        IF(IH.EQ.040B) IH=060B
0021  88    IBUF2(6-I)=IH*400B+IL
0022        WRITE(ISLU,102)IBUF2
0023  102   FORMAT(5A2,"(")
0024  C  ERROR CHECK
0025        IERR=IBERR(ISLU)
0026        IF(IERR)40,60,40
0027   40   WRITE(IP(1),10)IERR
0028   10   FORMAT("HP-IB ERROR NO. ",I1)
0029   60   CALL CLEAR (ISLU,1)
0030        END
```

HP 3455A DVM, DEVICE SUBROUTINE

PROBLEM:

WRITE A DEVICE SUBROUTINE THAT WILL SET UP THE DVM, TRIGGER IT, TAKE A MEASUREMENT, AND STORE IT IN MEMORY.

```
&MC6   T=00003 IS ON CR00014 USING 00005 BLKS R=0000

0001  FTN4,L
0002        SUBROUTINE DVM(IDLU,R1,IF,IR,IT,IM,IA,IH,ID)
0003  C
0004  C  IDLU=DVM LU       R1=VARIABLE TO CONTAIN THE MEASUREMENT READING
0005  C  IF=FUNCTION  PROGRAM CODE              IR=RANGE PROGRAM CODE
0006  C  IT=TRIGGER PROGRAM CODE                IM=MATH PROGRAM CODE
0007  C  IA=AUTO CAL.  PROGRAM CODE             IH=HIGH RES.  PROGRAM CODE
0008  C  ID=DATA READY RQS.  PROGRAM CODE
0009  C  FOR EXAMPLE:
0010  C  TO SET THE DVM TO:DC VOLT,AUTO RANGE,HOLD/MANUAL TRIGGER,MATH OFF
0011  C  AUTO CAL OFF,HIGH RESOLUTION OFF AND DATA READY RQS ON,THE CALL
0012  C  WILL BE:  CALL DVM(IDLU,R1,1,7,3,3,0,0,1)
0013  C
0014        COMMON IP(5)
0015  C  SET THE DVM
0016        WRITE(IDLU,10)IF,IR,IT,IM,IA,IH,ID
0017     10  FORMAT("F",I1,"R",I1,"T",I1,"M",I1,"A",I1,"H",I1,"D",I1)
0018  C ERROR CHECK
0019        IERR=IBERR(IDLU)
0020        IF(IERR)40,60,40
0021     40  WRITE(IP(1),20)IERR
0022     20  FORMAT("HP-IB ERROR NO. ",I1)
0023        CALL CLEAR (IDLU,1)
0024        STOP
0025  C   TRIGGER THE DVM
0026     60  CALL TRIGR(IDLU)
0027  C  TAKE A MEASUREMENT FROM DVM
0028        READ(IDLU,*)R1
0029        CALL CLEAR(IDLU,1)
0030        END
```

HP 59306A RELAY ACTUATOR, DEVICE SUBROUTINE

DESCRIPTION:

THE 59306A IS A RELAY ACTUATOR THAT CONTAINS 6 RELAYS. THE 59306A IS CONNECTED TO A STEP ATTENUATOR. BY CLOSING A CERTAIN COMBINATION OF CHANNELS 4, 5 AND 6, THE FOLLOWING ATTENUATION STEPS WILL BE PERFORMED.

DB	OPEN	CLOSE
0	4, 5, 6	—
10	5, 6	4
20	4, 6	5
30	6	4, 5
40	4, 5	6
50	5	4, 6

FOR THIS EXAMPLE ASSUME THAT ASCII A FOLLOWED BY THE CHANNEL NUMBER WILL CLOSE THE CHANNEL AND ASCII B WILL OPEN IT.

PROBLEM:

WRITE A DEVICE SUBROUTINE THAT WILL PERFORM THE ATTENUATION WHICH THE CALLING PROGRAM SPECIFIES.

```
&MC5   T=00003 IS ON CR00014 USING 00007 BLKS R=0000

0001  FTN4,L
0002        SUBROUTINE ATTEN(IRLU,IA)
0003  C     THIS IS A ROUTINE TO SET THE 59306A RELAY ACTUATOR TO A COMBINATI
0004  C     OF ATTENUATION STEPS ON RELAY CHANNELS 4, 5, AND 6.  THE STEPS AR
0005  C     CONNECTED AS FOLLOWS:
0006  C
0007  C                        STEP : DB : OPEN : CLOSED
0008  C                        ------------------------
0009  C                         1   :  0 : 567  : --   :
0010  C                         2   : 10 : 56   : 4    :
0011  C                         3   : 20 : 46   : 5    :
0012  C                         4   : 30 : 6    : 45   :
0013  C                         5   : 40 : 45   : 6    :
0014  C                         6   : 50 : 5    : 46   :
0015  C                        ------------------------
0016  C
0017        COMMON IP(5)
0018        INTEGER CODE(6,3)
0019  C     IT IS ASSUMED THAT THE LOGICAL UNIT FOR THE RELAY BOX IS 23.  IF
0020  C     ANOTHER LU IS DESIRED, CHANGE THE FOLLOWING STATEMENT.
0021        DLU=23
0022        CODE(1,1)=2HB4
0023        CODE(1,2)=2H56
0024        CODE(1,3)=2H
0025        CODE(2,1)=2HA4
0026        CODE(2,2)=2HB5
0027        CODE(2,3)=2H6
0028        CODE(3,1)=2HA5
0029        CODE(3,2)=2HB4
0030        CODE(3,3)=2H6
0031        CODE(4,1)=2HA4
0032        CODE(4,2)=2H5B
0033        CODE(4,3)=2H6
0034        CODE(5,1)=2HA6
0035        CODE(5,2)=2HB4
0036        CODE(5,3)=2H5
0037        CODE(6,1)=2HA4
0038        CODE(6,2)=2H6B
0039        CODE(6,3)=2H5
0040        WRITE(DLU,101) (CODE(IA,J),J=1,3)
0041   101  FORMAT(3A2)
0042  C  ERROR CHECK
0043        IERR=IBERR(IRLU)
0044        IF(IERR)40,60,40
0045   40   WRITE(IP(1),10)IERR
0046   10   FORMAT("HP-IB ERROR NO. ",I1)
0047   60   CALL CLEAR(IRLU,1)
0048        END
```

USE OF A DEVICE SUBROUTINE IN A PROGRAM

PROBLEM:

WRITE A PROGRAM THAT WILL PERFORM THE FOLLOWING:

1. SET UP THE 8660 SYNTHESIZER TO OUTPUT 26.35 MHZ.

2. SET UP THE ATTENUATION TO BE 50 DB.

3. SET UP THE DVM TO: AC VOLTS, AUTO RANGE, HOLD/MANUAL TRIGGER, MATH OFF, AUTO CALL OFF, HIGH RES. OFF, AND DATA READY RQS ON.

TAKE A MEASUREMENT AND PRINT IT ON THE TERMINAL.

MAKE USE OF THE PREVIOUS DEVICE SUBROUTINE EXAMPLES.

```
&MC4   T=0001 IS ON CR0014 USING 00002 BLKS R=0000

0001   FTN4,L
0002         PROGARM SRD
0003         COMMON IP(5)
0004         CALL RMPAR IP(5)
0005         IF(IP(1).EQ.0)IP(1)=1
0006         ISLU=37
0007         IDLU=38
0008         IRLU=39
0009   C     SET UP 8660
0010           CALL RFF(ISLU,26,35)
0011   C     SET UP ATTENUATION
0012           CALL ATTEN(IRLU,5)
0013   C     SET UP DVM
0014           CALL DVM(IDLU,R1,2,7,3,3,0,0,1)
0015   C     DISPLAY MEASUREMENT FROM DVM
0016           WRITE(IP(1),10)R1
0017     10    FORMAT("R1=",F8.3)
0018         END
```

9

HP-IB GENERATION & SYSTEM CONSIDERATIONS

HP-IB GENERATION

PROGRAM INPUT PHASE:

LOAD ONE OF THE TWO HP-IB DRIVERS

%2DV37 — HP-IB DRIVER WITH SRQ CAPABILITY

%1DV37 — HP-IB DRIVER WITHOUT SRQ CAPABILITY

IF BASIC IS TO BE USED, LOAD THE BASIC MEMORY RESIDENT LIBRARY AND THE SRQ/TRAP PROGRAM, SRQ.P.

BE SURE TO LOAD %IB4A, HP-IB RTE LIBRARY, AS A LIBRARY.

PARAMETER INPUT PHASE:

EXCEPT FOR RTE-M, RTE-II, AND RTE-III SYSTEMS, MAKE THE FOLLOWING PARAMETER INPUTS IF BASIC TRAPS ARE TO BE USED:

TTYEV, 17

TRAP, 30

TABLE GENERATION PHASE:

EQUIPMENT TABLE (EQT):

1. DETERMINE I/O SLOT OF BUS INTERFACE CARD AND NOTE SELECT CODE.
2. DETERMINE THE NUMBER OF EQT EXTENSION WORDS NEEDED USING THE FORMULA:

   # EXTENSION WORDS = 7N + 18 (255 MAX) add on more

   WHERE N = NUMBER OF DEVICES ON THE BUS

   BE SURE TO INCLUDE ENOUGH EXTENSION WORDS TO ALLOW FOR LATER ADDITIONS TO THE BUS.
3. IF DESIRED, SPECIFY BUFFERING. BUFFERING DVR37 IS NOT RECOMMENDED, HOWEVER.
4. DETERMINE THE MAXIMUM TIME OUT FOR THE SLOWEST DEVICE ON THE BUS IF IN RTE-IV SYSTEM. IN RTE-L, EACH DEVICE GETS ITS OWN TIME OUT VALUE.

HP-IB GENERATION (CONTINUED)

EACH EQT ENTRY IS MADE IN THE FORM:

EQT n?
sc,DVR37,B,T=xxxx,X=yy

where n = EQT # ASSIGNED BY RTE SYSTEM
sc = SELECT CODE OF BUS INTERFACE CARD
B = BUFFERING OPTION (NOT RECOMMENDED)
T = TIME OUT IN 10'S OF MILLESECONDS
X = # OF EQT EXTENSION WORDS

IF YOU DO NOT DESIRE THE BUFFERING OPTION, OMIT THE LETTER AND COMMA.

DEVICE REFERENCE TABLE (DRT):

THE DEVICE REFERENCE TABLE MAPS THE DEVICE LU'S INTO THE EQT. MAKE THE DRT ENTRIES AS SHOWN:

lu=EQT#?

n,m

where lu = LOGICAL UNIT # TO BE ASSIGNED TO DEVICE
n = BUS EQT #
m = EQT SUBCHANNEL NUMBER =- DEVICE ADDRESS CODE

NOTE:

1. ASSIGN SUBCHANNEL 0 TO THE BUS INTERFACE CARD
2. ASSIGN AN LU # AND A SUBCHANNEL # (1-31) TO EACH DEVICE TO BE AUTO-ADDRESSED.

INTERRUPT TABLE:

THE INTERRUPT TABLE ALLOWS YOU TO ESTABLISH INTERRUPT LINKS THAT TIE THE SELECT CODE TO ITS EQT NUMBER. MAKE TABLE ENTRIES AS SHOWN:

sc,EQT,n

where
sc = SELECT CODE OF BUS INTERFACE CARD
N = EQT NUMBER PREVIOUSLY ASSIGNED TO THE CARD

THE TABLE ENTRY PHASE MUST BE REPEATED FOR EACH BUS INTERFACE CARD IN THE SYSTEM.

EXPANDED HP-IB AND BASIC LIBRARIES

%IB4A IS SUBDIVIDED INTO FOUR MODULES:

IB4A — HEADER MODULE

IB4A2 — ENTRY POINTS FOR:

HP-IB Message routines
TRIGR,CLEAR,RMOTE,FTL,LLO,LOCL,
STATS,PPOLL,PSTAT,CNFG,ABRT

Secondary addressing routines
SECR,SECRR,SECW,SECWR

Direct addressing routines
CMDR,CMDW

Status, service and error routines
SRQ,IBERR,IOCNT

IB4A3 — ENTRY POINT FOR:

RTE-IV HP-IB utilities accessing SSGA ENT's
SRQSN

IB4A4 — ENTRY POINTS FOR :

RTE-IV HP-IB mature utilities
HPIB, IBSTS

%BASLB CONTAINS ENTRY POINTS FOR:

Mag tape routines
MTTRD,MTTRT,MTTPT,MTTFS

Bit manipulation routines
IEOR,ISHFT,AND,OR,NOT,
IBTST,IBSET,IBCLR,ISETC

Data conversion routines (for HP-IB)
DCODE,CHRS,NUM

Decimal string arithmetic routines
SADD,SSUB,SMPY,SDIV,SEDIT

String handling routines (for HP-IB)
DEB$,BLK$

Device subroutines
B.SET,B.STAT

%BAMLB CONTAINS ENTRY POINTS FOR:

Routines required for real time task scheduling
TRAP,TIME

THE HP-IB EQUIPMENT TABLE ENTRY

Each equipment table entry is made in the form:

```
sc,DVR37,T=nnn,X=xx
```

where:

sc = Select code of HP-IB interface card

T = Time out value

X = EQT extension
- xx = 10+3* # of devices on bus (up to Rev. 1826)
- xx = 12+5* # of devices on bus (from Rev. 1840 – Rev. 1940)
- xx = 18+7* # of devices on bus (255 words max) (from Rev. 1940)

The EQT entry is a 15 word block as shown in figure A-1.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	I/O LIST LINK															
2	DVR I. 37 ADDRESS															
3	DVR C. 37 ADDRESS															
4	D	B	P	S	T	UNIT #							CHANNEL #			
5	AV	37							STATUS							
6	CONWD (CURRENT I/O REQUEST WORD)															
7	REQUEST BUFFER ADDRESS															
8	REQUEST BUFFER LENGTH															
9	CONTROL BUFFER ADDRESS															
10	CONTROL BUFFER LENGTH															
11	S	A		E	B		H	L					C	M	D	I
12	S	P	A	# OF BEQT					# OF EQT EXTENSION WORDS							
13	I	EQT EXTENSION ADDRESS														
14	DEVICE TIMEOUT RESET VALUE															
15	DEVICE TIMEOUT CLOCK															

HP-IB EQT Entry

Word 4	D	=	DMA assigned during generation, 1 = Yes
	B	=	Buffering on, 1 = Yes
	P	=	PWR Fail serviced by DVR, 0 = No
	S	=	Time-out serviced by DVR, 1 = Yes
	T	=	Time-out occurrence, 1 = Yes
	Unit #	=	Unit or subchannel, present request
	Channel #	=	Select code of HP-IB card, present request
Word 5	AV	=	I/O controller availability
	37	=	HP-IB (device type)
	Status	=	Status byte
Word 11	S	=	SRQ service in progress, 1 = Yes
	A	=	I/O request aborted to service SRQ, 1 = Yes
	E	=	Expect/issue EOR at end of current I/O, 1 = Yes
	B	=	Expect/issue EOR with last data byte of current I/O, 1 = Yes
	H	=	Enable ASCII mode I/O card logic, 1 = Yes
	L	=	Suppress line feed. Only bit 7 of BEQT1 is checked
	C	=	Enable CR/LF post processing, 1 = Yes
	M	=	Data mode, 1 = ASCII, 0 = Binary
	D	=	DMA active on pending request, 1 = Yes
	I	=	I/O direction, 1 = Input, 0 = Output
Word 12	S	=	SRQ pending flag
	P	=	Alarm program scheduling active
	A	=	SRQ interrupt arming flag (via SRQ statement)
	# of BEQT	=	# Active BEQT entries, 0-31
	# EQT Extensions	=	# EQT Extension words, 18-255
Word 13	I	=	Initiator/Continuator flag
	A	=	EQT Extension address.

NOTE

This description assumes that DVR37 has the SRQ alarm service.

Word 13 of the HP-IB EQT entry contains the address of the HP-IB EQT Extension which consists of 18 fixed words plus an additional 7n words where n is the number of devices on the bus. The extension may use a maximum of 255 words (See lower byte of EQT 12). Figure A-2 shows the EQT Extension format.

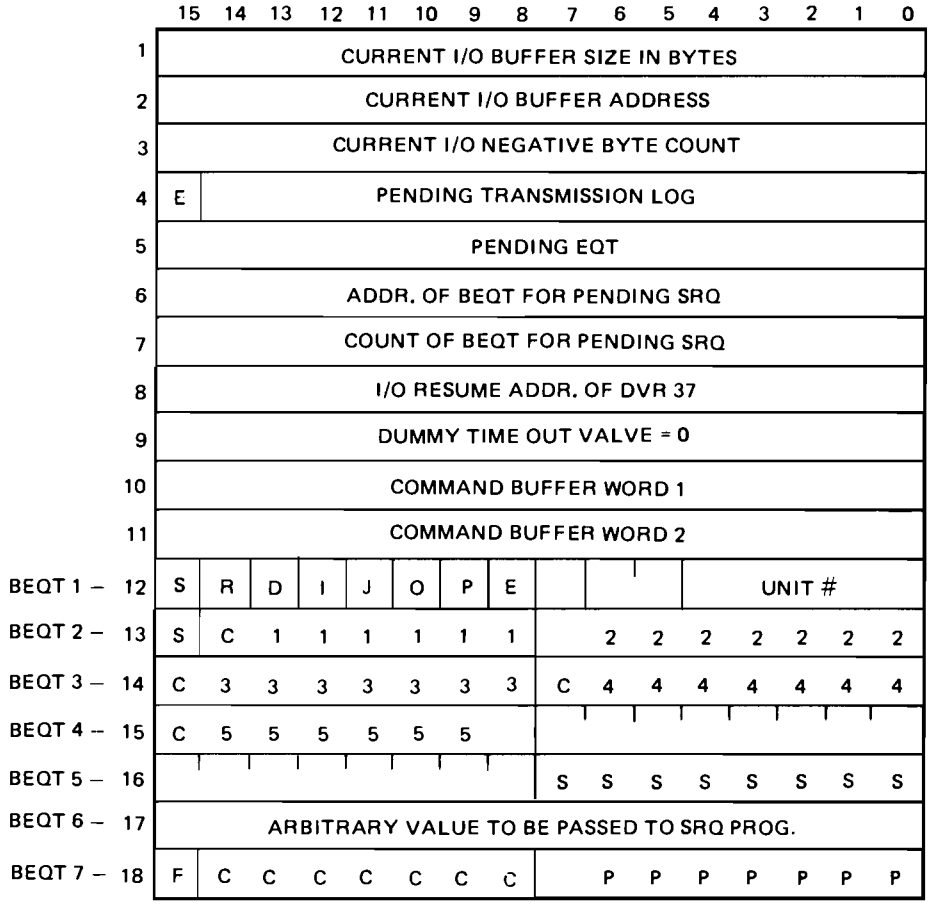


Figure -2. EQT Extension Fixed Area Format
(From Rev. 1940)

Besides the 18 word fixed area, the HP-IB EQT extension also consists of 7 words of information for each device on the bus. These 7 words are called the EQT Extension BEQT and are identical in format to the fixed area words 12-18 shown above. In fact, words 12-18 of figure A-2 represent BEQT 1-7 for the HP-IB itself.

Word 1, 2, and 3 are set during initiation using EQT7 and EQT8. See figure A-1.

Word 3 Negative value of byte count initially and is incremented by 1 for each byte processed for non-DMA operation.

Record type indicator rather than a character count for DMA operation.

= 0 EOI not required or EOI after last valid data byte
= −1 Odd byte record
= −2 EOI with last valid data byte

Word 4 E = 1 if previous I/O ended in error, bits 0-14 = # of bytes processed in previous I/O

Word 5 Same as EQT5 upon I/O completion, serial poll status during SRQ process.

Word 6 Address of BEQT used as working cell during SRQ process

Word 7 Negative value of # of BEQT used as working cell during SRQ process (−n to 0)

Word 8 DOIO return address for re-entrancy

Word 10-11 Temporary buffer used by DVR37 to transmit commands

Word 12 Bus configuration word (BEQT1 for bus)

S = 0 = Do not allow driver to abort a currently active I/O request in order to service an SRQ interrupt
R = 0 = Do not allow driver to attempt to restart an I/O request that was aborted
D = 0 = Disable DMA
I = 1 = Require EOI from device at end of transmission
J = 1 = Expect EOI with last data byte
O = 1 = Issue EOI at end of transmission
P = 1 = Issue EOI with last data byte
E = 0 = Allow occurrence of error to abort current program
L = 1 = Suppress LF on next output (Supplied *only* for line printer support)

Word 13 BEQT2 for bus
S = 1 = SRQ program is to be scheduled
C111111 = Character 1 of SRQ program name
C2222222 = Character 2 of SRQ program name

Word 14 BEQT3 for bus
C3333333 = Character 3 of SRQ program name
C4444444 = Character 4 of SRQ program name

Word 15 BEQT4 for bus
C5555555 = Character 5 of SRQ program name

Word 16 BEQT5 for bus
SSSSSSSS = Last read serial poll status

Word 17 BEQT6 for bus
Arbitrary value to be passed SRQ program

Note: SRQ program cannot be configured for bus.

Word 18 BEQT7 for bus
F = 1, PPPPPPPP = Error status of last operation (bits 0-7 of A-register)
F = 0, C• • • P • • • = Transmission log of last operation (bits 0-15 of B-register)

Note once again that each device on the bus also has a 7-word BEQT identical in format to that for the bus itself as described above.

SETTING UP A SYSTEM

GENERAL AREAS TO CONSIDER . . .

A. COMPUTER I/O SLOT FOR 59310B INTERFACE BUS INTERFACE (IBI).

B. FIRST INSTRUMENT TO ATTACH TO BUS CABLE. THIS IS USUALLY DESIGNATED AS A CONTROLLER.

C. ROLE OF EACH INSTRUMENT ON THE BUS AS A TALKER, LISTENER, OR BOTH.

IBI INSTALLATION

* SIMPLE INSTALLATION PROCEDURE AND RESPONSIBILITY OF USER:

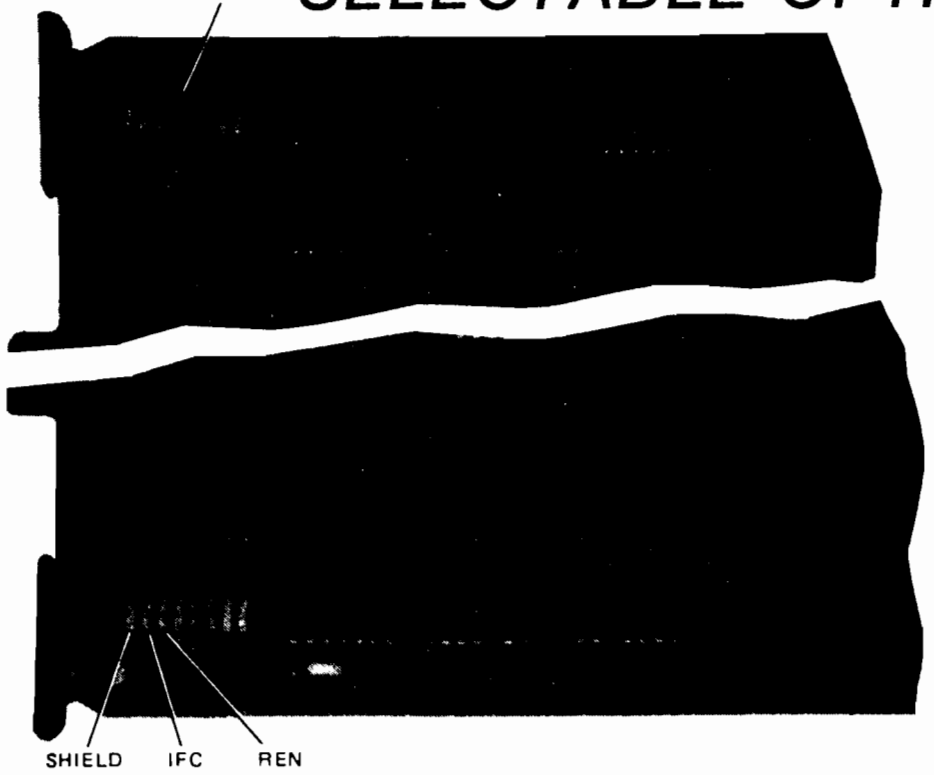
1. SET THE DESIRED ADDRESS ON THE I/O CARD USING SWITCHES OR JUMPERS FOR DESIRED SYSTEM FUNCTIONS. FOR RTE-IV THE RECOMMENDED ADDRESS SETTING IS 0B. FOR RTE-L, THE CARD IS HARDWIRE ADDRESSED TO 36B.

2. PLUG BUS I/O CARD INTO AN AVAILABLE COMPUTER OR I/O EXTENDER SLOT. DETERMINE WHETHER THE COMPUTER HAS SUFFICIENT POWER TO HANDLE THE LOAD OF THE BUS I/O CARD.

3. INTERCONNECT SYSTEM OBSERVING CABLING AND DISTANCE RESTRICTIONS.

4. VERIFY PROPER OPERATION BY RUNNING THE DIAGNOSTIC PROGRAM. REFER TO THE HP 59310A/B INTERFACE BUS DIAGNOSTIC REFERENCE MANUAL (59310-90061).

SETTING IBI SELECTABLE OPTIONS



Bus Interface Card Switch Descriptions

SWITCH DESIGNATION	FUNCTION	NORMAL SETTING
REN	When ON, enables the card to drive the bus signal line REN. When OFF, disables the card from driving the bus signal line REN	ON
IFC	When the switch is set to ON, enables the card to drive the bus signal line IFC. When the switch is set to OFF, disables the card from driving bus signal line IFC.	ON
SHIELD	When the switch is set to CNX, connects the shield in the bus cable to the logic circuit common. When the switch is set to DNX, disconnects the shield in the bus cable from the logic circuit common. (Unless some other device connects the shield, it will remain disconnected.)	CNX
PP ID CODE	Defines the code that the bus interface card will send in response to a Parallel Poll. Setting the switch to ON will cause the card to use the DIO signal line specified to be used. Note that only one switch may be on at one time.	all OFF
W1	Allows IBI to Send PP ID code	IN*

*Necessary if Parallel Poll Test of the diagnostic is executed.

To determine whether the computer power supply has sufficient power to handle the load of the 59310B interface card:

1. Determine the maximum current available from the power supply and the maximum current required by the CPU, MEMORY, and I/O cards by referring to:

   i. HP 1000 computers, Hardware Data Guide
   or
   ii. Installation and Service Manual
   or
   iii. Configuration and Site Preparation Guide
   or
   iv. Your local SE (or factory if you are an SE)

2. Subtract the maximum required current for each memory and I/O board already in the system from the power supply maximum output current observing the following rules:

   i. The 59310B card requires sources of +5V and −2V. The power supply provides different maximum current output at these two levels. Add the +5V maximum current requirements of the CPU board, memory boards and I/O cards already installed in the CPU box and main I/O section and subtract from the +5V maximum current rating of the supply. Add the −2V maximum current requirements of the CPU, memory, and I/O cards and subtract from the −2V maximum current rating of the supply. Since I/O extenders have their own power supplies, only I/O cards in the main I/O section in the backplane of the computer should be used in the calculation.

   ii. If no card slots remain in the main I/O section, an I/O extender is required. If an I/O extender with an empty card slot is already in use, perform the calculation of (i) for the I/O cards already in the extender omitting the CPU and memory maximum current requirements.

3. The 59310B requires 3.0A at +5V and 0.1A at −2V. If this much current is available for the card as calculated in (i) or (ii), it may be inserted without overloading the power supply.

SYSTEM CONSIDERATIONS

LU ASSIGNMENTS

Bus LU's may be assigned during system generation or on-line via FMGR commands. The HP-IB bus is assigned an LU and EQT entry at generation time during the table generation phase which will be described later. HP-IB device LU's may also be assigned at this time. The generator prompts:

