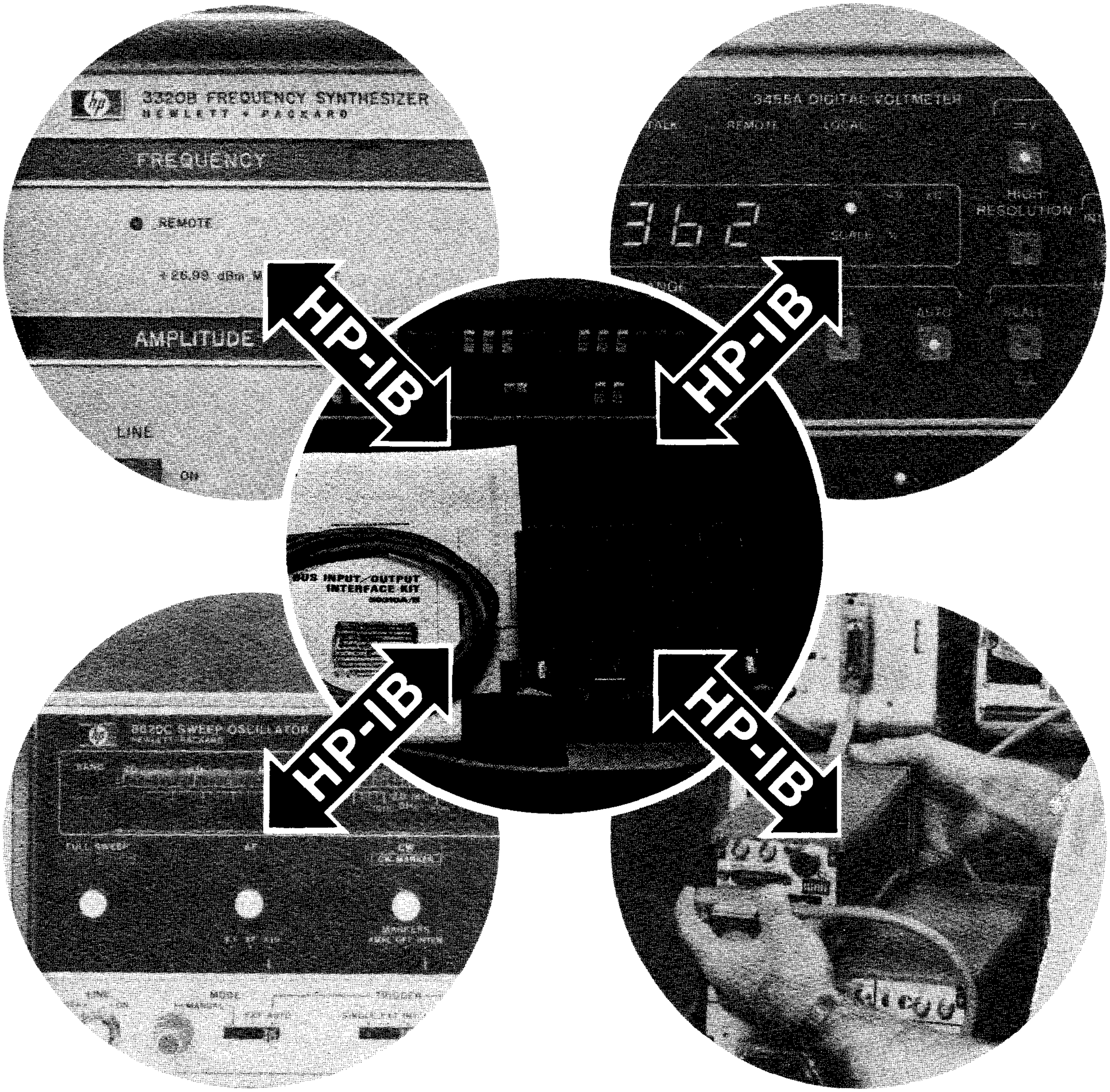
HP-IB

HP 1000/HP-IB Programming Procedures

HEWLETT PACKARD