```
lu = EQT # ?
```

Where `lu` = LU to be assigned to device

The appropriate response is:

```
nn,unit
```

Where `nn` = BUS EQT
`unit` = Address of BUS device (BUS subchannel)

An optional comment field may be included after the unit # and must be preceeded by an asterisk (*).

For example, to assign LU 10 as subchannel 4 of EQT 3 (BUS EQT):

The generator prompts:

```
10 = EQT #?
```

You respond:

```
4,3          * 3455A DIGITAL VOLTMETER
```

If LU's are not assigned at generation time or if additional LU's are required later, spare LU's may be assigned to BUS devices on-line using the FMGR LU command as follows:

```
:SYLU,LU #,BUS EQT #, DEVICE ADDRESS (BUS SUBCHANNEL)
```

TIME OUT CONSIDERATIONS

In RTE-IV systems, only one time out is assigned for the BUS and all BUS devices. This time out must be set to a value which will allow sufficient time for slow devices and devices which may pause during I/O to complete their operations. The bus time out may be assigned either at system generation time during the table generation phase or on-line via a FMGR command.

SYSTEM CONSIDERATIONS (CONTINUED)

Assigning the bus time out during generation will be discussed later. The following FMGR command may be used to observe and modify the bus time out:

:SYTO,BUS EQT	Displays present value of bus time out
:SYTO,BUS EQT,NEW TO	Modifies bus time out to new value specified

Note that time out units are displayed and specified in ten's of milleseconds.

In RTE-L systems, each device (including bus) is assigned its own time out value.

SESSION MONITOR CONSIDERATIONS

RTE-IVB users must be aware that a capability level of 60 is required to change time outs (SYTO) and to assign LU's (SYLU).

In order to access a device in the session environment, the device's session LU must be mapped to its system LU via the Session Switch Table. LU's may be added to the Session Switch Table in two ways. The first way is to use a FMGR command:

```
:SL,SESSION LU,SYSTEM LU
```

This command makes temporary additions to the Session Switch Table. When you log off, the temporary additions will be deleted. The user can also delete SST entries:

```
:SL,SESSION LU,-
```

The second way to add LU's to the SST is to ask the system manager to run the Accounts program and add them to your session. These additions are permanent. Every time you log on your SST will contain the SST entries specified in the system accounts file. Note that a capability level of 50 is required to add new session/system LU mappings to the Session Switch Table. The "SL" command by itself with no parameters will list the current SST.

SYSTEM CONSIDERATIONS (CONTINUED)

DMA AND BUFFERING

DMA and buffering may be assigned to the bus itself and to the individual bus devices. Both may be specified either as bus EQT parameters at generation time during the table generation phase or on-line using FMGR commands, EXEC calls, or bus message routines. Buffering the bus itself is not recommended since this precludes user error processing. DMA may be assigned to the bus EQT, however in RTE-IV systems only two DMA channels exist. If any other devices in the system use DMA exclusively or tie up the DMA channels for long periods of time, bus response time will significantly increase. In RTE-L systems, however, each I/O channel automatically employs DMA so all bus I/O is accomplished via DMA.

Individual bus devices may be assigned DMA even if the bus is not. The HP-IB driver automatically sets up DMA whenever an I/O operation is performed with a device configured for DMA regardless of whether or not the bus is configured for DMA. Individual devices, however, may only be configured for DMA on-line as specified above. Individual bus devices may also be buffered. Once again though, this precludes user processing of any I/O errors which may occur for the device. Each device reference manual should be consulted to determine if buffering and/or DMA is compatible with the device design, capabilities, and functions.

DEVICE CONSIDERATIONS

When HP-IB is configured into an RTE-IV system and devices connected to the bus, the HP-IB driver maintains a device configuration word for each device including the bus itself. This configuration word determines the format in which the driver will transmit data to the device, the format in which the driver will expect data from the device, whether the device will be buffered, whether the device will use DMA, and whether the device will be allowed to interrupt the bus to request service. The driver sets the configuration word with a default value at the time the bus is initialized. This default value should be adequate for most devices. Some devices, however, will require a different configuration. Device reference manuals should be consulted to determine if the default setting is satisfactory.

Refer to: AN 401 Series Application Notes
RTE-IV On-line Generator Manual
RTE-IVB Terminal User's Guide
Batch Spool Monitor Reference Manual (RTE-IVA)

OFF-LINE INTERFACE DIAGNOSTIC TROUBLESHOOTING MODULE

Purpose

TO ENABLE THE USER TO COMMUNICATE WITH THE INTERFACE BUS INTERFACE IN ORDER TO:

A. DIAGNOSE THE IBI

B. DIAGNOSE A DEVICE OR DEVICES ON THE BUS.

DIAGNOSTIC REQUIRES:

1. 2100 SERIES COMPUTER WITH AT LEAST 4K MEMORY
2. ABSOLUTE BINARY LOADER ROM COMPATIBLE WITH THE MEDIA FROM WHICH THE DIAGNOSTIC IS TO BE LOADED.
3. A CONSOLE
4. DMA-DCPC IS HIGHLY RECOMMENDED

THE DIAGNOSTIC IS SEPARATED INTO 4 AREAS:

1. CONFIGURATION –	OPERATOR ENTERS SELECT CODE, MYADDRESS CODE, AND SERVICE REQUEST ID
2. MAIN TEST –	VERIFIES PROPER OPERATION OF 59310 INTERFACE(S) AND TESTS FOR SHORTS IN ANY CONNECTING CABLING
3. CABLE TEST –	IF SELECTED AND IF TWO 59310 INTERFACES ARE UNDER TEST, VERIFYS THAT NO OPEN SIGNAL LINES EXIST IN CONNECTING CABLING. ALL DEVICES MUST BE DISCONNECTED.
4. TROUBLESHOOTING MODULE –	ALLOWS OPERATOR CONTROL AND MONITORING OF THE 59310 INTERFACE CARD AND HP-IB

TROUBLESHOOTING COMMANDS

GENERAL COMMAND FORMAT

ℓ [UP TO SIX OCTAL DIGITS]

EXAMPLE 1

: C53 (OCTAL 53)

DIAGNOSTIC PROMPT

SEND COMMAND "53" TO THE IBI

COMMANDS

C – SEND COMMAND TO IBI
D – SEND DATA ON BUS
S – PRINT IBI STATUS
I – PRINT INPUT BUS REGISTER OF IBI

COMMAND TERMINATION

1. CR-LF
2. WHEN THE SIXTH OCTAL DIGIT IS ENTERED
3. THE LETTER "E" IS ENTERED. THIS ABORTS THE CURRENT COMMAND AND A CR-LF IS ISSUED.

REFER TO THE DEVICE PROGRAMMING REFERENCE SHEETS, APPENDIX B, AND TO THE HP 59310 A/B INTERFACE BUS DIAGNOSTIC REFERENCE MANUAL (59310-90061).

ERROR MESSAGES

INVALID COMMANDS CAUSE THE MESSAGE

INPUT ERROR

TO BE PRINTED ON THE TERMINAL.

C AND D COMMANDS ➧ WILL NOT RECOGNIZE ANY NON-OCTAL CHARACTERS. FOR EXAMPLE,

```
:C49 DIAGNOSTIC ISSUES AUTOMATIC
CR-LF WHEN BAD DIGIT ENTERED.

INPUT ERROR
:
```

PRELIMINARY SYSTEM CONTROLLER INTERFACE COMMANDS

OCTAL	DECIMAL	ASCII	BUS FUNCTION
1	1	SOH	IFC
2	2	STX	REN FALSE
3	3	ETX	REN TRUE
40	32	SPACE	SET DATA MODE (ATN FALSE)
60	48	Ø	SET COMMAND MODE (ATN TRUE)

BUS SYSTEM UNDER STUDY

DEVICE NAME	BUS FUNCTION	MYADDR ON DEVICE (A5-A1)	TALK ADDRESS		LISTEN ADDRESS	
			OCTAL	ASCII	OCTAL	ASCII
59310B INTERFACE	SYSTEM CONTROLLER	0	100	@	40	SP
59309A DIGITAL CLOCK	TALK/LISTEN	21	121	Q	61	1
59304A NUMERIC DISPLAY	LISTEN	6	106	F	46	&

DEVICE ASCII PROGRAMMING CODES

DIGITAL CLOCK

CODE	OCTAL	FUNCTION
P	120	STOP CLOCK
T	124	START CLOCK
R	122	RESET CLOCK TO 01:01:00:00:00
S	123	UPDATE SECOND INTERVAL
M	115	UPDATE MINUTE INTERVAL
H	110	UPDATE HOUR INTERVAL
D	104	UPDATE DAY INTERVAL
C	103	RECORD TIME IN CLOCK OUTPUT REGISTER; VALUE IS OUTPUT WHEN ADDRESSED TO TALK. OTHER INPUTS NOT DISPLAYED.

NUMERIC DISPLAY

CHARACTERS DISPLAYED		
ASCII INPUT	CHARACTER DISPLAYED	
0-9	0-9	OVERFLOW:
E	E	WHEN LIT,
.	.	MORE THAN
—	—	12-DIGITS
+	(SPACE)	SENT WHERE
,	(SPACE)	MOST SIGNIF-
;	(SPACE)	ICANT DIG-
SP	(SPACE)	IT(S) LOST.

REFER TO THE DEVICE PROGRAMMING REFERENCE SHEETS, APPENDIX B, AND TO THE INDIVIDUAL DEVICE REFERENCE MANUALS.

PROGRAMMING THE CLOCK & DISPLAY

PROBLEM: PROGRAM THE CLOCK TO BEGIN TALKING STARTING AT DAY 1, HOUR 1, MINUTE 0, SEC 0 AND DISPLAY EACH SECOND ON THE NUMERIC DISPLAY.

SOLUTION:

:C1	IFC - OBSERVE BOTH DEVICES UNADDRESSED
:C60	COMMAND MODE
:D100	INTERFACE TALK
:D61	CLOCK LISTEN
:D46	DISPLAY LISTEN
:C40	DATA MODE
:D120	STOP CLOCK
:D122	RESET TO DAY 1, HOUR 1, SEC = MIN = 0
:D124	START CLOCK
:C60	COMMAND MODE
:D121	CLOCK TALK
:C40	DATA MODE

OBSERVE CLOCK OUTPUT APPEAR ON DISPLAY.

PROGRAMMING A SYSTEM WITH THE DIAGNOSTIC

THE PARAMETERS IN THE FOLLOWING TABLE INDICATE THE SWITCH SETTINGS ON THE I/O INTERFACE AND DEVICES, AND THE PROGRAM ADDRESSES USED.

SAMPLE CONFIGURATION

DEVICE NAME	MYADDR ON DEVICE (SWITCHES A5-A1)	PROGRAM ADDRESSES	
		TALK	LISTEN
59310B INTERFACE I/O	0	100	40
5150A PRINTER	25	---	65
59304A NUMERIC DISPLAY	12	112	52
3455A DIGITAL VOLTMETER	26	126	66

THIS EXAMPLE SETS UP THE DVM TO MAKE A RESISTANCE MEASUREMENT, TRIGGERS THE DVM TO TAKE THE MEASUREMENT, AND OUTPUTS THE READING TO A DISPLAY AND PRINTER.

REFER TO THE DEVICE PROGRAMMING REFERENCE SHEETS AND OCTAL/ASCII CONVERSION TABLE IN APPENDIX B.

COMPUTER I/O
TALKING TO MULTIMETER

EXAMPLE 1:

:C1	SET IFC (INTERFACE CLEAR)
:C60	SET ATN TRUE (COMMAND MODE)
:C3	SET REN (REMOTE ENABLE)
:D100	CONTROLLER TALK ADDRESS
:D66	MULTIMETER LISTEN ADDRESS
:C40	SET ATN FALSE (DATA MODE)
:D106	PROGRAM DVM TO 2 WIRE RESISTANCE MEAS.
:D64	
:D122	PROGRAM DVM TO AUTO-RANGE
:D67	
:D124	PROGRAM DVM TO HOLD/MANUAL TRIGGER
:D63	
:D124	TRIGGER MULTIMETER TO TAKE READING
:D63	
:C60	SET ATN TRUE (COMMAND MODE)
:D52	DISPLAY LISTEN ADDRESS
:D65	PRINTER LISTEN ADDRESS
:D126	MULTIMETER TALK ADDRESS
:C40	SET ATN FALSE (DATA MODE)

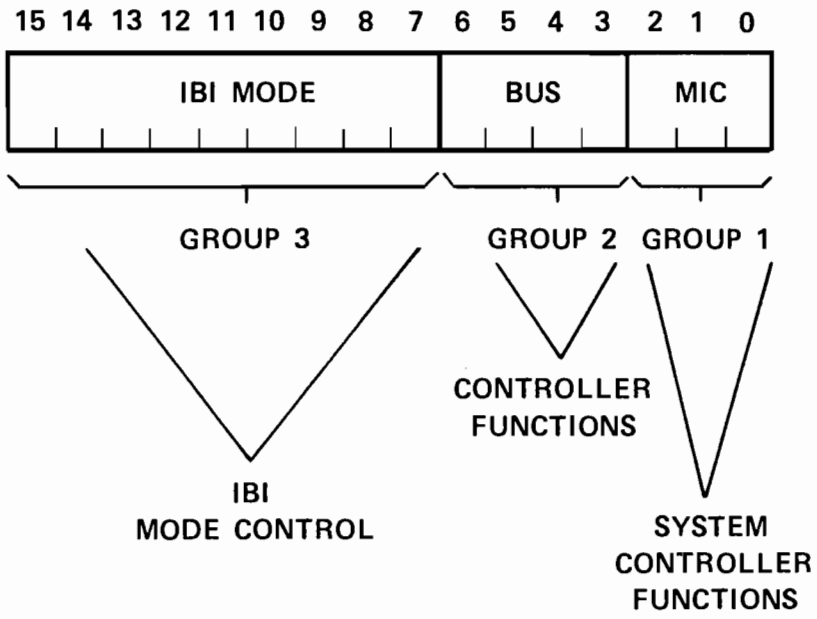
59310 INTERFACE CONCEPTS

BUS LINES	INTERFACE'S ROLE
5 BUS MANAGEMENT LINES	CONTROL
3 HANDSHAKE LINES	NON-CONTROL
8 DATA LINES	CONTROL

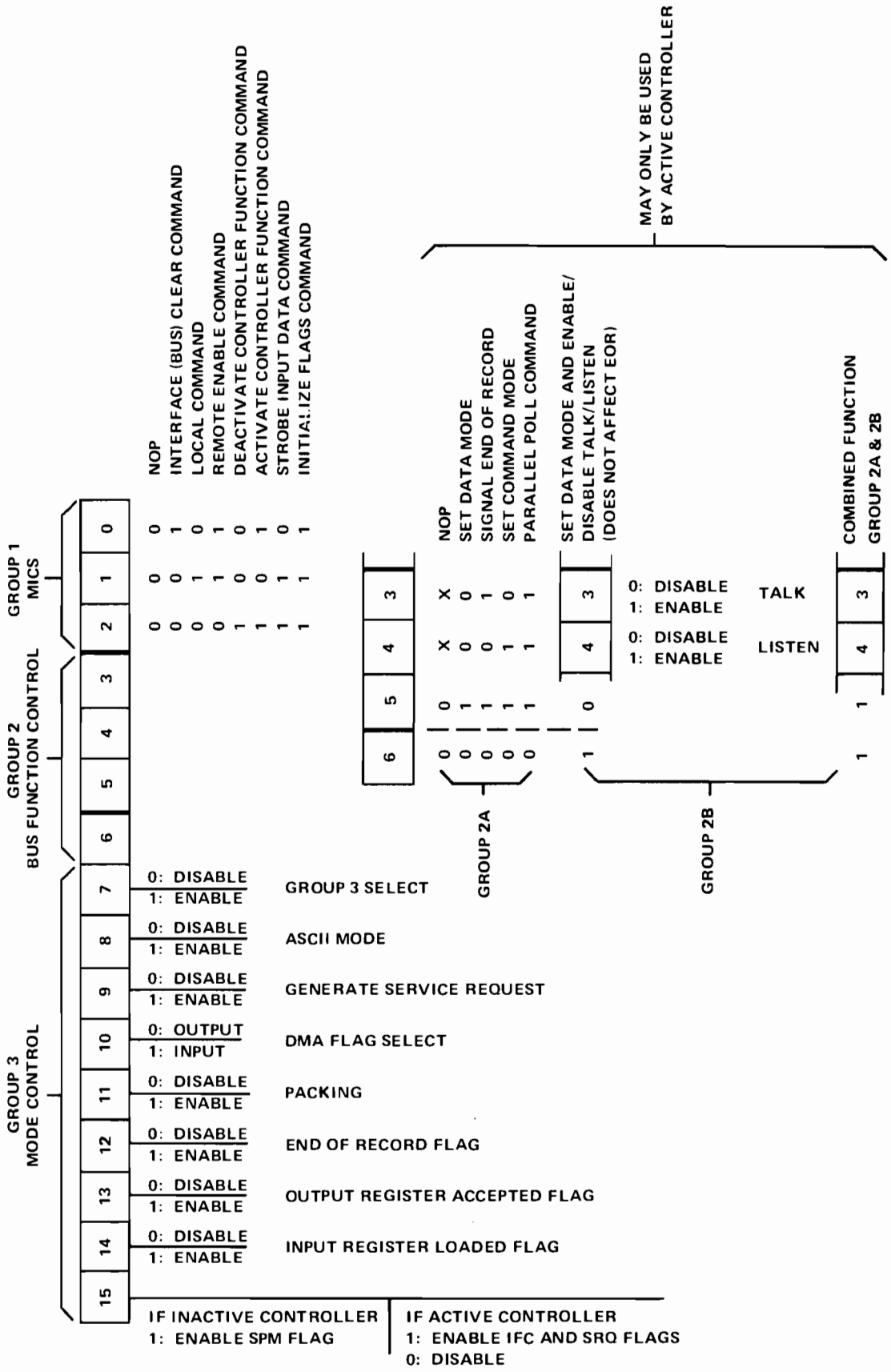
RECALL

THE 59310 INTERFACE BUS INTERFACE IS THE SYSTEM CONTROLLER AS WELL AS ANOTHER DEVICE ON THE BUS.

IBI CONTROL WORD FORMAT



IBI CONTROL WORD BREAKDOWN



MACHINE LANGUAGE PROGRAMMING SEQUENCES

SEND CONTROL WORD TO I/O CARD:

```
    LDA     CTLWD
    STF     IBI         PREPARE CARD TO ACCEPT CONTROL/WORD

    OTA     IBI         OUTPUT IT TO OWR
```

OUTPUT DATA TO I/O CARD:

```
    LDA     DATA        LOAD DATA
    OTA     IBI,C       OUTPUT TO OWR
```

OBTAIN STATUS FROM I/O CARD:

```
    STF     IBI         PREPARE CARD TO SEND STATUS
    LIA     IBI         LOAD STATUS
```

INPUT DATA WORD FROM I/O CARD:

```
    LIA     IBI,C
```

```
IBI     EQU     12B
DATA    OCT     5426
CTLWD   OCT     7
```

STC — ENABLES INTERRUPTS AND PREVENTS SUBSEQUENT CLF FUNCTIONS FROM CLEARING ALL BUT DMA OUTPUT REQUEST FLAG.

CLC — PREVENTS INTERRUPTS AND ENABLES SUBSEQUENT CLF FUNCTIONS TO CLEAR MAIN, EOR AND IFC FLAGS.

CLF — WILL FUNCTION AS A STF UNLESS EXECUTED IN CONJUNCTION WITH ANOTHER I/O INSTRUCTION (I.E., CLC IBI, C)

OBTAIN STATUS

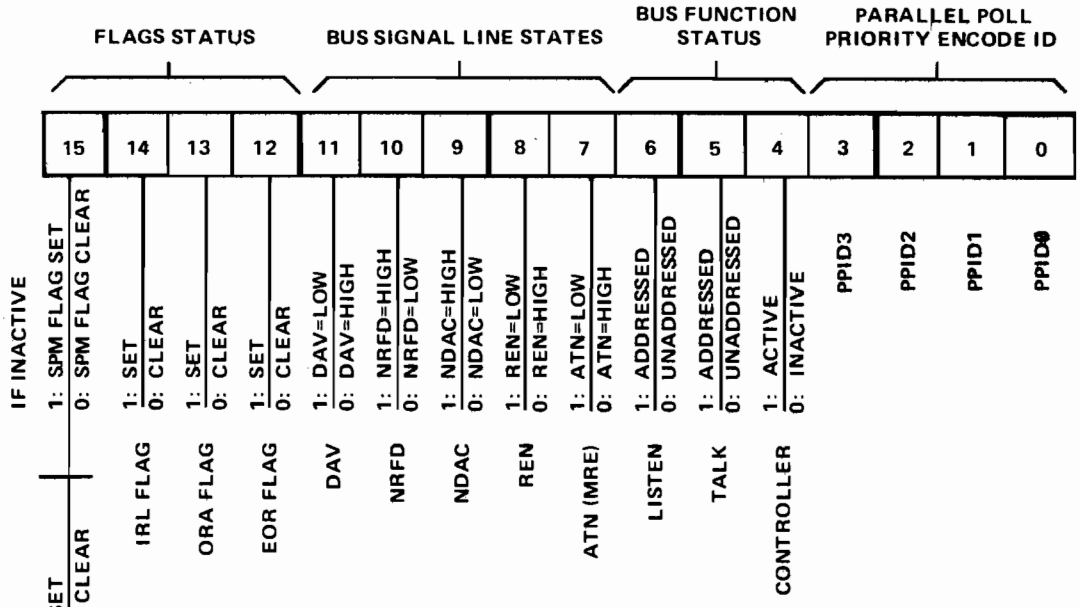
☆ PURPOSE: PROVIDES STATUS INFORMATION ABOUT THE BUS

☆ FORMAT:

STF	IBI	TELL BOARD STATUS REQ. COMING
LIA	IBI	GET STATUS FROM BOARD

✸ NOTE: OTA/B MAY NOT BE GIVEN BETWEEN STF AND LIA/B

STATUS WORD BREAKDOWN



PARALLEL POLL PRIORITY ENCODED ID

BUS SIGNAL LINES									STATUS WORD BITS			
IFC	DIO8	DIO7	DIO6	DIO5	DIO4	DIO3	DIO2	DIO1	BIT3	BIT2	BIT1	BIT0
H	H	H	H	H	H	H	H	H	0	0	0	0
H	H	H	H	H	H	H	H	L	0	0	0	1
H	H	H	H	H	H	H	L	X	0	0	1	0
H	H	H	H	H	H	L	X	X	0	0	1	1
H	H	H	H	H	L	X	X	X	0	1	0	0
H	H	H	H	L	X	X	X	X	0	1	0	1
H	H	H	L	X	X	X	X	X	0	1	1	0
H	H	L	X	X	X	X	X	X	0	1	1	1
H	L	X	X	X	X	X	X	X	1	0	0	0
L	X	X	X	X	X	X	X	X	1	0	0	1

CONTROL WORD TO BUS I/O

GROUP 1			
Octal	Decimal	ASCII Char	Function
1	1	SOH	IFC
2	2	STX	Local Command
3	3	ETX	REN
4	4	EOT	Deactivate Controller
5	5	ENQ	Activate Controller
6	6	ACK	Strobe Input Data
7	7	BEL	Initialize Flags
GROUP 2A Used To Control Controller			
40	32	SPACE	Set Data Mode
50	40	(	Set EOI
60	48	0	Set ATN
70	56	8	Parallel Poll
GROUP 2B Controls Talker/Listener Functions			
100	64	@	Disable Listen & Talk
110	72	H	Set Talk
120	80	P	Set Listen
130	88	X	Set Talk & Listen
GROUP 3 Mode Control			
600			ASCII Mode Enable
1200			SRQ Enable (If board is not controller)
2200			DMA Flag Select
4200			Packing Enable
10200			EOR Flag Enable
20200			OWRA Flag Enable
40200			IRL Flag Enable
100200			SPM Flag Enable (If Inactive Controller) IFC and SRQ Enable (If Active Controller)

NOTE: These commands are sent to the interface first. Any resulting action on the bus depends on the command.

BUS: IN COMMAND MODE RECEIVING AN OCTAL CODE OVER THE DATA LINES.

	COMMAND	ASCII Character	OCTAL CODE	PURPOSE
UNADDRESS COMMANDS	UNL UNLISTEN	?	077	Clears Bus of all listeners.
	UNT UNTALK	—	137	Unaddresses the current talker so that no talker remains on the Bus.*
UNIVERSAL COMMANDS	LLO Local Lockout	DC1	021	Disables front panel local-reset button on responding devices.
	DCL Device Clear	DC4	024	Returns all devices capable of responding to pre-determined states, regardless of whether they are addressed or not.
	PPU Parallel Poll Unconfigure	NAK	025	Sets all devices on the HP-IB with Parallel Poll capability to a predefined condition.
	SPE Serial Poll Enable	CAN	030	Enables Serial Poll Mode on the Bus.
	SPD Serial Poll Disable	EM	031	Disables Serial Poll Mode on the Bus.
ADDRESSED COMMANDS	SDC Selective Device Clear	EOT	004	Returns addressed devices, capable of responding to pre-determined states.
	GTL Go to Local	SOH	001	Returns responding devices to local control.
	GET Group Execute Trigger	BS	010	Initiates a simultaneous pre-programmed action by responding devices.
	PPC Parallel Poll Configure	ENQ	005	This command permits the DIO lines to be assigned to instruments on the Bus for the purpose of responding to a parallel poll.
	TCT Take Control	HT	011	This command is given when the active controller on the Bus transfers control to another instrument.

*NOTE
Talkers can also be unaddressed by transmitting an unused talk address on the Bus.

SUMMARY OF BUS COMMANDS

APPENDIX A

BASIC BRANCH & MNEMONIC TABLE GENERATION

BRANCH AND MNEMONIC TABLES

FUNCTION

The Branch and Mnemonic tables provide the necessary links between subroutines in memory and the BASIC subroutine call. If you plan to use any subroutines or functions, then these tables must be loaded prior to entering BASIC statements.

To load Branch & Mnemonic Tables:

>TABLES BTBL, MTBL

Where BTBL and MTBL are the names of the Branch & Mnemonic tables specified in the table generator answer file.

BASIC BRANCH & MNEMONIC TABLE GENERATION

PURPOSE

TO GENERATE BRANCH AND MNEMONIC TABLES FOR LINKING OF SUBROUTINES AND FUNCTIONS TO THE BASIC INTERPRETER.

ADVANTAGES

ON-LINE CONFIGURATION UNDER RTE.

CONVERSATIONAL OR COMMAND INPUT FILES.

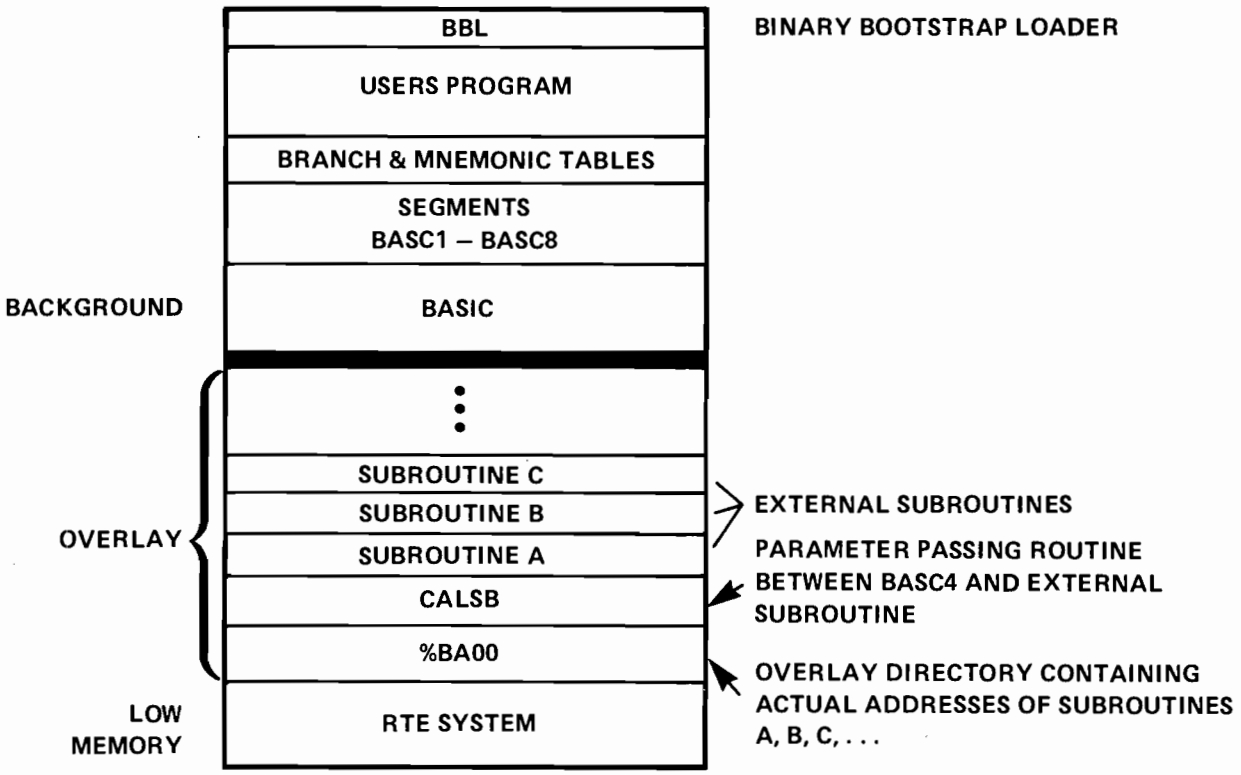
IMPLEMENTATION OF USERWRITTEN SUBROUTINES IN FORTRAN, ALGOL, OR ASSEMBLY.

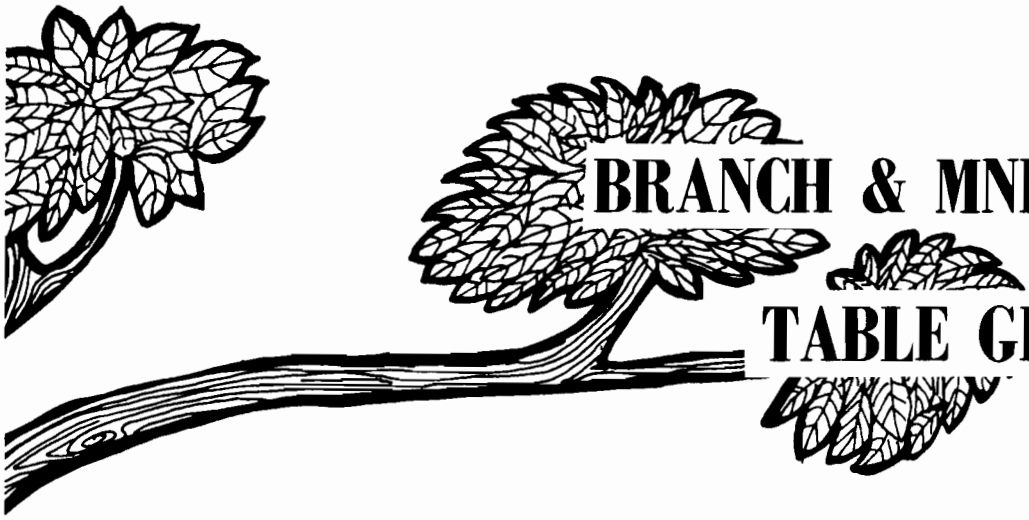
CONFIGURED SUBROUTINES ARE IN ABSOLUTE MACHINE CODE.

TABLE GENERATOR

FUNCTION

TO PRODUCE A BRANCH TABLE, MNEMONIC TABLE, OVERLAY DIRECTORIES, AND A TRANSFER FILE TO LOAD THE OVERLAYS.





BRANCH & MNEMONIC TABLE GENERATION

THE FOLLOWING SLIDES WILL ILLUSTRATE AN EXAMPLE OF A TYPICAL PROCEDURE FOR GENERATING BASIC BRANCH AND MNEMONIC TABLES.

TABLE GENERATOR INPUT

INITIAL CONTROL STATEMENT

THE FIRST ENTRY IN THE COMMAND INPUT FILE SPECIFIES FILE NAMES AND OVERLAY IDENTIFICATION FOR THE OUTPUT FILES.

EXAMPLE

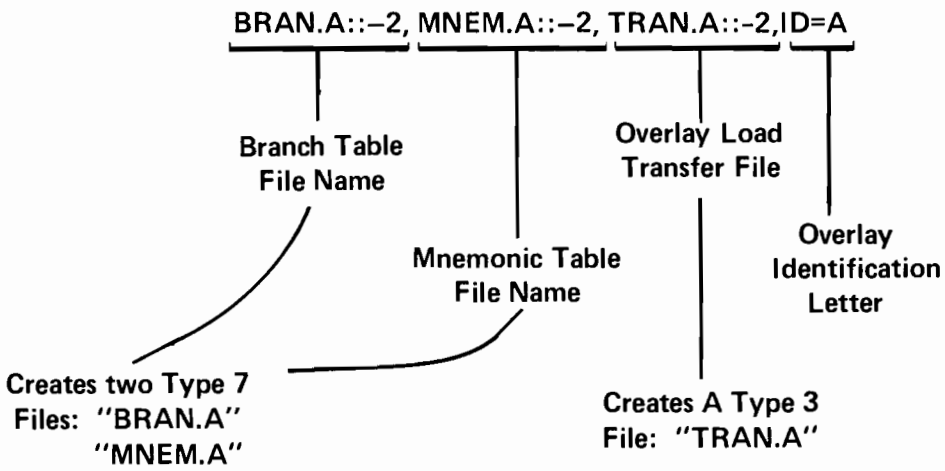
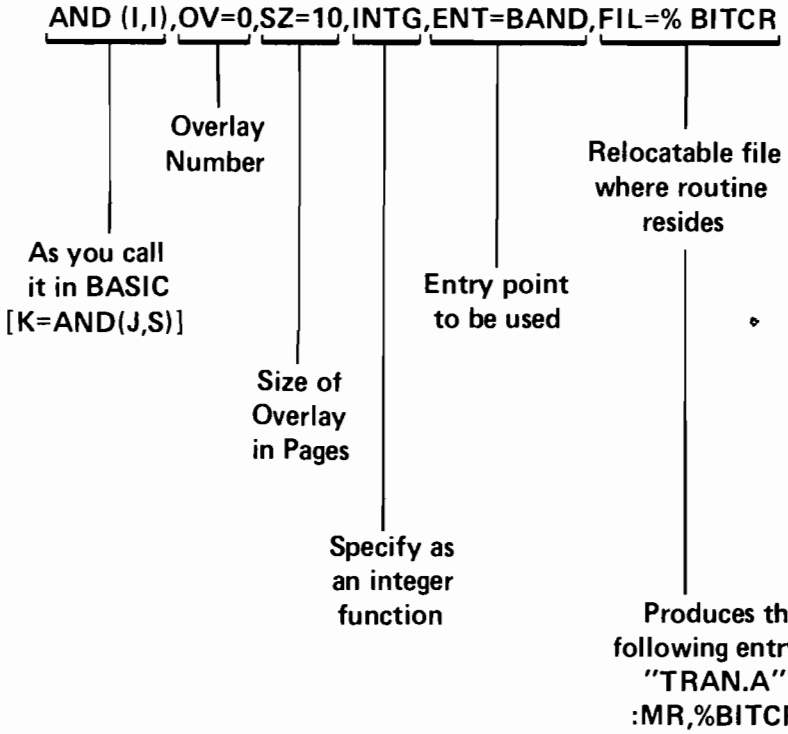


TABLE GENERATION (CONT.)

✱ ROUTINE SPECIFICATION ENTRY

ALL REMAINING ENTRIES IN THE COMMAND INPUT FILE SPECIFY INDIVIDUAL CHARACTERISTICS OF EACH ROUTINE TO BE CONFIGURED.

✱ EXAMPLE



COMMAND INPUT FILE LISTING

FILE NAME: TABLES

Initial Control Statement

```
BTBL::2,MTBL::2,TRFL::2,ID=A
TRIGR(I),          OV=1,SZ=4,FIL=%MESS
CLEAR(I,I),        OV=1,SZ=4,FIL=%MESS
RMOTE(I),          OV=1,SZ=4,FIL=%MESS
GTL(I),            OV=1,SZ=4,FIL=%MESS
LLO(I),            OV=1,SZ=4,FIL=%MESS
LOCL(I),           OV=1,SZ=4,FIL=%MESS
STATS(I,IV),       OV=1,SZ=4,FIL=%MESS
PPOLL(I,I,I),      OV=1,SZ=4,FIL=%MESS
PSTAT(I,IV),       OV=1,SZ=4,FIL=%MESS
CNFG(I,I,I),       OV=1,SZ=4,FIL=%MESS
ABRT(I,I),         OV=1,SZ=4,FIL=%MESS
HPIB(I,I,I),       OV=2,SS,SZ=4,      ENT=HPIB,    FIL=%HPIB
CMDR(I,RA,RVA),    OV=2,SS,SZ=4,      ENT=CMDR,    FIL=%HPIB
CMDW(I,RA,RA),     OV=2,SS,SZ=4,      ENT=CMDW,    FIL=%HPIB
IBERR(I),          OV=2,SS,SZ=4,INTG,ENT=IBERR,   FIL=%HPIB
IBSTS(I),          OV=2,SS,SZ=4,INTG,ENT=IBSTS,   FIL=%HPIB
SRQ(I,I,RA),       OV=2,SS,SZ=4,      ENT=SRQ,     FIL=%HPIB
SRQSN(I,I),        OV=2,SS,SZ=4,      ENT=SRQSN,   FIL=%HPIB
DCODE(RVA,RVA,RA),    OV=3,SZ=4,BP,       ENT=DCODE
NUM(RA),              OV=4,SZ=4,INTG,     ENT=NUM
CHRS(I,RVA),          OV=4,SZ=4,          ENT=CHRS
IBSET(I,I),           OV=4,SZ=4,INTG,     ENT=BBSET
IEOR(I,I),            OV=4,SZ=4,INTG,     ENT=BEOR
OR(I,I),              OV=4,SZ=4,INTG,     ENT=BIOR
AND(I,I),             OV=4,SZ=4,INTG,     ENT=BAND
NOT(I),               OV=4,SZ=4,INTG,     ENT=BNOT
ISHFT(I,I),           OV=4,SZ=4,INTG,     ENT=BSHFT
IBTST(I,I),           OV=4,SZ=4,INTG,     ENT=BBTST
IBCLR(I,I),           OV=4,SZ=4,INTG,      ENT=BBCLR
ISETC(RA),            OV=4,SZ=4,INTG,      ENT=ISETC
DEB$(RVA),            OV=4,SZ=4,           ENT=DEB$
BLK$(I,RVA),          OV=4,SZ=4,           ENT=BLK$
```

PARAMETER TYPE

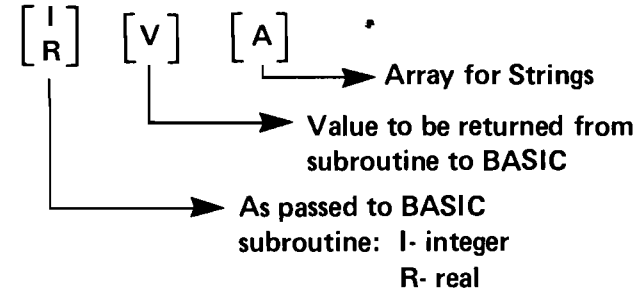


TABLE GENERATION PHASES 1, 2, 3: PREPARATION & CLEAN-UP

PHASE 1: REMOVING PREVIOUS OVERLAYS

IF THE SAME OVERLAYS OR OVERLAYS WITH THE SAME ID SEGMENT AS EXISTENT OVERLAYS ARE TO BE LOADED/RE-LOADED, THE LOADER MUST BE RUN TO REMOVE EACH EXISTING OVERLAY:

```
:RU,LOADR,,,,PU
 /LOADR:   PNAME ?%BA00
 /LOADR:$END
:RU,LOADR,,,,PU
 /LOADR:   PNAME ?%BA01
 /LOADR:$END
:RU,LOADR,,,,PU
 /LOADR:   PNAME ?%BA02
 /LOADR:$END
:RU,LOADR,,,,PU
 /LOADR:   PNAME ?%BA03
 /LOADR:$END
:RU,LOADR,,,,PU
 /LOADR:   PNAME ?%BA04
 /LOADR:$END
```

PHASE 2: PURGING PREVIOUS FMGR OUTPUT FILES

IF THE SAME FILE NAMES SPECIFIED IN THE INITIAL CONTROL STATEMENT OF THE TABLE GENERATOR COMMAND FILE ARE TO BE USED, YOU MUST FIRST PURGE EACH EXISTENT FILE.

```
:PU,%BA00
:PU,%BA01
:PU,%BA02
:PU,%BA03
:PU,%BA04
:PU,BTBL
:PU,MTBL
:PU,TRFL
```

PHASE 3: INSURE FMGR INTEGRITY

CHECK THE DISC CARTRIDGE WHERE THE OUTPUT FILES ARE TO BE CREATED FOR FATAL CONDITIONS AND RE-PACK:

```
:PK,-2
:DL,-2
```

TABLE GENERATION CONT.

PHASES 4, 5: OVERLAY GENERATION

PHASE 4: GENERATE OUTPUT FILES WITH TABLE GENERATOR

RUN THE PROGRAM "RTETG" TO GENERATE ALL OUTPUT FILES:

```
:RU,RTETG,TA,BL,ES
$END RTETG
```

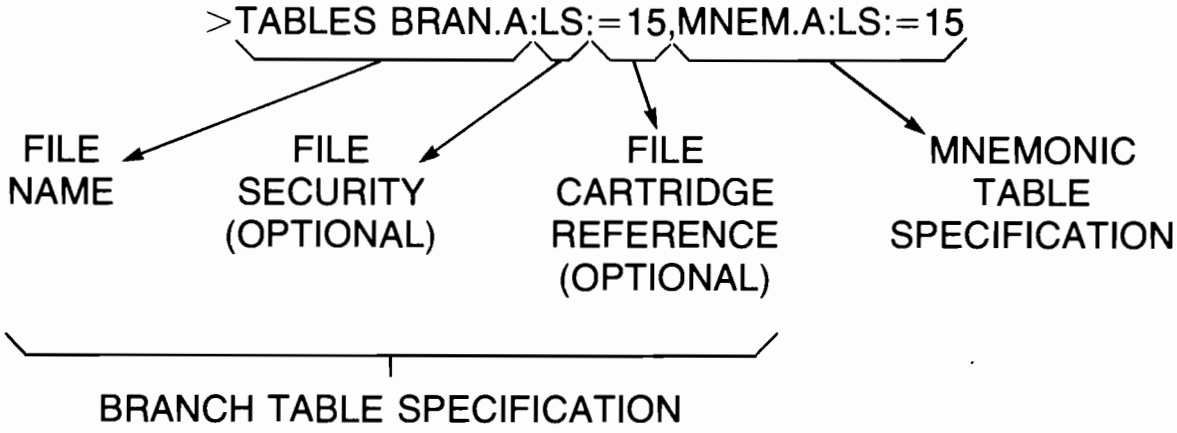
PHASE 5: LOADING THE OVERLAYS INTO THE SYSTEM

THIS IS THE FINAL PHASE TO PRODUCE THE FOREGROUND DISC-RESIDENT OVERLAY MODULES. TRANSFER TO THE THIRD FILE NAME SPECIFIED ON THE INITIAL CONTROL STATEMENT IN THE COMMAND FILE:

```
:PU,##.RTG
: ST, %BA01 :       0:       0,##.RTG,BR
: DU, %MESS :       0:       0,##.RTG,BR,2, 99
:RU,LOADR:IH,,##.RTG,6, RT   , PE,,   4
 /LOADR:%BA01  READY
 /LOADR:$END
:PU,##.RTG
: ST, %BA02 :       0:       0,##.RTG,BR
: DU, %HPIB :       0:       0,##.RTG,BR,2, 99
:RU,LOADR:IH,,##.RTG,6, RTSS, PE,,   4
 /LOADR:%BA02  READY
 /LOADR:$END
:PU,##.RTG
: ST, %BA03 :       0:       0,##.RTG,BR
:RU,LOADR:IH,##.RTG,6,  BGSS, PE,,   6
 /LOADR:%BA03  READY
 /LOADR:$END
:PU,##.RTG
: ST, %BA04 :       0:       0,##.RTG,BR
:RU,LOADR:IH,,##.RTG,6, RTSS, PE,,   4
 /LOADR:%BA04  READY
 /LOADR:$END
:PU,##.RTG
:SP,%BA01
:SP,#BA02        LOAD OVERLAYS
:SP,#BA03        AS PERMANENT
:SP,#BA04        PROGRAMS
::
```

BRANCH AND MNEMONIC TABLE LOADING

SPECIFYING THE TABLES: TYPE 7 FMGR FILES



ERRORS

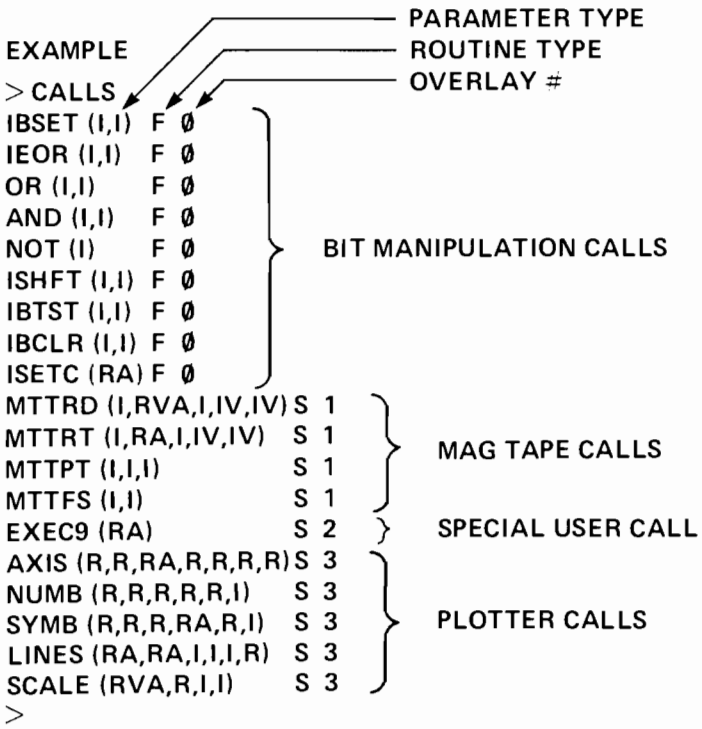
IF THE TABLES CANNOT BE FOUND, FMGR WILL SEND THE APPROPRIATE ERROR MESSAGE WHICH IS RE-FORMATTED BY BASIC BEFORE BEING PRINTED

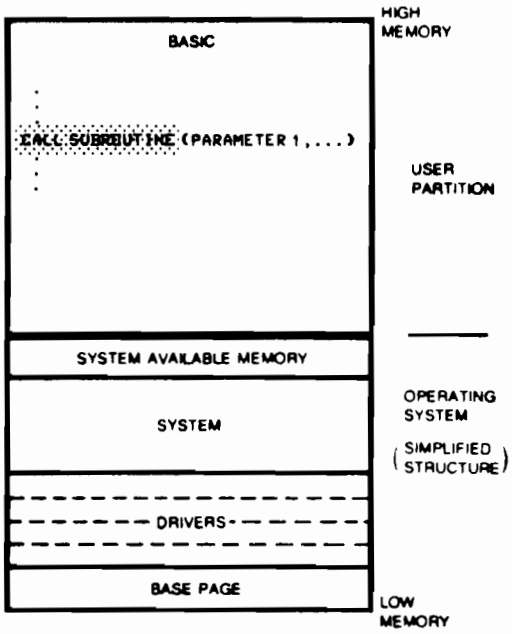
LISTING PROGRAM CALLS

PURPOSE

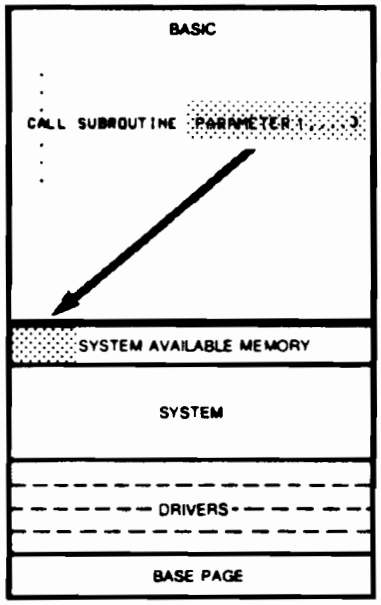
TO PROVIDE A LIST OF INFORMATION ABOUT ALL SUBROUTINE AND FUNCTION CALLS.

EXAMPLE

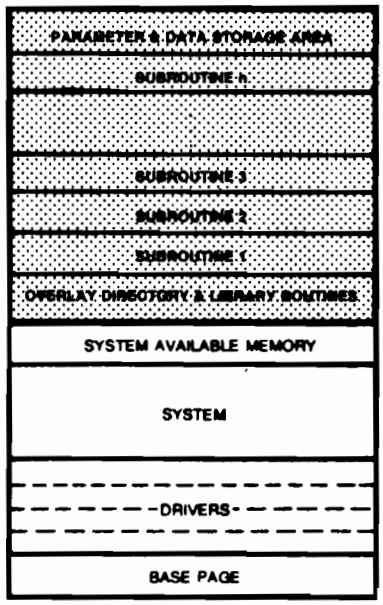




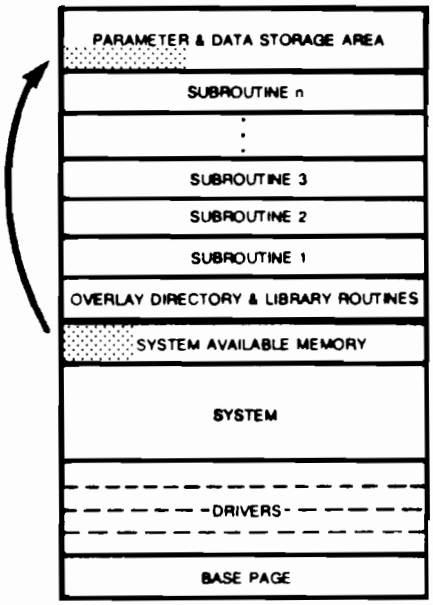
1 When a subroutine or FUNCTION subprogram is encountered in BASIC, the following procedure occurs.



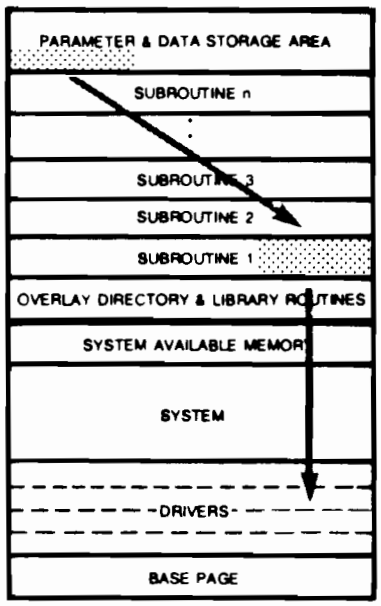
2. The subroutine call or FUNCTION subprogram is referenced in the Branch and Mnemonic Tables to determine which overlay is required, and what format the parameters are expected to be. The parameters are then stored in System Available Memory (S A M.) in the required format.



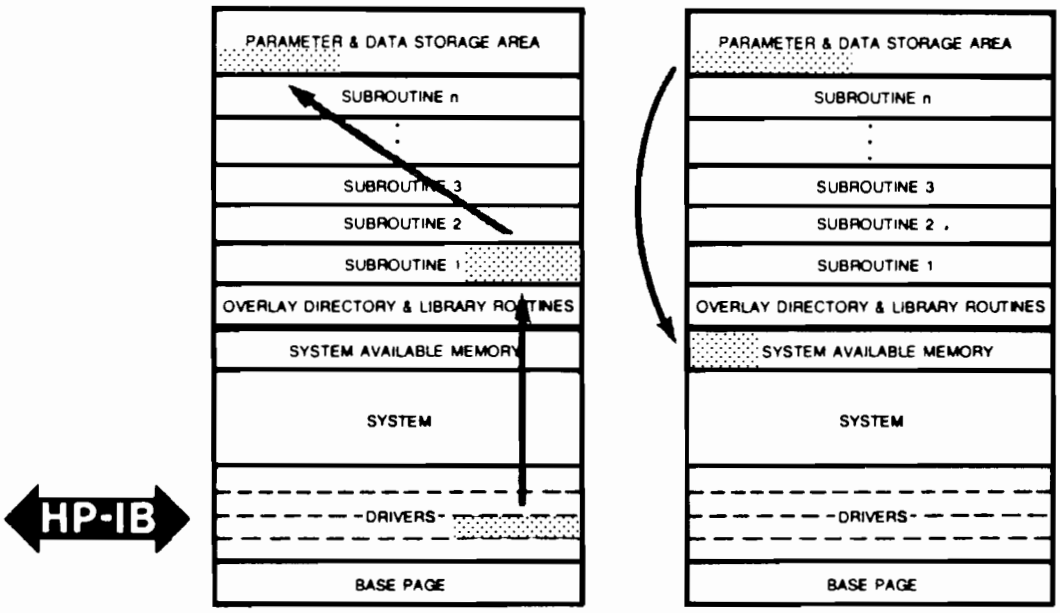
3. The overlay containing the subroutine or FUNCTION subprogram is brought into memory. If a free partition large enough to hold the overlay is available it will be used. Otherwise, BASIC could be swapped out to the disc.



4 The parameters stored in S A M are copied to a parameter storage area in the overlay.

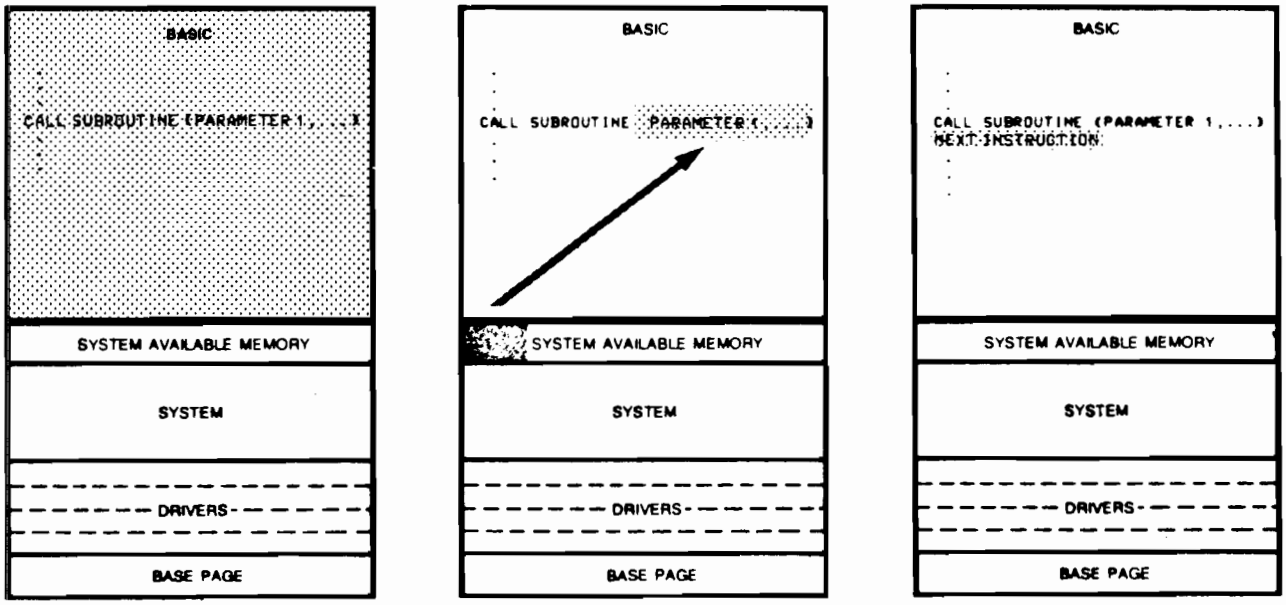


5 The parameters are handled by the device subroutine to form instructions for the device. These instructions are passed to the driver which regulates the flow of information out to the device.



6. Incoming information from a device is regulated by the driver and passed to the device subroutine. The device subroutine performs scaling and error checking of the incoming data, and stores it back in the parameter storage area.

7. The parameters stored in the parameter storage area of the overlay are copied back into S.A.M.



8. Control is returned to BASIC. If BASIC was swapped out, it is brought back into memory.

9. The parameters stored in S.A.M. are copied back to the respective parameters in the BASIC program.

10. The next BASIC instruction line is executed.

The "Ten Steps" Performed When a Subroutine or FUNCTION Subprogram is Called by BASIC

APPENDIX B

HP-IB DEVICES PROGRAMMING REFERENCE SHEETS

3495A SCANNER

HP-IB DEVICE PROGRAMMING REFERENCE SHEET
DATA SYSTEMS

BUS FUNCTIONS: Addressable Listener

DEVICE PROGRAMMING CODES

INSTRUCTION	ASCII CHARACTER	DECIMAL
Digit	0,1,2,3,4,5,6,7,8,9	48 thru 57
Space	SP	32
Clear	C	67
Execute	Carriage return (CR), E	13 , 69
No operator	NUL, DEL	0 , 127
Delimiter	Any other character	1 thru 126*

INSTRUCTION FORMATTING

1. The basic format for a channel-close instruction is:

   |← F →|

   $T_1\ U_1\ T_2\ U_2\ .\ .\ T_n\ U_n$ E

   Where T = Tens channel digit (decade select)
   U = Units channel digit (channel select)
   E = Execute or Carriage-Return (CR)
   F = Instruction field (shaded)

   For example, the instruction:

   07 35E

   Will close channels 07 and 35

2. The "Clear" instruction ("C") immediately opens all channels. An "Execute" ("E") instruction is not needed.

   Example: 24C Opens all channels.
   C24E Insures that only Channel 24 is closed.

The group execute trigger command (GET), may be used via the TRIGR subroutine to execute channel closures in place of "E" if the scanner is addressed to listen.

A space or other delimiter may be used to separate channel closure codes in the scanner programming sequence in order to enhance program readability.

3455A DVM

HP-IB DEVICE PROGRAMMING REFERENCE SHEET
DATA SYSTEMS

BUS FUNCTIONS: Listener, Talker, Service Request, Remote-Local

OUTPUT FORMAT: +/– D . D D D D D D E D D CRLF

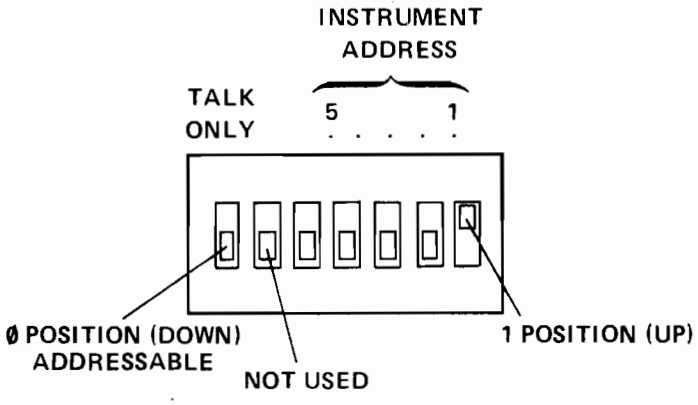
DEVICE PROGRAMMING CODES

	Control	Program Code
FUNCTION	DC Volts	F1
	AC Volts	F2
	Fast AC Volts	F3
	2 Wire kΩ	F4
	4 Wire kΩ	F5
	Test	F6
RANGE	.1	R1
	1	R2
	10	R3
	100	R4
	1 K	R5
	10 K	R6
	AUTO	R7
TRIGGER	Internal	T1
	External	T2
	Hold/Manual	T3
MATH	Scale	M1
	Error	M2
	Off	M3
ENTER	Y	EY
	Z	EZ
STORE	Y	SY
	Z	SZ
AUTO CAL	Off	A0
	On	A1
HIGH RESOLUTION	Off	H0
	On	H1
DATA READY RQS	Off	D0
	On	D1
BINARY PROGRAM		B

STATUS BYTE CODES:

ASCII CHAR	Decimal Code	
A	65	Data Ready – Indicates to the controller that measurement data is available. Applies to DATA READY Request feature.
B	66	Syntax Error – Indicates improper program code. Example - Program Code "F7" would cause a syntax error since the FUNCTION program set is only defined for codes F1 through F6.
D	68	BINARY FUNCTION Error – Indicates improper BINARY PROGRAM code or incomplete binary message. Similar to syntax error.
H	72	Trigger too Fast – Indicates the 3455A has been triggered while measurement data is being output to the bus. Warns of possible incorrect measurement information.

DEVICE ADDRESS SWITCHES



HP-IB DEVICE PROGRAMMING REFERENCE SHEET
DATA SYSTEMS

59304A NUMERIC DISPLAY

BUS FUNCTIONS: Addressable Listener

DEVICE PROGRAMMING CODES

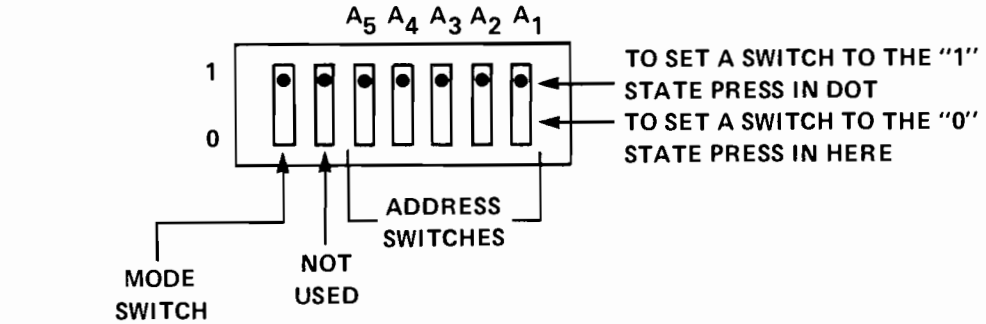
DIO Lines 7	6	5	4	3	2	1	Octal Equiv.	ASCII Equivalent	Displayed Character
0	1	1	0	0	0	0	060	0	0
0	1	1	0	0	0	1	061	1	1
0	1	1	0	0	1	0	062	2	2
0	1	1	0	0	1	1	063	3	3
0	1	1	0	1	0	0	064	4	4
0	1	1	0	1	0	1	065	5	5
0	1	1	0	1	1	0	066	6	6
0	1	1	0	1	1	1	067	7	7
0	1	1	1	0	0	0	070	8	8
0	1	1	1	0	0	1	071	9	9
1	0	0	0	1	0	1	105	E	E
0	1	0	1	1	1	0	056	. (decimal point)	.
0	1	0	1	0	1	1	053	+(plus sign)	Space
0	1	0	1	1	0	0	054	, (comma)	Space
0	1	1	1	0	1	0	072	: (colon)	Space
0	1	0	1	1	0	1	055	- (minus sign)	-
0	1	0	0	0	0	0	040	SP (Space)	Space

Note: Any other inputs are not recognized and are not displayed.

Device Errors:

1. OVERFLOW indicator light illuminated when the 12-digit capacity of the display has been exceeded. Most significant digit(s) are lost.
2. Any character sent which is not on the above chart clears the display to ASCII blanks.
3. Two successive periods are changed to one period.

DEVICE ADDRESS SWITCH



(ADDRESSABLE = "1" STATE)
(LISTEN ONLY = "0" STATE; display all data on bus)

The Numeric Display requires a CRLF before any data sent to it will be displayed.

HP-IB DEVICE PROGRAMMING REFERENCE SHEET
DATA SYSTEMS

59309A DIGITAL CLOCK

BUS FUNCTIONS: Addressable Listener, Talker

DEVICE PROGRAMMING CODES

When it is addressed to Listen, the 59309A responds to these ASCII codes:

P	Stops the clock
T	Starts the clock
R	Resets the clock to 01:01:00:00:00
S	Updates counting chain 1 second
M	Updates counting chain 1 minute
H	Updates counting chain 1 hour
D	Updates counting chain 1 day
C	Records time command C is accepted and stores time value in output register; value is output when 59309A is addressed to Talk

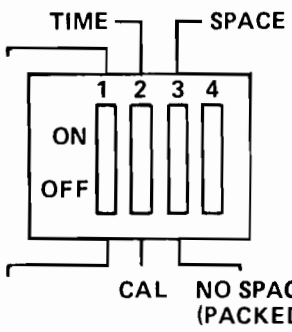
Device Errors:

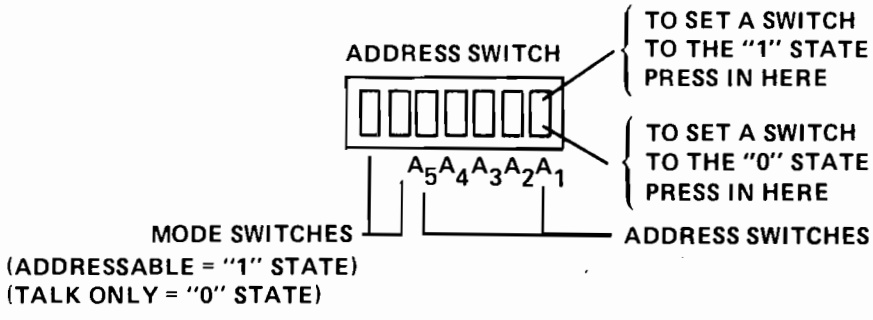
1. Display decimal points all light to indicate possible timebase tick miss or power supply has been glitched.
2. Status character "?" appears as first byte in data string.

For quick reference, refer to the label on the bottom of the clock.

DEVICE ADDRESS SWITCHES

FORMAT SWITCH (INSIDE TOP COVER)





59309A DIGITAL CLOCK (CONTINUED)

CLOCK OUTPUT FORMAT:

a. SPACE (not packed)

1. TIME (Time Only)

: (Colon) (Status)* (SP) 1 1 : 2 3 : 1 4 CR (LF)

, (Comma) (Status)* (SP) 1 1 , 2 3 , 1 4 CR (LF)

2. CAL (Calendar and Time)

: (Colon) (Status)* (SP) 1 2 : 2 8 : 1 1 : 2 3 : 1 4 (CR) (LF)

, (Comma) (Status)* (SP) 1 2 , 2 8 , 1 1 , 2 3 , 1 4 (CR) (LF)

b. NO SPACE (packed)

1. TIME (Time Only)

: (Colon) (Status)* (SP) 1 1 2 3 1 4 CR (LF)

, (Comma) (Status)* (SP) 1 1 2 3 1 4 CR (LF)

2. CAL (Calendar and Time)

: (Colon) (Status)* (SP) 1 2/2 8/1 1 2 3 1 4 (CR) (LF)

, (Comma) (Status)* (SP) 1 2 2 8 1 1 2 3 1 4 (CR) (LF)

*The ASCII character in this position of the data output string will be either ? or (SP) depending on the error status.

HP-IB DEVICE PROGRAMMING REFERENCE SHEET
DATA SYSTEMS

59308A TIMING GENERATOR

BUS FUNCTIONS: Listener, Talker, Service Request, Remote-Local

DEVICE PROGRAMMING CODES

Time in microseconds: [±] DDD [E±] D
(Note: bracketed entries are optional.)

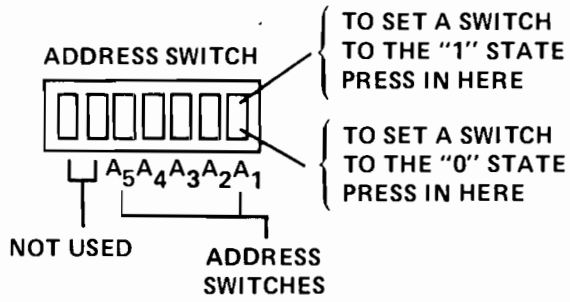
FUNCTION	ASCII	OCTAL	DECIMAL
Pacer	P	120	80
Timer	T	124	84
Trigger/Reset Command . .	R	122	82
Enable rear panel trigger . .	U	125	85
Disable rear panel trigger . .	A	101	65
Enable Service Request . . .	S	123	83
Disable Service Request. . .	D	104	68

Output is the number of timing periods since the generator was last triggered.

Output Format (space)(space) DDDDDD (CR) (LF)

With SRQ enabled, the generator asserts the SRQ line at the end of each timing period.

DEVICE ADDRESS SWITCHES



59308A TIMING GENERATOR CONTINUED

Thumbwheel Switches: Select time interval in microseconds.
Range, 001E0 to 999E8 μs.

Example Interval Settings*

Time Interval	Thumbwheel Setting
1 μs	001E0
100 μs	100E0
1 ms	001E3
100 ms	100E3
1 sec	001E6
100 sec	100E6
1 min	060E6
1 hour	036E8
1 day	864E8

*Square wave setting cannot be used if the exponent is 0.

The leading space becomes a 0 if the generator overflows, i.e. if more than 999999 timing periods have occurred since the last trigger.

PROGRAMMING SEQUENCE:

1. Send untlk, unlsn____?
2. Send remote enable
3. Send device programming codes, if external triggering is used, enable rear panel trigger (A)
4. Trigger the generator by one of the methods indicated below

TRIGGERING:

1. Send group execute trigger (GET), TRIGR subroutine
2. Send "R" if in trigger/reset mode
3. Use external trigger

59401A BUS ANALYZER

BUS FUNCTIONS: Talker, Listener, Controller

LISTEN: With memory on, stores up to 32 characters for later perusal.

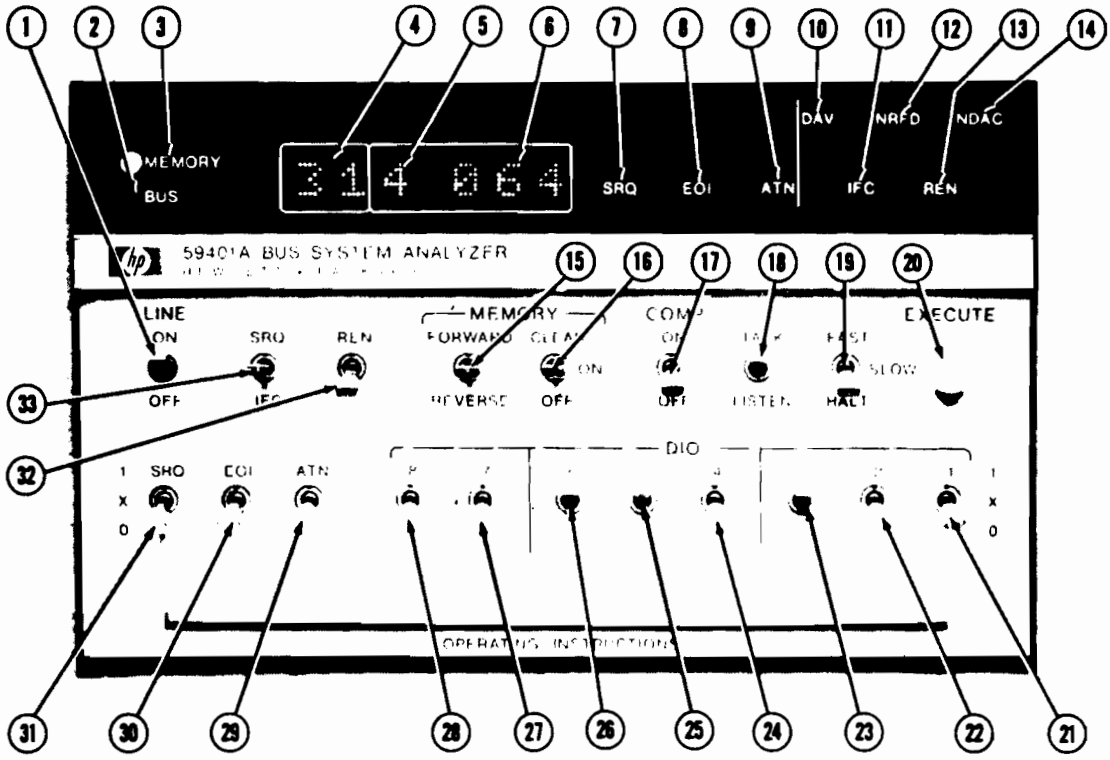
TALK: Up to 32 characters can be stored in memory as a program or individual characters can be output to the bus one at a time using DIO toggle switches.

HALT: Allows one character at a time to be read from the bus or output to the bus.

SLOW: Allows two characters/second to be output to or read from the bus.

FAST: Allows full HP-IB transfer speed.

COMP: With compare on, the analyzer will output data to the bus or read from the bus until a value equal to the setting of the DIO toggle switches is encountered. The data transfer will then be halted.



(1) **Line Control.** Switches 59401A power on and off.

(2) **Bus Indicator.** Indicates that the conditional display is disclosing real time Bus information.

(3) **Memory Indicator.** Indicates that the conditional display is disclosing memory contents.

(4) **Memory Location.** Indicates the memory location of the current instruction.

(5) **ASCII Character Indicator.** Displays the ASCII character equivalent to the current octal instruction. For example, "077_8" is displayed as "?".

(6) **Octal Indicator.** Displays the Octal equivalent of the current binary instruction. For example, a Bus or memory instruction of "00 111 111_2" is displayed as "077_8".

(7) **SRQ Indicator.** Lights when the SRQ state is true (subject to condition of Bus/Memory indicators).

(8) **EOI Indicator.** Lights when the EOI state is true (subject to condition of Bus/Memory indicators).

(9) **ATN Indicator.** Lights when the ATN state is true (subject to condition of Bus/Memory indicator).

(10) **DAV Indicator.** Indicates that the active controller has valid data on the Bus lines.

(11) **IFC Indicator.** Lights when the IFC Bus line is in the true state (see also (33)).

(12) **NRFD Indicator.** Indicates that one or more instruments is not ready for data.

(13) **REN Indicator.** Lights when the REN Bus line is in the true state (see also (32)).

(14) **NDAC Indicator.** Indicates that one or more instruments has not accepted the data.

(15) **Forward/Reverse Control.** Increments the memory in the appropriate direction when the 59401A is in LISTEN or TALK, HALT and MEMORY is ON.

(16) **CLEAR/ON/OFF.** All memory locations are set to 000_8 when cleared. When the memory is set to ON and when the 59401A is set to LISTEN or TALK HALT, memory locations can be incremented in the appropriate direction with the FORWARD/REVERSE control.

The memory is also incremented forward by the EXECUTE button when the memory is set to ON, but the information in the lower switch register is first memorized (in the TALK HALT mode). The memory is protected when it is turned off, but it is available to the Bus lines in the TALK, FAST, or SLOW modes.

(17) **COMP Control.** When COMP is set to ON, the data on the Bus is compared to the lower switch register and Bus traffic is stopped when the data matches the settings of the switches. If a switch is set to the "X" (don't care) position, it automatically matches the corresponding data bit on the Bus. When the 59401A is in the talk mode, the COMP, ON function will also stop Bus traffic at memory location 31.

(18) **Talk/Listen Control.** Places the 59401A in either a talk or listen mode.

(19) **FAST/SLOW/HALT.** In the fast mode, the 59401A can interchange information at the maximum system transfer speed. In the slow mode, the 59401A limits Bus speed to two bytes/second. When this switch is set to HALT, information interchange on the Bus is stopped. The halt function is overridden by the EXECUTE control.

(20) **Execute Control.** When EXECUTE is pressed:

a. In LISTEN, HALT, the 59401A accepts one character.

b. In TALK, HALT, MEMORY OFF, the 59401A sends one character.

c. In TALK, HALT, MEMORY ON, the 59401A loads one character into memory.

d. In LISTEN, FAST or SLOW, COMP ON, the 59401A permits Bus traffic to continue.

e. In TALK, FAST or SLOW the 59401A continues sending data.

f. In LISTEN, MEMORY ON (after use of the FORWARD/REVERSE switch has put the 59401A into memory mode), the 59401A leaves the memory mode.

(21) **DIO1 Control*.** This switch controls the 1_2 (1_8) data line.

(22) **DIO2 Control*.** This switch controls the 10_2 (2_8) data line.

(23) **DIO3 Control*.** This switch controls the 100_2 (4_8) data line.

(24) **DIO4 Control*.** This switch controls the 1000_2 (10_8) data line.

(25) **DIO5 Control*.** This switch controls the 10000_2 (20_8) data line.

(26) **DIO6 Control*.** This switch controls the 100000_2 (40_8) data line.

(27) **DIO7 Control*.** This switch controls the 1000000_2 (100_8) data line.

(28) **DIO8 Control*.** This switch controls the 10000000_2 (200_8) data line.

(29) **ATN Control*.** The 59401A can address an instrument or deliver universal commands when this switch is set to the "1" position.

(30) **EOI Control*.** EOI true (1) may indicate the end of a data string or, with ATN true, EOI puts the Bus in the parallel polling mode.

(31) **SRQ Control*.** SRQ calls for the attention of the controller. Typically, this line is used with either the COMP switch or the COMPARE OUTPUT to detect the presence of SRQ in a program.

(32) **REN Control.** Instruments that can be set for remote operation are enabled to do so when this switch is set to REN.

(33) **SRQ/IFC Control.** Setting this switch to IFC stops all communications on the Bus.
Setting this switch to SRQ calls for the attention of the controller.

*The asterisked switches are in the true state when set to the "1" position. These switches are in the false state when set to the "0" position. The "X" (don't care) position is used when comparing the lower switch register to the bus contents. In the "X" position, a comparison is true whether the Bus contains a 1 or a 0. For example, if the DIO switches are set to 00 110 $10X_2$, a comparison will be valid for either 00 110 100_2 or 00 110 101.

ASCII CHARACTER CODES

ASCII CHARACTER	OCTAL CODE	DECIMAL CODE	ASCII CHARACTER	OCTAL CODE	DECIMAL CODE
NUL	00	0	SP	40	32
SOH	01	1	!	41	33
STX	02	2	"	42	34
ETX	03	3	#	43	35
ETO	04	4	$	44	36
ENQ	05	5	%	45	37
ACK	06	6	&	46	38
BEL	07	7	'	47	39
BS	10	8	(	50	40
HT	11	9	)	51	41
LF	12	10	*	52	42
VT	13	11	+	53	43
FF	14	12	,	54	44
CR	15	13	—	55	45
SO	16	14	.	56	46
SI	17	15	/	57	47
DLE	20	16	0	60	48
DC1	21	17	1	61	49
DC2	22	18	2	62	50
DC3	23	19	3	63	51
DC4	24	20	4	64	52
NAK	25	21	5	65	53
SYN	26	22	6	66	54
ETB	27	23	7	67	55
CAN	30	24	8	70	56
EM	31	25	9	71	57
SUB	32	26	:	72	58
ESC	33	27	;	73	59
FS	34	28	<	74	60
GS	35	29	=	75	61
RS	36	30	>	76	62
US	37	31	?	77	63

ASCII CHARACTER CODES (continued)

ASCII CHARACTER	OCTAL CODE	DECIMAL CODE	ASCII CHARACTER	OCTAL CODE	DECIMAL CODE
@	100	64	(Apost.)	140	96
A	101	65	a	141	97
B	102	66	b	142	98
C	103	67	c	143	99
D	104	68	d	144	100
E	105	69	e	145	101
F	106	70	f	146	102
G	107	71	g	147	103
H	110	72	h	150	104
I	111	73	i	151	105
J	112	74	j	152	106
K	113	75	k	153	107
L	114	76	l	154	108
M	115	77	m	155	109
N	116	78	n	156	110
O	117	79	o	157	111
P	120	80	p	160	112
Q	121	81	q	161	113
R	122	82	r	162	114
S	123	83	s	163	115
T	124	84	t	164	116
U	125	85	u	165	117
V	126	86	v	166	118
W	127	87	w	167	119
X	130	88	x	170	120
Y	131	89	y	171	121
Z	132	90	z	172	122
[	133	91	{	173	123
\	134	92	:	174	124
]	135	93	}	175	125
^	136	94	~	176	126
—	137	95	DEL	177	127

APPENDIX C

LAB ASSIGNMENTS

GENERAL INSTRUCTIONS FOR THE LAB ASSIGNMENTS:

(1) INITIALIZE AND REMOTE THE DEVICES ON THE BUS AT THE BEGINNING OF EACH PROGRAM.

(2) UNBUFFER THE BUS AND SET THE E BIT OF THE CONFIGURATION WORD FOR EACH DEVICE FOR WHICH I/O ERRORS WILL BE PROCESSED BY THE USER.

(3) CHECK ERROR CODE AFTER EACH I/O OPERATION IF THE E-BIT IS SET.

(4) IF YOU TAKE A READING FROM AN ACTIVE DEVICE (DVM, COUNTER, ETC.), MAKE SURE THAT YOU PROGRAM THE DEVICE TO BE IN HOLD/MANUAL TRIGGER POSITION, AND THAT YOU ARE GOING TO TRIGGER THE DEVICE BEFORE EACH READING IF PROGRAM IS TO HANDLE ERRORS.

(4) AFTER FINISHING THE PROGRAM, RETURN THE DEVICES TO THE LOCAL POSITION.

LEARNING MODULE 1

REFERENCES: Lessons 1-3 of the HP-IB student workbook
HP-IB User's Manual: Pgs. B-1 to B-4, C-2 to C-4, 3-1 to 3-10.

OBJECTIVES: To allow the student to configure an HP-IB instrument cluster into an HP 1000 computer system and to develop an understanding of addressing conventions and device roles on the bus.

1) How does one know what address code to assign to each device? Can just any code be assigned?

2) What is the difference between a bus controller and a system controller?

3) What are three ways to "untalk" a current talker?

4) What is the limiting factor on data transfer rates between bus devices?

   #2 of Learning Module 3 is a good example of this. Note the display and printer during this exercise.

5) How are "talk" and "listen" addresses formed from the 5 bit octal device address?

6) To prepare the HP-IB instrument cluster for programming, each device must have a 5 bit octal address assigned and the cluster must be connected to the system. The following steps should assure smooth operation:

   1. Set the address switches on each unit to the desired address code.
   2. Connect all devices in a serial configuration observing the 20m cabling restriction. The bus analyzer should be the last device on the bus.

3. On the bus interface card, set the parallel poll ID (PPID) switches so that the card will respond on DIO line 1. This is required for the bus diagnostic to execute the parallel poll test properly. Set the IFC and REN switches to the "on" position. Set the shield switch to CNX. For RTE-IV systems, set the address switches to 0.

4. Install the interface card in the computer at the select code indicated in the system generation map.

5. Connect the bus interface cable to the 59310B bus interface card.

6. Connect the other end of the interface cable to the first device on the bus.

7) What is the difference between secondary addressing and multiple addressing for a device?

   Can the same device support both types of addressing?

LEARNING MODULE II

REFERENCES: Lessons 4-6 of the HP-IB Student Workbook
HP-IB User's Manual: Pgs. 2-3 to 2-7, 5-10 to 5-13
Device Programming Reference Sheets
DVR37 HP-IB Driver Reference Manual

OBJECTIVES: Allow the student to become familiar with programming the HP-IB in an RTE-IV system using auto-addressing.

1) Connect the thermal printer, digital clock, and numeric display to the bus and assign LU's and address codes. Be sure to include the LU's in your SST.

   Write a program using auto-addressing that will:

   a. Take a reading from the digital clock

   b. Print that reading on the thermal printer and/or display it on the numeric display

   c. Clear the interface and exit

2) Run the program of #1 again. This time set the bus analyzer functions as described below and observe the bus traffic.

```
MEMORY: OFF
  COMP: OFF
FUNCTION: LISTEN
 SPEED: HALT
```

   Each time the EXECUTE button is depressed, a character is transmitted over the bus and displayed on the analyzer. Step through the program.

   What bus traffic corresponded to the program statements shown:

```
Write/Print ?
       Read ?
```

3) FMGR commands can be used to do limited programming of the bus. Refer to the AN401-1 Application Note and use FMGR commands to display the time from the digital clock on the numeric display.

   Note that the FMGR commands :ST and :DU do not complete until an EOF is sensed. Since the clock doesn't send EOF, FMGR never returns. To return to FMGR, get into breakmode and issue the RS command to restart the program.

   After issuing the :DU or :ST command, run WHZAT from the breakmode and observe the status of FMGR.

4) Unless the user specifies that I/O error handling for a bus device will be performed in his/her program, the system normally processes I/O errors. Refer to the AN401 Series Application Notes and/or the RTE-IVB Programmer's Reference Manual and do the following:

   a. Set the bus time out to 10 seconds.

   b. Write a Fortran program that will take a single reading from one of the bus devices that is NOT a talker such as the thermal printer or numeric display.

   The system I/O error message format is shown below:

```
IONR L www Exx Syy zzz
```

   where: www is the device LU
   xx is the device EQT
   yy is the device subchannel
   zzz is the device status returned by the driver

   The HP-IB Driver Reference Manual contains error codes. Check the error code returned by the system when you ran your program. What does it show?

   The bus can be restored simply by getting to breakmode and using the UP command.

   c. Up the bus but let the program continue.

   Note that the bus will continue to go down with the same condition every 10 seconds until the error situation is corrected.

d. Off the program and up the bus EQT if necessary.

e. Set the bus time out back to its original setting.

NOTE: The purpose of labs 5 and 6 is to let the student use and manipulate data retrieved from the bus devices. Don't use the system clock or loops or WAIT statements to implement the required time delays. Rather, use only the data obtained from the timing generator and clock.

5) Using auto-addressing, write a program that will:

a. Set up the timing generator to:

i. Pacer mode
ii. SRQ disable
iii. Trigger/Reset
iv. 1 sec. interval.

"PDR

b. Trigger the generator and let it count 10 seconds. When the generator reaches 10 secs. output the current clock time to the thermal printer and the number of timing intervals from the generator to the display.

c. Clear the interface and terminate.

6) Using auto-addressing write a program that will:

a. Set the digital clock to a starting time specified by run string parameters or by prompting the user to input the time from the terminal.

b. Start the clock and let 10 seconds elapse. When ten seconds have elapsed, output the time to the display and a message and/or the time to the thermal printer.

LEARNING MODULE III

REFERENCES: Lessons 4-6 of HP-IB Workbook
HP-IB User's Manual: Pgs. 3-2 to 3-6.

OBJECTIVES: Allow student to become familiar with programming the bus in an RTE system using direct addressing.

1) What is the major difference between auto and direct addressing?

2) Write a program using direct addressing that will:

   a. Display on the numeric display and print on the thermal printer the current time from the digital clock.

   b. Clear the interface and exit.

3) Run the program of #2 again. This time set the bus analyzer functions as indicated below and observe bus traffic.

```
MEMORY: OFF
  COMP: OFF
TALK/LISTEN: LISTEN
EXECUTE: HALT
```

   Single step through the program allowing one character on the bus at a time using the execute button.

   What bus traffic corresponded to the CMDW subroutine?

4) Auto-addressing is very easy to use. Direct addressing, however, has one major advantage over auto-addressing. Compare the programs you wrote for LM II-#1 and LM III-#2 and determine what that advantage is.

   Why are we able to make use of this capability using direct addressing but not auto-addressing?

LEARNING MODULE IV

REFERENCES: Lessons 4-6 of HP-IB Workbook
HP-IB User's Manual: Pgs. 4-1 to 4-4
Device Programming Reference Sheets

OBJECTIVES: Allow student to become familiar programming the bus in an RTE system using the bus message routines.

Allow the student to observe the actual bus traffic created by the bus message routines.

Introduce use of device configuration words and show the speed difference between DMA and interrupt methods.

1) Run program LAB41 and observe bus traffic on the analyzer. Also observe the DVM and/or timing generator front panel indicators. Set the bus analyzer functions as indicated below. For proper execution, the DVM address code must be set to 3B and the timing generator address code set to 6B.

```
     MEMORY: OFF
       COMP: OFF
TALK/LISTEN: LISTEN
    EXECUTE: HALT OR SLOW
```

What bus traffic did you observe for each message routine the program called? What happened when you tried to change the DVM front panel settings after LLO?

2) Write a program that will:

   a. Initialize and remote the devices on the bus

   b. Set up the DVM to:

      i. 2 wire resistance measurement
      ii. Auto-range
      iii. High resolution
      iv. Math off
      v. Hold/manual trigger

c. Program the scanner to scan 10 channels, each time triggering the DVM to make resistance reading.

d. Print the DVM results on the numeric display and thermal printer.

e. Clear and local devices and terminate.

3) Write a program that will show the difference in speed between DMA and interrupt methods of data transfer.

a. Use RMPAR and provide the # of readings to be taken and the length of each reading to be taken in the run string.

b. Set up the timing generator to 1 msec. pacer mode, SRQ disable, trigger/reset.

c. To provide a constant source of data, reconfigure the clock to "no EOR required" and set its mode switches to "talk only".

d. Trigger the timing generator and take the specified number of readings from the clock. Use an EXEC call for the readings because it's faster and the length of the reading can be specified by a run string variable.

e. Read the number of timing generator periods required to complete the specified number of clock readings and output it to a suitable list device.

f. Reconfigure the clock for DMA and "no EOR required".

g. Trigger the timing generator and again take the specified number of clock readings.

h. Read the number of timing generator periods required and output to a list device.

i. Clear, local and unconfigure the appropriate bus devices and terminate.

Does DMA always beat the interrupt method? If not, when is the interrupt method faster? How do you account for this?

LEARNING MODULE V

REFERENCES: Lessons 5, 6, 7 of the HP-IB Workbook
HP-IB User's Manual: Pgs. 5-1 to 5-5
DVR37 Operating and Programming Manual: Pgs. 2-13 to 2-17

OBJECTIVES: Allow students to handle service requests from bus devices as well as becoming familiar with the situations in which they may occur and in which they are useful.

Familiarize the student with the necessary prerequisites needed for successful SRQ handling.

Introduce the student to user error handling in place of system error handling.

1) Write a program using EXEC calls that will:

   a. Set up the timing generator to:

      i. Pacer mode, 1 sec. period
      ii. SRQ disable
      iii. Trigger/reset

   b. Let the generator count 10 timing periods

   c. After 10 timing periods, output the current clock time to the display and/or the thermal printer

   d. IFC and terminate

2) Name 4 things that must happen before an SRQ may be successfully processed.

3) Write a program that will:

   a. Initialize and remote bus devices

b. Set up the DVM to:

   i. 2 wire resistance measurement
   ii. Auto-range
   iii. High resolution
   iv. Math off
   v. Hold/manual trigger
   vi. Data ready RQS on

c. Set the scanner to channel 0

d. Schedule a service program each time the DVM completes a reading. The DVM will assert the SRQ line.

e. Trigger the DVM

f. Terminate

Write a service program that will:

a. Print the DVM measurement on the thermal printer with an appropriate heading.

b. Increment scanner to the next channel

c. Trigger the DVM

d. Terminate saving resources, EXEC(6,,1)

e. Clear and local devices and terminate after the scanner has gone through all 10 channels.

Run the program of #3 but first use the FMGR :OF command to purge the service program. What happens?

To restore the bus, simply off the program from breakmode and up the bus EQT from FMGR. Refer to the indicated pages of the DVR37 Manual to discover the meaning of the error code returned by the DVR37 driver.

Reload the service program. In your main program replace the service program name with blanks. Recompile and load your main program. What do you expect to happen when you run the program? Run it. What happened? To restore the bus, "OF,progname" from breakmode then ":UP,bus EQT" from FMGR.

Restore the service program name in your main program, recompile and load, and see if the program runs correctly.

4) Re-run the program of #3 or #7 but use the bus configuration message to set the E-bit in the DVM or timing generator configuration word. Use IBERR to check for I/O errors after each I/O operation and write an error handling routine that will output an error message, clean up the bus and devices and terminate the program.

5) Repeat the service program manipulations dicussed in #3. Does your error handling routine adequately handle the I/O errors obtained? Note that the bus no longer requires restoration, i.e. it is not set down automatically on an I/O error.

6) What would happen if the DVM were set up to assert SRQ after each measurement but the SRQ call was never made in your program?

7) Write a program that will:

    a. Initialize and remote bus devices

    b. Set up the timing generator to:

        i. Timer mode, 10 second timing period
        ii. Trigger/reset
        iii. SRQ enable

    c. Schedule a service program on SRQ. The timing generator will assert the SRQ line at the end of each timing period.

    d. Trigger the generator

    e. Terminate

    Write a service program that will:

    a. Print the time from the clock and a message on the thermal printer.

    b. Trigger the generator

    c. Terminate saving resources, EXEC(6,,1)

    d. Clear and local devices and terminate after 1 minute.

Perform the same service program manipulations as described in #3 and answer the questions.

LEARNING MODULE VI

REFERENCES: Lessons 6 and 7 of the HP-IB Workbook
HP-IB User's Manual: Pgs. 5-1 to 5-5, 5-10 to 5-12

OBJECTIVES: Allow the student to gain familiarity with interrupt and error handling when programming the HP-IB in an RTE System.

1) What does the statement CALL SRQ(device lu,17) actually do to keep a service program from being scheduled?

   Why is it a good idea to use this call before terminating a service program?

2) What must occur before an I/O error can be handled by a user's program? What is the default procedure for error handling? How could we change the default error handling to the user?

3) Write a program that will take resistance measurements of 10 resistors every 5 seconds. The timing generator will assert the SRQ line every 5 seconds which will schedule a service program to make the measurement. The program should terminate after 25 seconds.

   The program should perform the following:

   a. Initialize and remote bus devices

   b. Set up the DVM to:
      i. Resistance Measurement
      ii. Auto-range
      iii. High resolution
      i. Math off
      v. Hold/manual trigger

   c. Set the E-bit of each device's configuration word

   d. Set the timing generator to:

      i. Pacer mode, 5 second timing period
      ii. Enable service request
      iii. Trigger/reset

e. Check for I/O error after each I/O operation. If an error occurs, print the error message on the terminal and terminate the program. If no error exists, continue.

The service program should:

a. Take a reading from the digital clock

b. Take a measurement of 10 resistors and print their values on the thermal printer. Also print the clock time as a header.

c. Check for I/O errors after each I/O operation and process as in the main program.

d. Trigger the timing generator.

e. Terminate saving resources
EXEC(6,,1)

f. Disable itself from being scheduled, clear devices, and terminate after 25 seconds.

4) You have just been hired by a local terrorist organization as an instrumentation programmer. Your first mission (Jim) is to use:

a. HP 59308A Timing Generator

b. HP 5150A Thermal Printer

c. HP 59304A Numeric Display

to develop an automatic ransom note writer, variable delay electronic time bomb fuse, and quasi-explosive effect system. You are to write a program that will:

a. Clear and initialize bus devices

b. Set up timing generator to:

i. Pacer mode, 10 second interval
ii. Trigger/reset
iii. SRQ enable

c. Take clock readings and loop until clock time reaches 12:00:00.

d. At 12:00:00, output a ransome message to the "remote" thermal printer.

e. Allow 2 minutes (how generous) for compliance with demands (FREE THE CUPERTINO BINARY!!!!). Trigger the timing generator to start the countdown in 10 second intervals.

f. Set up service program to be scheduled on SRQ from timing generator.

g. Terminate

NOTE: Set the clock to about 11:59:30 before running the program.

The service program should:

a. Output the time remaining to the "remote" thermal printer and to the local display.

b. If the 2 minutes is not up, terminate saving resources EXEC(6,,1) If the 2 minutes is up, prompt the terrorist leader for the length of the fuse (use the timing generator).

c. Activate the fuse (trigger the generator)

d. When the fuse time is up (generator SRQ), set off the bomb and print a pseudo "BOOM" on the thermal printer.

e. Clear and local devices, disable the service program from being scheduled, and terminate.

f. Go into hiding.

LEARNING MODULE VII

REFERENCES: Lesson 8 of the HP-IB Workbook
Appendix A of the HP-IB Workbook
Basic 1000 Manual
HP-IB User's Manual: Pgs. C-5 to C-6

OBJECTIVES: Allow the student to program bus devices using device subroutines.

1) Repeat #2 of Learning Module 4 but:

   a. Write a device subroutine to set up the DVM

   b. Write a device subroutine that will:

      i. Set the scanner to any channel (0-9)
      ii. Trigger the DVM
      iii. Read from the DVM and output the value to the thermal printer

   c. If you wrote #2 of Learning Module 4 in Fortran, rewrite it in Basic. Build a Branch and Mnemonic Table for the subroutines you wrote.

2) When are device subroutines an advantage over the usual method of device programming? Do device subroutines save time and memory?

LEARNING MODULE VIII

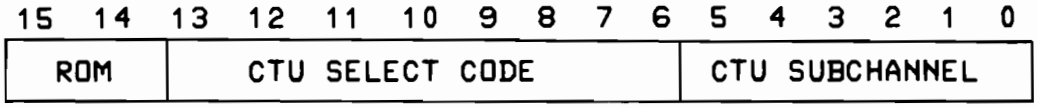
REFERENCES: Lesson 9 of HP-IB Student Workbook
AN401 Series Application Notes, especially AN401-1.
HP-IB User's Manual: Pgs. A-4 to A-7
HP 59310A/B Interface Bus Interface Diagnostic Reference Manual

OBJECTIVES: To familiarize the student with the use of the HP-IB diagnostic in troublshooting mode to program the bus.

To allow the student to observe actual bus traffic via the bus analyzer.

1) Follow these steps to load the HP-IB diagnostic:

   1. Power down the system and set the HP-IB interface card address to 0. Set the PPID code to 1.
   2. Power up the system and leave halted. Be sure to allow the disc to come up to speed.
   3. The diagnostic software is on 2 files on mini-cassette. The first file contains the diagnostic configurator, the second the diagnostic program.
   4. Set the S-register from the CPU front panel as indicated below:



The "ROM" value is the location of the loader ROM used to load the configurator and diagnostic off the minicassette.

The CTU select code can be found by determining the system LU of the terminal being used and finding that LU entry in the Device Reference Table of the system generation map. Another way is to list the Session Switch Table, find the EQT of the terminal and then use the FMGR command, :SYEQT,term. EQT. The first two digits of the reply indicate the select code of the terminal and thus the CTU.

Usually the left CTU is subchannel 1, the right CTU is subchannel 2.

5. Press STORE,PRESET,IBL,PRESET,RUN.

   The configurator will be loaded off the tape. Wait for HALT 77 (bits 0-5 lit and run light off).

6. Set the P-register to 100B and STORE.

   Set the S-register as indicated below:

15	14	13	12	11 10 9 8 7 6	5 4 3 2 1 0
0	1	0	0	CTU SELECT CODE	TERMINAL SC

The terminal and CTU select codes are the same.

Press STORE,PRESET,RUN.

The configurator will load the diagnostic from file 2 of the tape.

Wait for HALT 77.

7. Set the P-register to 100B and STORE

   Set the S-register as follows:

   bits 0-5 Select code of 1st HP-IB interface card under test

   bits 6-11 Select code of 2nd card under test or 0 if not applicable

   bits 12-15 Not applicable

   Press STORE,PRESET,RUN.

   Wait for HALT 70.

8. Set the S-register as follows:

   bits 0-5 Bus address of first interface card under test

   bits 6-11 Bus address of 2nd card under test if applicable

   bits 12-15 Not applicable

   The recommended address assignment for the bus interface card is 0B for RTE-IV systems and 36B for RTE-L systems.

Press STORE,PRESET,RUN.

Wait for HALT 71.

9. Set the S-register as follows:

   bits 0-7 PPID code of 1st interface card under test

   bits 8-15 PPID code of 2nd card under test or 0 if not applicable

   The PPID code should be set to 1 if the diagnostic will perform the parallel poll response test.

   Press STORE,PRESET,RUN.

   Wait for HALT 74.

10. Set the S-register with the desired diagnostic options as indicated below. No instruments should be connected to the card during the MAIN and PARALLEL POLL tests.
    (1=TRUE=YES)

bit 15	Reserved
bit 14	Suppress error halts
bit 13	Reserved
bit 12	Loop through diagnostic
bit 11	Suppress error messages
bit 10	Suppress non-error messages
bit 9	Reserved
bit 8	Suppress preset test
bit 7-4	Reserved
bit 3	Do parallel poll request test
bit 2	Use 2nd IBI select code in troubleshooting module
bit 1	Use 1st IBI select code in troubleshooting module
bit 0	Execute cable test (requires 2 IBI's)

A good value for class use is S = 050402B.

Press STORE,PRESET,RUN.

The following message appears at the terminal:

59310B Diagnostic
HP-IB Troubleshooting Module
:
:

The HP-IB devices may now be connected to the bus and programmed using the diagnostic.

2) Refer to pgs. 9-16 to 9-23 of the Workbook and the Device Programming Reference Sheets Appendix and use the diagnostic to perform the following programming sequence.

   a. Clear the interface

   b. Set the controller (interface) to talk

   c. Set the clock and printer to listen

   d. Stop and reset the clock. Restart the clock.

   e. Set the clock to talk.

   f. Untalk the clock.

   g. Set the clock to talk.

   h. Clear the interface.

   Observe bus traffic on bus analyzer.

3) Read HP-IB User's Manual pgs. 4-2 to 4-6. How does the HP-IB driver know in what format to send data to a device? To receive data from a device?

4) Convince yourself of the validity of the preliminary system controller interface commands on pg. 9-19 using the diagnostic to communicate to the bus.

   a. Clear the interface

   b. Put bus in command mode

   c. Send timing generator listen address

   d. Put bus in data mode and observe timing generator front panel.

What happened?

e. Clear the interface.

What happened? What does IFC do?

f. Remote enable bus and observe bus analyzer front panel.

g. Put bus in command mode.

h. Send timing generator listen address and observe front panel.

i. Send DVM listen address and observe DVM front panel.

Did the expected happen? Of course it did!!

j. Observe timing generator, DVM, and bus analyzer front panels while sending GTL (go to local).

k. Repeat f., g., h., i.

l. Clear the interface.

What is the difference between GTL or "not" REN and IFC?

APPENDIX D

LAB SOLUTIONS

LEARNING MODULE I SOLUTIONS

1) How does one know what address code to assign to each device?

   Can just any code be assigned?

   Basically, the choice of address code for each bus device lies with the user. When the user assigns an LU to a bus device, whether at generation time or on-line, that LU represents a subchannel of the bus EQT. This subchannel is the address code which must be set on the device via the address switches. Thus, a device may be accessed by any bus LU as long as setting of the device's address switches corresponds to the bus subchannel indicated by that LU.

   BUT —

   Not just any address code can be assigned to a bus device. Only addresses in the range 0-31 decimal (0-37B) are valid. Furthermore, in RTE-IV systems, 0 is reserved for the interface card and 37B is used for "untalk", "unlisten". In RTE-L systems, 36B is reserved for the interface card address.

2) What is the difference between a bus controller and a system controller?

   The exclusive functions IFC and REN can only be asserted by a system controller. Otherwise, there is no difference between the two. A system controller can be a bus controller but not necessarily vice-versa.

3) What are three ways to "untalk" a current talker?

   1. Send "UNTALK" on the bus
   2. Address another talker. This always unaddresses the current talker.
   3. Send IFC on the bus. This untalks, unlistens, and unaddresses all bus devices.

4) What is the limiting factor on data transfer rates between bus devices?

Since the HP-IB is an asynchronous device, the data transfer rate between devices is limited by the speed of the slowest device involved in the transfer. #2 of Learning Module 3 shows a good example of this. Note the difference in speed of the numeric display and the thermal printer.

5) How are "talk" and "listen" addresses formed from the 5 bit octal device address?

The HP-IB driver, DVR37, forms talk and listen addresses from the same 5 bit device address code by adding two function bits to form a 7 bit address as shown:

	b7	b6	b5	b4	b3	b2	b1
TALK	1	0	x	x	x	x	x
LISTEN	0	1	x	x	x	x	x

Therefore —

$$\text{Talk address} = \text{device addr. code} + 100B$$

$$\text{Listen address} = \text{device addr. code} + 40B$$

7) What is the difference between secondary addressing and multiple addressing for a device?

Can the same device support both types of addressing?

A secondary address refers to a buffer, I/O port, etc. internal to a bus device. It is used in conjuction with a device primary address to write/read data to/from an internal device location, i.e. a location other than that accessed by the device's primary address. Secondary addressing routines will be discussed later.

A multi-addressable device is one that can be addressed by multiple LU's without changing its address code. A multi-address device has only 4 address code switches as compared to the 5 normally seen on other bus devices. The fifth and least significant address bit of the multi-address device is a "don't care". Thus, the device represents two bus EQT subchannels and can be referenced via the two corresponding LU's.

The same bus device can support multiple and secondary addressing. Note, however, that only 31 secondary locations can be accessed per device regardless of the number of LU's which reference that device.

LEARNING MODULE II SOLUTIONS

```
0001   FTN4,L
0002         PROGRAM LAB2
0003         DIMENSION ITIME(5)
0004   C  DIGITAL CLOCK LU:
0005         LUCK=48
0006   C  NUM. DISPLAY LU:
0007         LUDP=25
0008   C  THER.  PRINTER LU:
0009         LUTP=50
0010         READ(LUCK,10)ITIME
0011         WRITE(LUDP,20)ITIME
0012         WRITE(LUTP,30) (ITIME(J),J=1,5)
0013    10   FORMAT(2X,5A2)
0014    20   FORMAT(5A2)
0015    30   FORMAT(A2,4(":",A2),/////)
0016         END
```

```
0001    10REM    *LEARNING MODULE 2 - #1*
0002    15REM
0003    20DIM A$[20]
0004    25REM
0005    30REM    *CLOCK, DISPLAY, PRINTER LU'S*
0006    35REM
0007    40LET C=48
0008    50LET D=25
0009    60LET P=50
0010    65REM
0011    70REM    *TAKE CLOCK READING*
0012    75REM
0013    80READ #C;A$
0014    85REM
0015    90REM    *OUTPUT CURRENT TIME TO PRINTER, DISPLAY*
0016    95REM
0017    100PRINT #D;A$
0018    110PRINT #P;A$
0019    120FOR I=1TO 4
0020    130PRINT #P
0021    140NEXT I
0022    150END
```

2)

CLOCK LU	48	SUBCHANNEL	5
PRINTER LU	50	SUBCHANNEL	7
DISPLAY LU	25	SUBCHANNEL	4

	ASCII	DESCRIPTION	OCTAL
Read from clock:	−?	Untalk,Unlisten	137 77
	E	Clock Talk	105
		10 Chars. Data	
	CRLF	Carriage Return	15
		Line Feed	12
Write to display:	−?	Untalk,Unlisten	137 77
	$	Display Listen	44
		10 Chars. Data	
	CRLF	Carriage Return	15
		Line Feed	12
Write to printer:	−?	Untalk,Unlisten	137 77
	'	Printer Listen	47
		10 Chars. Data	
	CRLF	Carriage Return	15
		Line Feed	12

3) This is a very easy solution requiring only one command. It is easier to do interactively than by a transfer file but a transfer file could be used if more extensive programming is to be done.

:ST,48,25
or
:DU,48,25

Where: Clock LU = 48
Display LU = 25

WHZAT shows that FMGR is in the I/O suspend state, waiting on the bus while the :DU or :ST command is active.

4) a. Set the bus time out to 10 seconds.

:SYTO,bus EQT,1000

b. Write a Fortran program that will take a single reading from one of the bus devices that is NOT a talker such as the thermal printer or numeric display.

```
FTN4,Q,L
        PROGRAM TEST
        DIMENSION INBUF(20)
        LUTP=50
        READ(LUTP,4) INBUF
        END
```

The HP-IB Driver Reference Manual contains error codes. Check the error code returned by the system when you ran your program. What does it show?

```
IONR L   50 E10 S 7     1

      PRINTER LU      = 50
     PRINTER EQT      = 10
PRINTER SUBCHANNEL    = 7
   PRINTER STATUS     = 1 = Device timed out
```

c. Up the bus but let the program continue.

S= xx COMMAND ?UP,10

d. Off the program and up the bus EQT if necessary.

S= xx COMMAND?OF,TEST

:SYUP,10

e. Set the bus time out back to its original setting.

:SYTO,10,xxxxx

```
0001  FTN4,Q,L
0002        PROGRAM LAB25
0003        DIMENSION IDATA(3),ICLOK(5)
0004  C
0005  C THIS PROGRAM SETS THE TIMING GENERATOR TO PACER MODE, 1 SEC. INTERVAL
0006  C AND LETS 10 INTERVALS PASS, THEN OUTPUTS THE ELAPSED TIME ON THE DISPLAY
0007  C AND THE CURRENT CLOCK TIME ON THE THERMAL PRINTER.
0008  C
0009        IBLU = 9
0010        IGLU = 49
0011        ICLU = 48
0012        IDLU = 25
0013        IPLU = 50
0014  C
0015  C SET UP GENERATOR TO PACER MODE,TRIGGR/RESET,SRQ DISABLE,1 SEC. INTERVAL
0016  C
0017        WRITE(IGLU,10)                ''PRD
0018        CALL RMOTE(IGLU)
0019  10    FORMAT("PRD001E6R")
0020  C
0021  C READ NUMBER OF CYCLES SINCE TRIGGER FROM GENERATOR
0022  C
0023  20    READ(IGLU,30) IDATA
0024  30    FORMAT(2X,3A2)
0025  C
0026  C CHECK TO SEE IF 10 SECONDS HAVE PASSED
0027  C
0028        WRITE(1,30) IDATA
0029        IF(IDATA(3) .LT. 2H10) GOTO 20
0030  C
0031  C RESET GENERATOR
0032  C
0033        WRITE(IGLU,40)
0034  40    FORMAT("_R")
0035  C
0036  C OUTPUT NUMBER OF GENERATOR CYCLES TO THE DISPLAY
0037  C
0038        WRITE(IDLU,50) (IDATA(I),I=1,3)
0039  50    FORMAT(3A2)
0040  C
0041  C GET CURRENT CLOCK READING
0042  C
0043        READ(ICLU,60) ICLOK
0044  60    FORMAT(2X,5A2)
0045  C
0046  C OUTPUT CURRENT CLOCK READING TO PRINTER
0047  C
0048        WRITE(IPLU,70) ICLOK
0049  70    FORMAT(A2,4(":",A2),/////)
0050        END
```

```
0001    5REM  *** LEARNING MODULE 2 - #5 ***
0002    10REM
0003    15DIM A$[10],B$[12]
0004    20REM  *** SET TIMING GEN. TO PACER MODE, TRIGGER RESET, SRQ DISABLE, ***
0005    30REM  *** 1 SEC. INTERVAL AND TRIGGER                                 ***
0006    40REM
0007    50LET A$="PRD001E6R"
0008    60REM
0009    70REM  *** SET UP CLOCK, DISPLAY, PRINTER, BUS, GENERATOR LU'S ***
0010    80REM
0011    90LET C=48
0012    100LET D=25
0013    110LET P=50
0014    120LET B=9
0015    130LET G=49
0016    140REM
0017    150REM  *** REMOTE ENABLE GENERATOR AND PROGRAM IT ***
0018    160REM
0019    170CALLRMOTE(G)
0020    180PRINT #G;A$
0021    190REM
0022    200REM  *** TAKE GENERATOR READING AND PRINT TO TERMINAL ***
0023    210REM
0024    220READ #G;B$
0025    230PRINT B$
0026    240REM
0027    250REM  *** TEN SECONDS ELAPSED? NO - TAKE NEXT READING ***
0028    260REM
0029    270IF B$[5,6]<"10"THEN 220
0030    280REM
0031    290REM  *** YES - DISPLAY NUMBER OF TIMING PERIODS, PRINT CURRENT TIME ***
0032    300REM
0033    310PRINT #D;B$
0034    320READ #C;B$
0035    330PRINT #P;B$
0036    340FOR I=1TO 4
0037    350PRINT #P
0038    360NEXT I
0039    370REM
0040    380REM  *** CLEAR BUS AND EXIT ***
0041    390REM
0042    400CALLCLEAR(B,2)
0043    410END
```

```
0001   FTN4,Q,L
0002         PROGRAM LAB26
0003   C
0004   C THIS PROGRAM SETS THE CLOCK TO THE DESIRED STARTING TIME, LET'S IT
0005   C TICK OFF 10 SECONDS, THEN OUTPUTS A MESSAGE AND THE TIME TO THE PRINTER
0006   C
0007   C RUN STRING IS :RU,LAB26,HR,MIN,SEC
0008   C
0009         DIMENSION IBUF(5),IDATA(5)
0010         ICLU = 48
0011         IPLU = 50
0012         IDLU = 25
0013         IBLU = 9
0014   C
0015   C RETRIEVE RUN STRING PARAMETERS TO USE FOR CLOCK SETTING
0016   C
0017         CALL RMPAR(IBUF)
0018         IHR = IBUF(1)
0019         IMIN = IBUF(2)
0020         ISEC = IBUF(3)
0021   C
0022   C RESET AND STOP CLOCK
0023   C
0024         WRITE(ICLU,5)
0025   5     FORMAT("RP")
0026   C
0027   C SET CLOCK TO STARTING TIME SPECIFIED BY RUNSTRING PARAMETERS
0028   C
0029         DO 10 I=1,IHR
0030         WRITE(ICLU,40)
0031   10    CONTINUE
0032         DO 20 I=1,IMIN
0033         WRITE(ICLU,50)
0034   20    CONTINUE
0035         DO 30 I=1,ISEC
0036         WRITE(ICLU,60)
0037   30    CONTINUE
0038   40    FORMAT("H")
0039   50    FORMAT("M")
0040   60    FORMAT("S")
0041   C
0042   C START CLOCK
0043   C
0044         WRITE(ICLU,70)
0045   70    FORMAT("T")
0046   C
0047   C DETERMINE IF 10 SECONDS HAVE EXPIRED
0048   C
0049   75    READ(ICLU,80) IDATA
0050         IDIF=IDATA(5)-IBUF(3)
0051         IF(IDIF .LT. 2H10) GOTO 75
0052   80    FORMAT(2X,5A2)
0053   C
0054   C OUTPUT TIME TO DISPLAY
0055   C
0056         WRITE(IDLU,90) IDATA
0057   90    FORMAT(5A2)
0058   C
0059   C OUTPUT MESSAGE TO PRINTER
0060   C
0061         WRITE(IPLU,100)
0062   100   FORMAT("IT'S ABOUT TIME!!!",//////)
0063         END
```

```
10   REM  *** LEARNING MODULE II  - #6 ***
30   REM
40   REM  *** DIMENSION INPUT BUFFER  ASSIGN I/O UNIT & OPEN ***
70   REM  *** DISPLAY I/O ***
80    E$
90   DIM E$[2]
95   LET C=40
97   LET E=0
98   LET F=5
99   LET D=25
85   REM
90   REM  *** PROMPT USER FOR CLOCK STARTING TIME AND SPECIFY IT ***
95   REM
100  PRINT "SET CLOCK TIME  HOUR,MINUTE,SECOND "
110  PRINT
120  INPUT H,M,S
125  REM
130  REM  *** REMOTE ENABLE, RESET, STOP CLOCK ***
135  REM
140  CALLRMOTE(C)
145  PRINT #C;"RP"
150  REM
155  REM  *** SET CLOCK TO STARTING TIME ***
160  REM
170  FOR I=1TO H
180  PRINT #C "H"
190  NEXT I
200  FOR I=1TO M
210  PRINT #C;"M"
220  NEXT I
230  FOR I=1TO S
240  PRINT #C;"S"
250  NEXT I
255  REM
260  REM *** START CLOCK ***
265  REM
270  PRINT #C  "R"
275  REM
280  REM  *** TAKE CLOCK  READING ***
285  REM
290  READ #C;B$
300  REM
305  REM  *** CONVERT SECONDS PART OF ASCII READING TO 2 DIGIT ***
310  REM  *** REAL FOR ARITHMETIC OPERATION                    ***
315  REM
320  CALLDLODE(B$[9,10],V,"F2.0")
330  PRINT V
335  REM
340  REM  *** HAVE 10 SEC  ELAPSED? IF NOT TAKE ANOTHER READING ***
345  REM
350  IF V-S<10THEN 290
355  REM
360  REM  *** IF SO, DISPLAY CURRENT TIME AND PRINT MESSAGE ***
365  REM
370  PRINT #D;B$
380  PRINT #F;"IT'S ABOUT TIME!"
390  FOR I=1TO 4
400  PRINT #F
410  NEXT I
415  REM
420  REM  *** CLEAR BUS AND END ***
425  REM
430  CALLCLEAR(C)
440  END
```

LEARNING MODULE III SOLUTIONS

1) What is the major difference between auto and direct addressing?

   In auto-addressing mode the HP-IB driver supplies all bus protocol, i.e. untalk, unlisten, ATN, REN, etc. In direct addressing mode, however, the user is responsible for bus protocol since commands and data are dumped directly to the bus with no driver intervention.

2) Write a program using direct addressing that will:

   a. Display on the numeric display and print on the thermal printer the current time from the digital clock.

   b. Clear the interface and exit.

   See Fortran and Basic listings of program LAB32 on following pages.

3)

CLOCK LU	48	SUBCHANNEL 5
PRINTER LU	50	SUBCHANNEL 7
DISPLAY LU	25	SUBCHANNEL 4

	ASCII	DESCRIPTION	OCTAL
CMDW:	−?	Untalk,Unlisten	137 77
	E	Clock Talk	105
	$	Display Listen	44
	'	Printer Listen	47

4) Auto-addressing is very easy to use. Direct addressing, however, has one major advantage over auto-addressing. Compare the programs you wrote for LM II-#1 and LM III-#2 and determine what that advantage is.

   Why are we able to make use of this capability using direct addressing but not auto-addressing?

   The advantage to using direct addressing over auto-addressing is flexibility. Direct addressing allows the user to simultaneously address multiple bus devices and to program them. This capability is not available in auto-addressing since access is to a single, specific LU at a time. Each READ/WRITE statement specifies one LU.

```
0001  FTN4,Q,L
0002        PROGRAM LAB32
0003        DIMENSION ICOM(5),ICOM1(2)
0004        DATA ICOM/5,2H_?,2HE$,2H' /,ICOM1/2,2H_?/
0005        IBLU=9
0006  C
0007  C SEND UNTLK, UNLSN, LOCK TALK, DISPLAY AND PRINTER LISTEN
0008  C
0009        CALL CMDW(IBLU,ICOM,0)
0010  C
0011  C LET DISPLAY AND PRINTER LISTEN FOR 327670 ITERATIONS OF THE LOOP
0012  C
0013        DO 20 J=1,10
0014        DO 10 I=1,32767
0015   10   CONTINUE
0016   20   CONTINUE
0017  C
0018  C SEND UNTLK, UNLSN AND EXIT
0019  C
0020        CALL CMDW(IBLU,ICOM1,0)
0021        END
```

```
0001    10 REM  *** LEARNING MODULE 3 - # 2 ***
0002    15 REM
0003    20 DIM A$[5],B$[2]
0004    25 REM
0005    30 REM  *** UNTLK, UNLSN, BUS LU ***
0006    35 REM
0007    40 LET B$="_?"
0008    50 LET B=9
0009    60 REM
0010    70 REM *UNTALK,UNLISTEN,CLOCK TALK,DISPLAY AND PRINTER LISTEN*
0011    80 REM
0012    90 LET A$="-?E$'"
0013    95 CALLCMDW(B,A$,0)
0014    100 REM
0015    110 REM *DELAY FOR DISPLAY TO ACCEPT DATA THEN UNTLK, UNLSN, EXIT *
0016    120 REM
0017    125 FOR I=1TO 2500
0018    130 NEXT I
0019    140 CALLCMDW(B,B$,0)
0020    150 END
```

LEARNING MODULE IV SOLUTIONS

1) Run program LAB41 and observe bus traffic on the analyzer and the DVM and timing generator front panels.

	ANALYZER	GENERATOR	DVM
ABRT	024(DCL)	NOT ADDRESSED LOCAL MODE	NOT ADDRESSED LOCAL MODE
RMOTE	?#(43B)REN ?&(46B)REN	REMOTE ADDRESSED	REMOTE
LLO	021(DCL)REN	NO FRONT PANEL RESPONSE	
LOCL	$\overline{\text{REN}}$	LOCAL MODE FRONT PANELS OPERATIVE	LOCAL MODE
CLEAR	?#(43B)004(SDC) ?&(46B)004(SDC)		

3) Write a program that will show the difference in speed between DMA and interrupt methods of data transfer.

   Does DMA always beat the interrupt method? If not, when is the interrupt method faster? How do you account for this?

   When readings of about 5 words or less are made, regardless of the number of those readings made or the size of the data buffer, the interrupt method is faster than DMA because of the overhead time involved in setting up DMA. When the readings are too small, DMA overhead takes more time than the actual data transfer. Since the number of interrupts required using the interrupt method is small, the interrupt method actually takes less time than the DMA overhead.

```
0001   FTN4,Q,L
0002         PROGRAM LAB41
0003   C
0004   C THIS PROGRAM ALLOWS THE STUDENT TO OBSERVE THE BUS TRAFFIC GENERATED
0005   C BY HP-IB MESSAGE ROUTINES ON THE BUS ANALYZER AND TO OBSERVE THE EFFECT
0006   C OF THESE MESSAGES ON INSTRUMENTATION VIA FRONT PANEL INDICATORS.
0007   C
0008         DIMENSION IDATA(1)
0009         IDVMLU=12
0010         IGENLU=49
0011         IBUSLU=9
0012   C
0013   C PROMPT USER TO OBSERVE ABRT(IBUSLU,2) ON ANALYZER AND PROCESS RESPONSE
0014   C
0015   5     WRITE(1,10)
0016   10    FORMAT("TYPE GO TO OBSERVE ABRT(IBUSLU,2) ON THE ANALYZER")
0017         READ(1,15) IDATA
0018   15    FORMAT(A2)
0019         IF(IDATA(1) .NE. 2HGO) GOTO 5
0020         CALL ABRT(IBUSLU,2)
0021   C
0022   C PROMPT USER TO OBSERVE RMOTE MESSAGE ON ANALYZER AND PROCESS RESPONSE
0023   C
0024         WRITE(1,20)
0025   20    FORMAT(//"TYPE EX TO STOP OR CR TO OBSERVE RMOTE(IDVMLU) AND",/,
0026        *"RMOTE(IGENLU) ON ANALYZER"//)
0027         READ(1,15) IDATA
0028         IF(IDATA(1) .EQ. 2HEX) GOTO 50
0029         CALL RMOTE(IDVMLU)
0030         CALL RMOTE(IGENLU)
0031   C
0032   C PROMPT USER TO OBSERVE LOCAL LOCKOUT MESSAGE ON BUS AND EXAMINE
0033   C DEVICE FRONT PANEL RESPONSE
0034   C
0035         WRITE(1,25)
0036   25    FORMAT(//"TYPE EX TO STOP OR CR TO OBSERVE LLO(BUSLU) ON",/,
0037        *"THE BUS ANALYZER"//)
0038         READ(1,15) IDATA
0039         IF(IDATA(1) .EQ. 2HEX) GOTO 50
0040         CALL LLO(IBUSLU)
0041         WRITE(1,26)
0042   26    FORMAT(//"SEE IF THE DVM AND/OR TIMING GENERATOR RESPOND TO",/,
0043        *"FRONT PANEL STIMULI"//)
0044   C
0045   C PROMPT USER TO OBSERVE LOCL MESSAGE ON ANALYZER AND PROCESS RESPONSE
0046   C
0047         WRITE(1,30)
0048   30    FORMAT(//"TYPE EX TO STOP OR CR TO OBSERVE LOCL(BUSLU)"//)
0049         READ(1,15) IDATA
0050         IF(IDATA(1) .EQ. 2HEX) GOTO 50
0051         CALL LOCL(IBUSLU)
0052         WRITE(1,35)
0053   35    FORMAT(//"NOW TRY THE DVM AND GENERATOR FRONT PANELS!!"//)
0054   C
0055   C PROMPT USER TO OBSERVE CLEAR MESSAGE ON ANALYZER AND PROCESS RESPONSE
0056   C
0057         WRITE(1,40)
0058   40    FORMAT(//"TYPE EX TO STOP OR CR TO OBSERVE CLEAR(IDVMLU,1) AND",/,
0059        *"CLEAR(IGENLU,1) ON THE ANALYZER"//)
0060         READ(1,15) IDATA
0061         IF(IDATA(1) .EQ. 2HEX) GOTO 50
0062   50    CALL CLEAR(IDVMLU,1)
0063         CALL CLEAR(IGENLU,1)
0064         END
```

```
0001   FTN4,L
0002         PROGRAM LAB42
0003   C  DVM LU:
0004         LUDM=12
0005   C  SCANNER LU:
0006         LUSC=52
0007   C  THER. PRINTER LU:
0008         LUTP=50
0009   C  DISPLAY LU:
0010         LUDP=25
0011   C
0012   C  INITIALIZE DEVICES
0013   C
0014         CALL CLEAR(LUDM,1)
0015         CALL CLEAR(LUSC,1)
0016         CALL CLEAR(LUTP,1)
0017         CALL RMOTE(LUDM)
0018         CALL RMOTE(LUSC)
0019         CALL RMOTE(LUTP)
0020   C
0021   C  SETUP DVM TO 2 WIRE RESISTANCE MEAS., AUTO-RANGE, HIGH
0022   C  RESOLUTION, HOLD/MANUAL TRIGGER
0023   C
0024         WRITE(LUDM,10)
0025    10   FORMAT("F4R7H1T3")
0026   C
0027   C  SCAN 10 CHANNELS TRIGGERING THE DVM AT EACH CHANNEL AND OUTPUTTING
0028   C  THE READING TO THE DISPLAY AND/OR PRINTER
0029
0030         DO 65 I=1,10
0031         J=I-1
0032   C
0033   C  SETUP SCANNER
0034   C
0035         WRITE(LUSC,20)J
0036    20   FORMAT("C0",I1,"E")
0037   C
0038   C  TRIGGER THE DVM
0039   C
0040         CALL TRIGR(LUDM)
0041   C
0042   C  READ FROM DVM
0043   C
0044         READ(LUDM,*)READ
0045   C
0046   C  PRINT ON THER. PRINTER AND DISPLAY ON NUMERIC DISPLAY
0047   C
0048         WRITE(LUTP,30)READ
0049         WRITE(LUDP,30)READ
0050    30   FORMAT(E11.6)
0051    65   CONTINUE
0052         WRITE(LUTP,40)
0053    40   FORMAT(//,"RESIST. MEASR.",//////)
0054   C
0055   C  DVM TO LOCAL:
0056   C
0057         CALL GTL(LUDM)
0058         CALL CLEAR(LUSC,1)
0059         END
```

```
0001    10  REM  *** LEARNING MODULE 4 -  2 ***
0002    15  REM
0003    20  DIM A$[4]
0004    25  REM
0005    30  REM  *** DVM, SCANNER, PRINTER, DISPLAY LU'S
0006    35  REM
0007    40  LET V=12
0008    50  LET S=52
0009    60  LET P=50
0010    65  LET D=25
0011    70  REM
0012    80  REM  *** INITIALIZE DEVICES ***
0013    90  REM
0014   100 CALL CLEAR(V,1)
0015   110 CALL CLEAR(S,1)
0016   120 CALL CLEAR(P,1)
0017   125 CALL RMOTE(V)
0018   130 CALL RMOTE(S)
0019   135 CALL RMOTE(P)
0020   140 REM
0021   145 REM  *** SET UP DVM TO RESIST. MEAS., AUTO-RANGE, HIGH
0022   150 REM  *** RESOLUTION, HOLD/MANUAL TRIGGER
0023   155 REM
0024   160 PRINT # V;"F4R7H1T3"
0025   165 LET A$="C00E"
0026   170 REM
0027   175 REM  *** SCAN 10 CHANNELS, TRIGGER THE DVM TO MAKE A READING
0028   180 REM  *** AT EACH CHANNEL AND OUTPUT THE READING TO THE PRINTER
0029   185 REM  *** AND DISPLAY
0030   190 REM
0031   200 FOR I=1 TO 10
0032   210 LET J=I-1
0033   215 REM
0034   220 REM  *** PROGRAM SCANNER ***
0035   225 REM
0036   230 CALL DCODE(J,A$[3,3],"(I1)")
0037   240 PRINT # S;A$
0038   245 REM
0039   250 REM  *** TRIGGER THE DVM ***
0040   255 REM
0041   260 CALL TRIGR(V)
0042   265 REM
0043   270 REM  *** READ FROM DVM ***
0044   275 REM
0045   280 READ # V;R
0046   285 REM
0047   290 REM  *** PRINT ON THERMAL PRINTER AND DISPLAY ON NUMERIC DISPLAY ***
0048   295 REM
0049   300 PRINT # P;R
0050   305 PRINT # D;R
0051   310 NEXT I
0052   315 PRINT # P
0053   320 PRINT # P
0054   325 PRINT # P;"RESIST. MEASR"
0055   330 PRINT # P
0056   335 PRINT # P
0057   340 PRINT # P
0058   345 REM
0059   350 REM  *** DVM TO LOCAL AND CLEAR SCANNER ***
0060   355 REM
0061   360 CALL GTL(V)
0062   365 CALL CLEAR(S,1)
0063   370 END
```

```
0001   FTN4,Q,L
0002         PROGRAM DVSI
0003   C
0004   C THIS PROGRAM SHOULD DRAMATIZE THE DIFFERENCE BETWEEN DMA AND INTERRUPT
0005   C METHOD TRANSFERS. THE NUMBER OF READINGS TO BE MADE IS SPECIFIED BY THE
0006   C USER. FIRST THE DESIRED NUMBER OF READINGS IS MADE VIA INTERRUPT METHOD
0007   C THEN BY DMA AND BOTH TIMES PRINTED OUT ON THE SPECIFIED LIST DEVICE.
0008   C
0009   C THE RUN STRING IS :RU,DVSI, READINGS TO TAKE,LENGTH OF READING
0010   C IN POSITIVE WORDS,LIST DEVICE(DEFAULTS TO TERMINAL)
0011   C
0012         DIMENSION IDATA(100),IBUF(5),ITIME(5)
0013         IGLU = 49
0014         ICLU = 48
0015         IBLU = 9
0016         J = 0
0017   C
0018   C RETRIEVE RUN STRING TO GET NUMBER OF READINGS TO TAKE, LENGTH OF
0019   C READINGS, AND LIST DEVICE
0020   C
0021         CALL RMPAR(IBUF)
0022         IREAD=IBUF(1)
0023         ILNG=IBUF(2)
0024         ILIST=IBUF(3)
0025   C
0026   C LIST DEFAULTS TO TERMINAL. SET UP LENGTH OF TIMING GENERATOR READ.
0027   C
0028         IF(ILIST .EQ. 0) ILIST = 1
0029         ITLNG=10
0030   C
0031   C CLEAR AND REMOTE BUS DEVICES
0032   C
0033   5     CALL CLEAR(IBLU,2)
0034         CALL RMOTE(IGLU)
0035   C
0036   C SET UP TIMING GENERATOR TO PACER MODE, 1 MILLESEC. PERIOD, TRIGGER/RESET
0037   C
0038   10    WRITE(IGLU,20)
0039   20    FORMAT("PRD001E3")
0040   C
0041   C RECONFIGURE CLOCK TO "NO EOR REQUIRED" IF FIRST PASS AND DMA AND "NO
0042   C EOR REQUIRED" IF SECOND PASS
0043   C
0044         CALL CNFG(ICLU,1,7000B)
0045         IF(J .EQ. 1) CALL CNFG(ICLU,1,27000B)
0046   C
0047   C START TIMING GENERATOR AND TAKE SPECIFIED NUMBER OF READINGS OF
0048   C SPECIFIED LENGTH FROM CLOCK
0049   C
0050         CALL TRIGR(IGLU)
0051         DO 30 I=1,IREAD
0052         CALL EXEC(1,ICLU,IDATA,ILNG)
0053   30    CONTINUE
0054   C
0055   C TAKE TIME READING FROM GENERATOR AND OUTPUT TO LIST DEVICE
0056   C
0057         CALL EXEC(1,IGLU,ITIME,ITLNG)
0058         WRITE(ILIST,40) ITIME
0059   40    FORMAT(5A2," MILLESECONDS")
0060   C
0061   C INCREMENT COUNTER. IF FIRST PASS, GO TO NEXT PASS. IF SECOND PASS,
0062   C CLEAR BUS, RESTORE DEVICE CONFIGURATIONS, AND TERMINATE.
0063   C
0064         J=J+1
0065         IF(J .EQ. 2) GOTO 50
0066         GOTO 5
0067   50    CALL CNFG(ICLU,2)
0068         CALL CLEAR(IBLU,2)
0069         END
```

LEARNING MODULE V SOLUTIONS

2) Name 4 things that must happen before an SRQ may be successfully processed.

   1. %2DV37, the HP-IB driver with SRQ capability must have been generated into the system.
   2. The RTE HP-IB library, %IB4A, must be loaded so that the SRQ and SRQSN subroutine calls can be made.
   3. The SRQ service program must be established and its starting address placed in the device BEQT via the SRQ subroutine call.
   4. The device which is to assert SRQ must be enabled to do so.

3/7) Run the program of #3/7 but first use the FMGR :OF command to purge the service program. What happens?

The bus is set down and the driver returns the error message:

```
IONR L* A EB SC 4
```

where: A = device lu (DVM for #3, generator for #7)
B = bus EQT
C = bus subchannel # of device
4 = driver error code = service prog. not found

Reload the service program. In your main program replace the service program name with blanks. Recompile and load your main program. What do you expect to happen when you run the program? Run it. What happened?

Once again the service program is not found because it was never established and once again the bus EQT goes down with the service program not found error code.

5) Repeat the service program manipulations discussed in #3. Does your error handling routine adequately handle the I/O errors obtained? Note that the bus no longer requires restoration, i.e. it is not set down automatically on an I/O error.

   The only difference between system error handling and user error handling is in the format of the error message and in the fact that the user must check for I/O errors in his/her program. A nice advantage to user error handling is that the bus is not automatically set down when an I/O error occurs.

6) What would happen if the DVM were set up to assert SRQ after each measurement but the SRQ call was never made in your program?

   The DVM will still assert SRQ but since no service program was ever established to handle the SRQ, no action is taken. The bus is not set down but neither is the SRQ line cleared.

```
0001   FTN4,Q,L
0002         PROGRAM EXEC5
0003   C
0004   C THIS PROGRAM SETS UP THE TIMING GENERATOR, LETS IT COUNT TEN CYCLES,
0005   C AND OUTPUTS THE CURRENT CLOCK READING TO THE DISPLAY AND PRINTER.
0006   C
0007         DIMENSION IGBUF(6),INPUT(5),IDATA(4),ICBUF(6),IDBUF(5),IPBUF(9)
0008         EQUIVALENCE(INPUT(2),IDATA)
0009         EQUIVALENCE(ICBUF(2),IDBUF)
0010         DATA IGBUF/10,2HP,2HRD,2H00,2H1E,2H6R/
0011   C
0012   C SET UP BUS, GENERATOR, CLOCK, DISPLAY, PRINTER LU'S
0013   C
0014         IBLU=9
0015         IGLU=49
0016         ICLU=48
0017         IDLU=25
0018         IPLU=50
0019   C
0020   C REMOTE ENABLE GENERATOR
0021   C
0022         CALL EXEC(3,1661B)
0023   C
0024   C SET UP GENERATOR TO PACER, TRIGGER/RESET, SRQ DISABLE, 1 SEC. INTERVAL
0025   C
0026         IGLNG=-12
0027         CALL EXEC(2,IGLU,IGBUF,IGLNG)
0028   C
0029   C READ GENERATOR OUTPUT
0030   C
0031         INLNG=-10
0032   10    CALL EXEC(1,IGLU,INPUT,INLNG)
0033   C
0034   C HAVE 10 SECONDS ELAPSED? IF NOT, LOOP BACK AND READ AGAIN
0035   C
0036         IF(IDATA(3) .LT. 2H10) GOTO 10
0037   C
0038   C IF SO, READ CURRENT CLOCK TIME
0039   C
0040         ICLNG=-14
0041         CALL EXEC(1,ICLU,ICBUF,ICLNG)
0042   C
0043   C FORMAT CLOCK READING FOR PRINTER OUTPUT
0044   C
0045         IPBUF(1)=ICBUF(2)
0046         IPBUF(2)=1H:
0047         IPBUF(3)=ICBUF(3)
0048         IPBUF(4)=1H:
0049         IPBUF(5)=ICBUF(4)
0050         IPBUF(6)=1H:
0051         IPBUF(7)=ICBUF(5)
0052         IPBUF(8)=1H:
0053         IPBUF(9)=ICBUF(6)
0054   C
0055   C OUTPUT CURRENT CLOCK TIME TO DISPLAY AND PRINTER
0056   C
0057         IDLNG=-10
0058         IPLNG=-18
0059         CALL EXEC(2,IDLU,IDBUF,IDLNG)
0060         CALL EXEC(2,IPLU,IPBUF,IPLNG)
0061   C
0062   C CLEAR GENERATOR, DISPLAY, CLOCK, AND PRINTER
0063   C
0064         CALL EXEC(3,IGLU)
0065         CALL EXEC(3,ICLU)
0066         CALL EXEC(3,IDLU)
0067         CALL EXEC(3,IPLU)
0068         END
```

```
0001  FTN4,L
0002        PROGRAM LAB53
0003  C
0004  C THIS PROGRAM SETS UP THE DVM AND SCANNER FOR A RESISTANCE MEASURE
0005  C ON CHANNEL 0, SCHEDULES A SERVICE PROGRAM THEN TERMINATES AND THE
0006  C SERVICE PROGRAM TAKES OVER
0007  C
0008        DIMENSION IPROG(4)
0009        DATA IPROG/5,2HSE,2HRV,2HE /
0010        LUSC=52
0011        LUDM=12
0012        IBLU=9
0013  C
0014  C INITIALIZE AND REMOTE HP-IB DEVICES:
0015  C
0016        CALL CLEAR(LUSC,1)
0017        CALL CLEAR(LUDM,1)
0018        CALL RMOTE(LUSC)
0019        CALL RMOTE(LUDM)
0020  C
0021  C  SETUP THE DVM TO 2 WIRE RES. MEAS., AUTO-RANGE, HIGH RESOLUTION,
0022  C  DATA READY RQS ON, HOLD/MANUAL TRIGGER
0023  C
0024        WRITE(LUDM,10)
0025   10   FORMAT("F4R7H1D1T3")
0026  C
0027  C  SETUP SCANNER TO CHANNEL 0
0028  C
0029        WRITE(LUSC,20)
0030   20   FORMAT("C00E")
0031  C
0032  C  TRIGGER THE DVM:
0033  C
0034        CALL TRIGR(LUDM)
0035  C
0036  C  SCHEDULE "SERVE" UPON SRQ:
0037  C
0038        CALL SRQ(LUDM,16,IPROG)
0039        END
```

```
0001   10   REM  *** LEARNING MODULE 5 - #3 ***
0002   15   REM
0003   20   REM  *** DVM AND SCANNER LU'S ***
0004   25   REM
0005   30   LET V=52
0006   40   LET S=51
0007   45   REM
0008   50   REM  *INITIALIZE DEVICES*
0009   55   REM
0010   60   CALL CLEAR(V,1)
0011   70   CALL CLEAR(S,1)
0012   80   CALL RMOTE(V)
0013   90   CALL RMOTE(S)
0014   95   REM
0015   100  REM  *** SETUP DVM TO RESISTANCE MEAS., AUTO-RANGE, HIGH RES.,
0016   105  REM  *** DATA READY RQS ON, HOLD/MANUAL TRIGGER
0017   110  REM
0018   115  PRINT #V;"F4R7H1D1T3"
0019   120  REM
0020   125  REM  *** SET UP SCANNER TO CHANNEL 0 ***
0021   130  REM
0022   140  PRINT #S;"C00E"
0023   150  REM
0024   160  REM  *** TRIGGER THE DVM ***
0025   165  REM
0026   170  CALL TRIGR(V)
0027   175  REM
0028   180  REM  *** SCHEDULE "SERVE" UPON SRQ ***
0029   185  REM
0030   190  CALL SRQ(V,16,"SERVE")
0031   200  END
```

```
0001  FTN4,L
0002        PROGRAM SERVE
0003  C
0004  C THIS SERVICE PROGRAM READS A RES. MEAS. FROM THE DVM, OUTPUTS IT
0005  C TO THE THERMAL PRINTER, THEN TRIGGERS THE DVM, INCREMENTCS THE SCANNER
0006  C CHANNEL AND TERMINATES SAVING RESOURCES UNTIL A RESISTANCE MEASURE
0007  C HAS BEEN MADE AT EACH SCANNER CHANNEL
0008  C
0009        I=1
0010        LUDM=12
0011  C
0012  C  TAKE READING FROM DVM:
0013  C
0014   550  READ(LUDM,*)READ
0015  C
0016  C  PRINT READING ON THER. PRINTER
0017  C  THER. PRINTER LU:
0018  C
0019        LUTP=50
0020        WRITE(LUTP,10)READ
0021    10  FORMAT(E11.6)
0022        IF(I-10)20,30, 30
0023  C
0024  C  INCREMENT SCANNER:
0025  C  SCANNER LU:
0026  C
0027    20  LUSC=52
0028        WRITE(LUSC,40)I
0029    40  FORMAT("C0",I1,"E")
0030  C
0031  C  TRIGGER DVM
0032  C
0033        CALL TRIGR(LUDM)
0034  C
0035  C  TERMINATE PROGRAM WITH SAVING RESOURCES
0036  C
0037        CALL EXEC(6,0,1)
0038        I=I+1
0039        GO TO 550
0040  C
0041  C  DISABLE SRQ
0042  C
0043    30  CALL SRQ(LUDM,17)
0044  C
0045  C  DVM TO LOCAL
0046  C
0047        CALL GTL(LUDM)
0048        WRITE(LUTP,50)
0049    50  FORMAT(//,"RESIST. MEASR."//////)
0050        END
```


```
0001  FTN4,Q,L
0002        PROGRAM LAB57
0003  C
0004  C THIS PROGRAM SETS UP THE TIMING GENERATOR, ESTABLISHES A SERVICE
0005  C PROGRAM TO BE SCHEDULED ON SRQ THEN EXITS.
0006  C
0007        DIMENSION IPROG(4)
0008        DATA IPROG/5,2HTI,2HME,2H5 /
0009  C
0010  C ASSIGN BUS, GENRATOR LU'S
0011  C
0012        IBLU=9
0013        IGLU=49
0014  C
0015  C ESTABLISH SERVICE PROGRAM TO BE SCHEDULED ON SRQ FROM TIMING
0016  C GENERATOR
0017  C
0018        CALL SRQ(IGLU,16,IPROG)
0019  C
0020  C CLEAR BUS
0021  C
0022        CALL CLEAR(IBLU,2)
0023  C
0024  C REMOTE ENABLE GENERATOR
0025  C
0026        CALL RMOTE(IGLU)
0027  C
0028  C SET UP TIMING GENERATOR TO TIMING MODE, 10 SEC. INTERVAL,
0029  C DISABLE REAR PANEL TRIGGER, ENABLE SERVICE REQUEST AND
0030  C TRIGGER THE GENERATOR. THEN QUIT AND LET THE SERVICE ROUTINE
0031  C DO THE REST.
0032  C
0033        WRITE(IGLU,20)
0034  20    FORMAT("T001E7USR")
0035        END
```

```
0001  FTN4,Q,L
0002        PROGRAM TIME5
0003  C
0004  C THIS PROGRAM IS THE SERVICE PROG. FOR &LAB67. ON RECEIPT OF SRQ
0005  C FROM THE TIMING GENERATOR THIS PROGRAM IS SCHEDULED AND OUTPUTS
0006  C A MESSAGE AND THE CURRENT TIME TO THE PRINTER. IT THEN TRIGGERS
0007  C THE TIMING GENERATOR TO START A NEW TIMING CYCLE AND PERFORMS
0008  C THE SAME FUNCTIONS ON SRQ. AFTER ONE MINUTE IT CLEARS THE BUS AND
0009  C EXITS.
0010  C
0011        DIMENSION IDATA(5)
0012  C
0013  C INITIALIZE COUNTER AND GENERATOR, CLOCK, PRINTER LU'S
0014  C
0015        I=1
0016        IGLU=49
0017        ICLU=48
0018        IPLU=50
0019  C
0020  C TAKE CURRENT CLOCK READING
0021  C
0022  5     READ(ICLU,10) IDATA
0023  10    FORMAT(2X,5A2)
0024  C
0025  C OUTPUT MESSGE AND CURRENT TIME TO PRINTER
0026  C
0027        WRITE(IPLU,20) IDATA
0028  20    FORMAT(X,"IT'S ABOUT TIME",/,A2,4(":",A2),/////)
0029  C
0030  C TRIGGER NEXT TIMING SEQUENCE
0031  C
0032        CALL TRIGR(IGLU)
0033  C
0034  C EXIT SAVING RESOURCES
0035  C
0036        CALL EXEC(6,0,1)
0037  C
0038  C INCREMENT COUNTER. HAS ONE MINUTE ELAPSED? IF NOT, TAKE NEXT CLOCK
0039  C READING. IF SO, CLEAR BUS, DISABLE SERVICE PROG. FROM BEING
0040  C SCHEDULED AGAIN AND EXIT.
0041  C
0042        I=I+1
0043        IF(I .LT. 7) GOTO 5
0044        CALL CLEAR(IBLU,2)
0045        CALL SRQ(IGLU,17)
0046        END
```

LEARNING MODULE VI SOLUTIONS

1) What does the statement CALL SRQ (device lu,17) actually do to keep a service program from being scheduled?

   SRQ(devicelu,17) causes the HP-IB driver to go to the EQT extension area of the indicated device and clear the service program name from words 2-4 of the BEQT. Then, when SRQ is asserted, the driver does a serial poll to determine which device wants SRQ and goes to the device's BEQT to find its service program. No service program name is found so no service program is scheduled.

   Why is it a good idea to use this call before terminating a service program?

   It is usually a good idea to disable a device's service routine from being scheduled right before it terminates so that if the device happens to assert SRQ unexpectedly sometime after the routine has terminated, the service program won't get scheduled accidently and mess up present bus I/O operations.

2) What must occur before an I/O error can be handled by a user's program? What is the default procedure for error handling? How could we change the default error handling to the user?

   In order for a program to do its own error handling:

   a. The RTE HP-IB Library, %IB4A, must be loaded so that the IBERR routine will be callable.
   b. The E-bit of the device configuration word must be set using the CNFG routine or an EXEC call.

   Error handling defaults to the system normally. We could alter the default for error handling to the user by setting the E-bits of all devices' configuration word although the only permanent way to do this would be to rewrite the HP-IB driver.

```
0001  FTN4,L
0002        PROGRAM LAB6
0003  C
0004  C THIS PROGRAM WILL SET UP THE DVM, SCANNER, TIMING GENERATOR AND A
0005  C SERVICE PROGRAM SO THAT THE DVM WILL TAKE RESISTANCE READINGS FROM
0006  C ALL 10 CHANNELS OF THE SCANNER EVERY 5 SECONDS AND OUTPUT THE VALUES
0007  C TO THE THERMAL PRINTER
0008  C
0009        DIMENSION IPROG(4),IPRAM(5)
0010        DATA IPROG/5,2HRE,2HAD,2HM /
0011        CALL RMPAR(IPRAM)
0012        IF(IPRAM(1).EQ.0)IPRAM(1)=1
0013  C
0014  C DVM AND TIMING GENERATOR LU'S
0015  C
0016        LUDM=12
0017        LUTG=49
0018  C
0019  C  INITIALIZE, REMOTE AND CONFIGURE DEVICES FOR USER ERROR HANDLING
0020  C
0021        CALL CLEAR(LUDM,1)
0022        CALL CLEAR(LUTG,1)
0023        CALL RMOTE(LUDM)
0024        CALL RMOTE(LUTP)
0025        CALL CNFG(LUDM,1,17400B)
0026        CALL CNFG(LUTP,1,17400B)
0027  C
0028  C  SETUP DVM TO RESISTANCE MEAS., AUTO-RANGE, HIGH RESOLUTION, HOLD/
0029  C  MANUAL TRIGGER
0030
0031        WRITE(LUDM,10)
0032   10   FORMAT("F4R7H1T3")
0033  C
0034  C  CHECK FOR I/O ERROR
0035  C
0036        IERR=IBERR(LUDM)
0037        IF(IERR)20,30,20
0038  C
0039  C  SETUP TIMING GEN. TO PACER MODE, 5 SEC. INTERVAL, SRQ ENABLE,
0040  C  TRIGGER/RESET
0041
0042   30   WRITE(LUTG,40)
0043   40   FORMAT("005E6PSR")
0044  C
0045  C  CHECK FOR I/O ERROR
0046  C
0047        IERR=IBERR(LUTG)
0048        IF(IERR)20,50,20
0049  C
0050  C  SCHEDULE "READM" UPON SRQ
0051  C
0052   50   CALL SRQ(LUTG,16,IPROG)
0053        GO TO 100
0054   20   WRITE(IPRAM(1),60)IERR
0055   60   FORMAT("HP-IB ERROR NO.",I1)
0056   100  END
```

```
0001   10  REM  *** LEARNING MODULE 6 - #3 ***
0002   20  REM
0003   30  REM *** DVM, TIMING GENERATOR LU'S
0004   35  REM
0005   40  LET V=12
0006   50  LET G=49
0007   60  REM
0008   70  REM *** CLEAR, INITIALIZE AND CONFIGURE DEVICES FOR USER ERROR HANDLING
0009   80  REM
0010   90  CALL CLEAR(V,1)
0011   100 CALL CLEAR(G,1)
0012   110 CALL RMOTE(V)
0013   120 CALL RMOTE(G)
0014   130 CALL CNFG(V,1,7936)
0015   140 CALL CNFG(G,1,7936)
0016   150 REM
0017   160 REM *** SET UP DVM TO RES. MEAS., AUTO-RANGE, HIGH RES., HOLD/
0018   165 REM *** MANUAL TRIGGER
0019   170 REM
0020   180 PRINT #V;"F4R7H1T3"
0021   190 REM
0022   200 REM *** CHECK FOR I/O ERROR ***
0023   210 REM
0024   220 LET I=IBERR(V)
0025   230 IF I>0 THEN 370
0026   240 REM
0027   250 REM *** SET UP GENERATOR TO PACER MODE, 5 SEC. INTERVAL, TRIGGER/
0028   255 REM *** RESET, SRQ ENABLE
0029   260 REM
0030   270 PRINT #G;"005E6PSR"
0031   280 REM
0032   290 REM *** CHECK FOR I/O ERROR ***
0033   300 REM
0034   310 LET I=IBERR(G)
0035   320 IF I>0 THEN 370
0036   330 REM
0037   340 REM *** SCHEDULE "READM" UPON SRQ ***
0038   350 REM
0039   360 CALL SRQ(G,16,"READM")
0040   365 GOTO 380
0041   370 PRINT "HP-IB ERROR NO.";I
0042   380 END
0043
0044
```

```
0001   FTN4,L
0002         PROGRAM READM
0003   C
0004   C THIS SERVICE PROGRAM READS DVM RESISTANCE MEASUREMENTS FROM ALL 10
0005   C SCANNER CHANNELS AND OUTPUTS THEM TO THE THERMAL PRINTER ALONG WITH
0006   C THE CLOCK TIME AS A HEADING. AFTER 25 SECONDS IT DISABLES ITSELF
0007   C FROM BEING SCHEDULED AND TERMINATES
0008   C
0009         DIMENSION ICLK(7)
0010         I=1
0011   C
0012   C SET CLOCK, SCANNER, DVM, PRINTER, GENERATOR LU'S
0013   C
0014         LUCK=48
0015         LUSC=52
0016         LUDM=12
0017         LUTP=50
0018         LUTG=49
0019   C
0020   C READ TIME FROM CLOCK
0021   C
0022    550  READ(LUCK,10)ICLK
0023    10   FORMAT(2X,7A2)
0024   C
0025   C SET UP SCANNER, DVM TO TAKE RESISTANCE MEASURES FROM 10 SCANNER CHANNELS
0026   C
0027         DO 60 K=1,10
0028         J=K-1
0029   C
0030   C  SETUP SCANNER
0031   C
0032         WRITE(LUSC,20)J
0033    20   FORMAT("C0",I1,"E")
0034   C
0035   C  CHECK ERROR
0036   C
0037         IERR=IBERR(LUSC)
0038         IF(IERR)40,50,40
0039   C
0040   C  TRIGGER DVM
0041   C
0042    50   CALL TRIGR(LUDM)
0043   C
0044   C  READ RESISTANCE MEASUREMENT
0045   C
0046         READ(LUDM,*)DVM
0047   C
0048   C  PRINT MEASUREMENTS ON THERMAL PRINTER
0049   C
0050         WRITE(LUTP,30)DVM
0051    30   FORMAT(E11.6)
0052   C
0053   C  CHECK ERROR
0054   C
```

```
0055          IERR=IBERR(LUTP)
0056          IF(IERR)40,60,40
0057     60   CONTINUE
0058     C
0059     C  PRINT TIME ON THERMAL PRINTER
0060     C
0061          WRITE(LUTP,80) ICLK
0062     80   FORMAT(///,A2,4(":",A2),//////)
0063          IF(I-5)85,100,100
0064     C
0065     C  TRIGGER THE TIMING GEN.
0066     C
0067     85   CALL TRIGR(LUTG)
0068     C
0069     C  TERMINATE PROGRAM WITH SAVING RESOURCES
0070     C
0071          CALL EXEC(6,0,1)
0072          I=I+1
0073          GO TO 550
0074     40   WRITE(1,70)IERR
0075     70   FORMAT("HP-IB ERROR NO.",I1)
0076     C
0077     C  DISABLE SRQ
0078     C
0079     100  CALL SRQ(LUTG,17)
0080     C
0081     C  DVM AND TIMING GEN. TO LOCAL
0082     C
0083          CALL GTL(LUDM)
0084          CALL GTL(LUTG)
0085          END
```

```
0001  FTN4,Q,L
0002        PROGRAM LAB64
0003        DIMENSION IBUF(5),IPROG(4)
0004        DATA IPROG/5,2HSE,2HRV,2HE /
0005  C
0006  C GENERATOR, PRINTER, DISPLAY, CLOCK LU'S
0007  C
0008        LUTG=49
0009        LUTP=50
0010        LUDP=25
0011        LUCK=48
0012  C
0013  C CLEAR AND INITIALIZE DEVICES
0014  C
0015        CALL CLEAR(LUTG,1)
0016        CALL CLEAR(LUTP,1)
0017        CALL CLEAR(LUDP,1)
0018        CALL RMOTE(LUTP)
0019        CALL RMOTE(LUTG)
0020  C
0021  C SET UP TIMING GENERATOR TO PACER MODE, 10 SEC. PERIOD, TRIGGER/
0022  C RESET, SRQ ENABLE
0023  C
0024        WRITE(LUTG,10)
0025  10    FORMAT("PRS010E6")
0026  C
0027  C TAKE CLOCK READINGS UNTIL 12:00:00
0028  C
0029  15    READ(LUCK,20) IBUF
0030  20    FORMAT(5A2)
0031        IF(IBUF(3) .NE. 2H12) GOTO 30
0032        IF(IBUF(4) .NE. 2H00) GOTO 30
0033        IF(IBUF(5) .NE. 2H00) GOTO 30
0034        GOTO 15
0035  C
0036  C OUTPUT RANSOM MESSAGE TO PRINTER AND TRIGGER GENERATOR TO START 2
0037  C MINUTE COUNTDOWN
0038  C
0039  30    WRITE(LUTP,40)
0040  40    FORMAT("YOU HAVE 2 MINUTES",//,"TO FREE THE CUPERTINO",//,"BINARY",//,
0041        CALL TRIGR(LUTG)
0042  C
0043  C SET UP SERVICE PROGRAM TO BE SCHEDULED ON SRQ AND TERMINATE
0044  C
0045        CALL SRQ(LUTG,16,IPROG)
0046        END
0047
```

```
0001   FTN4,Q,L
0002         PROGRAM SERVE
0003   C
0004   C THIS SERVICE PROGRAM DISPLAYS THE TIME REMAINING OF THE ORIGINAL
0005   C TWO MINUTES AT 10 SEC. INTERVALS. AT THE END OF TWO MINUTES, IT
0006   C SETS THE FUSE AND AT THE END OF THE FUSE IT PROVIDES A MOCK "BOOM"
0007   C
0008   C PRINTER, GENERATOR, DISPLAY LU'S
0009   C
0010         I=1
0011         LUTP=50
0012         LUTG=49
0013         LUDP=25
0014   C
0015   C OUTPUT REMAINING TIME TO PRINTER, DISPLAY
0016   C
0017   10    J=I*10
0018         K=120-J
0019         IF(K .EQ. 0) GOTO 40
0020         WRITE(LUTP,20) K
0021   20    FORMAT("HURRY! ONLY ",I3," SECONDS LEFT!",/////)
0022         WRITE(LUDP,30) K
0023   30    FORMAT(I3)
0024   C
0025   C UPDATE TIME COUNTER AND TERMINATE SAVING RESOURCES
0026   C
0027         I=I+1
0028   35    CALL EXEC(6,,1)
0029         IF(IBOOM .EQ. 1) GOTO 80
0030         GOTO 10
0031   C
0032   C PROMPT TERRORIST LEADER FOR FUSE LENGTH
0033   C
0034   40    WRITE(1,50)
0035   50    FORMAT("ENTER FUSE TIME IN SECONDS: ")
0036         READ(1,60) ITIME
0037   60    FORMAT(I3)
0038   C
0039   C SET FUSE TIME BY SETTING UP GENERATOR
0040   C
0041         WRITE(LUTG,70) ITIME
0042   70    FORMAT("TRS",I3,"E6R")
0043   C
0044   C SET UP FLAG TO TELL THAT THIS IS THE TIME THAT SRQ MAKES A BIG BOOM
0045   C
0046         IBOOM=1
0047         GOTO 35
0048   C
0049   C SET OFF PSEUDO BOMB
0050   C
0051   80    WRITE(LUTP,90)
0052   90    FORMAT("BOOOOOOOOOM",/////)
0053   C
0054   C CLEAR DEVICES, DISABLE SERVICE PROGRAM, AND TERMINATE
0055   C
0056         CALL CLEAR(LUTG,1)
0057         CALL CLEAR(LUTP,1)
0058         CALL SRQ(LUTG,17)
0059         END
0060
```

LEARNING MODULE VII SOLUTIONS

2) When are device subroutines an advantage over the usual method of device programming? Do device subroutines save time and memory?

Device subroutines constitute a programming advantage whenever a device must be programmed several times in the same program and especially when the device's programming codes change several times in the program. It is much easier for a programmer to make a one line subroutine call than to write out the device programming sequence each time.

Using device subroutines saves neither CPU time nor memory. It does save the programmer's time.

```
0001   FTN4,Q,L
0002         PROGRAM LAB71
0003         COMMON IP(5)
0004         CALL RMPAR(IP)
0005         IF(IP(1).EQ.0)IP(1)=1
0006   C  DVM LU:
0007         LUDM=12
0008   C  SCANNER LU:
0009         LUSC=52
0010   C  THER. PRINTER LU:
0011         LUTP=50
0012   C  DISPLAY LU:
0013         LUDP=25
0014   C
0015   C  INITIALIZE AND REMOTE DEVICES
0016   C
0017         CALL CLEAR(LUDM,1)
0018         CALL CLEAR(LUSC,1)
0019         CALL RMOTE(LUDM)
0020         CALL RMOTE(LUSC)
0021         CALL RMOTE(LUTP)
0022   C
0023   C  SETUP DVM USING DEVICE SUBROUTINE
0024   C
0025         CALL DVM(LUDM,4,7,3,0,0,0,0)
0026   C
0027   C  TAKE RES. MEAS. FROM 10 SCANNER CHANNELS USING A DEVICE SUBROUTINE
0028   C
0029         DO 65 I=1,10
0030         ICHN=I-1
0031         CALL DVMRD(LUDM,LUSC,R1,ICHN)
0032   C
0033   C OUTPUT READINGS TO THERMAL PRINTER
0034
0035         WRITE(LUTP,30)R1
0036    30   FORMAT(E11.6)
0037    65   CONTINUE
0038         WRITE(LUTP,40)
0039    40   FORMAT(//,"RESIST. MEASR.",//////)
0040   C
0041   C  DVM TO LOCAL AND CLEAR SCANNER
0042   C
0043         CALL CLEAR(LUSC,1)
0044         CALL GTL(LUDM)
0045         END
```

```
0001   10  REM  *** LEARNING MODULE 7 - #1 ***
0002   15  REM
0003   20  REM  *** DVM, SCANNER, PRINTER LU'S
0004   25  REM
0005   30  V=12
0006   40  S=52
0007   50  P=50
0008   60  REM
0009   70  REM  *** INITIALIZE AND REMOTE DEVICES ***
0010   75  REM
0011   80  CLEAR(V,1)
0012   90  CLEAR(S,1)
0013   95  CLEAR(P,1)
0014  100  RMOTE(V)
0015  110  RMOTE(S)
0016  115  RMOTE(P)
0017  120  REM
0018  125  REM *** SETUP DVM VIA DEVICE SUBRUOTINE ***
0019  130  REM
0020  135  DVM(V,4,7,3,0,0,0,0)
0021  140  REM
0022  150  REM *** TAKE RES. MEASUREMENT OF 10 SCANNER CHANNELS VIA DEVICE SUB.
0023  150  REM *** AND OUTPUT TO THERMAL PRINTER
0024  155  REM
0025  160  FOR I=1 TO 10
0026  165  C=I-1
0027  170  DVMRD(V,S,R1,C)
0028  180  PRINT #P;R1
0029  190  NEXT
0030  200  PRINT #P
0031  210  PRINT #P
0032  220  PRINT #P;"RESIST. MEASR."
0033  230  PRINT #P
0034  240  PRINT #P
0035  250  PRINT #P
0036  255  REM
0037  260  REM *** DVM TO LOCAL AND CLEAR SCANNER
0038  265  REM
0039  270  CLEAR(S,1)
0040  275  GTL(D)
0041  280  END
```

```
0001  FTN4,Q,L
0002        SUBROUTINE DVM(LUDM,IF,IR,IT,IM,IA,IH,ID)
0003  C
0004  C LUDM=DVM LU
0005  C IF=FUNCTION CODE                    IR=RANGE CODE
0006  C IT=TRIGGER CODE                     IM=MATH ON/OFF CODE
0007  C IA=AUTO-CAL. ON/OFF CODE            IH=HIGH RESOLUTION CODE
0008  C ID=DATA READY RQS CODE
0009  C
0010  C FOR EXAMPLE:
0011  C TO DET THE DVM TO DC VOLT,AUTO RANGE,HOLD/MANUAL TRIGGER,MATH OFF,
0012  C AUTO CAL. OFF,HIGH RES. OFF,DATA READY RQS ON, THE CALL WILL BE:
0013  C CALL DVM(LUDM,1,7,3,3,0,0,1)
0014  C
0015        COMMON IP(5)
0016  C
0017  C SET UP THE DVM
0018  C
0019        WRITE(LUDM,10) IF,IR,IT,IM,IA,IH,ID
0020  10    FORMAT("F",I1,"R",I1,"T",I1,"M",I1,"A",I1,"H",I1,"D",I1)
0021  C
0022  C CHECK FOR I/O ERROR
0023  C
0024        IERR=IBERR(LUDM)
0025        IF(IERR) 40,60,40
0026  40    WRITE(IP(1),20) IERR
0027  20    FORMAT("HP-IB ERROR NO.",I1)
0028  60    END
```

```
0001  FTN4,Q,L
0002        SUBROUTINE DVMRD(LUDM,LUSC,R1,ICHN)
0003  C
0004  C THIS SUBROUTINE WILL SET UP THE SCANNER TO CHANNEL ICHN, TRIGGER
0005  C THE DVM AND TAKE A READING FROM THE DVM AND STORE IT IN R1.
0006  C
0007        COMMON IP(5)
0008  C
0009  C SET UP SCANNER
0010  C
0011        WRITE(LUSC,10) ICHN
0012  10    FORMAT("C0",I1,"E")
0013  C
0014  C CHECK FOR I/O ERROR
0015  C
0016        IERR=IBERR(LUSC)
0017        IF(IERR) 40,60,40
0018  40    WRITE(IP(1),20) IERR
0019  20    FORMAT("HP-IB ERROR NO.",I1)
0020        GOTO 70
0021  60    CALL TRIGR(LUDM)
0022        READ(LUDM,*) R1
0023  70    END
```

LEARNING MODULE VIII SOLUTIONS

2) This solution assumes that some programming codes overlap into the next answer and that the programming sequences will be done sequentially. Therefore, some command mode steps are not repeated in adjacent sequences.

a. Clear the interface

```
C1      IFC
```

b. Set the controller (interface) to talk

```
C60     COMMAND MODE
D100    INTERFACE TALK ADDRESS.
```

c. Set the clock and printer to listen

```
C60     COMMAND MODE
C3      REMOTE ENABLE.
D45     CLOCK LISTEN ADDRESS
D47     PRINTER LISTEN ADDRESS
```

d. Stop and reset the clock. Restart the clock.

```
C40     DATA MODE
D120    STOP CLOCK
D122    RESET CLOCK
D124    START CLOCK
```

e. Set the clock to talk.

```
C60     COMMAND MODE
D105    CLOCK TALK ADDRESS
C40     DATA MODE
```

f. Untalk the clock.

```
C60     COMMAND MODE
D137    UNTALK ADDRESS
```

g. Set the clock to talk.

```
D105    CLOCK TALK ADDRESS
C40     DATA MODE
```

h. Clear the interface.

```
C1      IFC
```

3) How does the HP-IB driver know in what format to send data to a device? To receive data from a device?

Simple. Each device has a configuration word stored in its EQT entry which indicates whether the device is buffered, whether it uses DMA, what its EOT format is, whether it can interrupt the controller to request service, etc. Most devices can use the default setting of the configuration word but some may require a different configuration. Each device reference manual should be consulted to determine if the default setting is appropriate for the device.

4) Convince yourself of the validity of the preliminary system controller interface commands on pg. 9-15 using the diagnostic to communicate to the bus.

a. Clear the interface

b. Put bus in command mode

c. Send timing generator listen address

d. Put bus in data mode and observe timing generator front panel.

```
C1      IFC
C60     COMMAND MODE
D46     TIMING GENERATOR LISTEN ADDRESS
C40     DATA MODE
```

What happened?

The timing generator panel lights should indicate that the generator is addressed to listen but not remote enabled.

e. Clear the interface.


```
C1      IFC
```

What happened? What does IFC do?

The timing generator panel lights now indicate that the generator is no longer addressed to listen. IFC untalks, unlistens, and unaddresses all addressable devices.

f. Remote enable bus and observe bus analyzer front panel.

g. Put bus in command mode.

h. Send timing generator listen address and observe front panel.

i. Send DVM listen address and observe DVM front panel.

```
C3      REMOTE ENABLE (REN)
C60     COMMAND MODE
D46     TIMING GENERATOR LISTEN ADDRESS
D43     DVM LISTEN ADDRESS
C40     DATA MODE
```

Did the expected happen? Of course it did!!

Remote enable causes the REN light to light on the bus analyzer front panel. The combination of REN, ATN and device listen addresses causes the panel lights on the generator and DVM to indicate that both devices are addressed to listen and remote enabled.

j. Observe timing generator, DVM, and bus analyzer front panels while sending GTL (go to local).

```
C2      GO TO LOCAL
```

Go to local or "not" REN causes the remote indicator lights on the DVM, bus analyzer, and generator to be extinguished. The devices are no longer able to be programmed remotely.

k. Repeat f., g., h., i. programming sequences

l. Clear the interface.

What is the difference between GTL or "not" REN and IFC?

GTL only causes devices to return to their local programming modes making remote programming impossible. IFC, however, untalks, unlistens and unaddresses all addressable devices. It does not change the remote or local status of the devices though.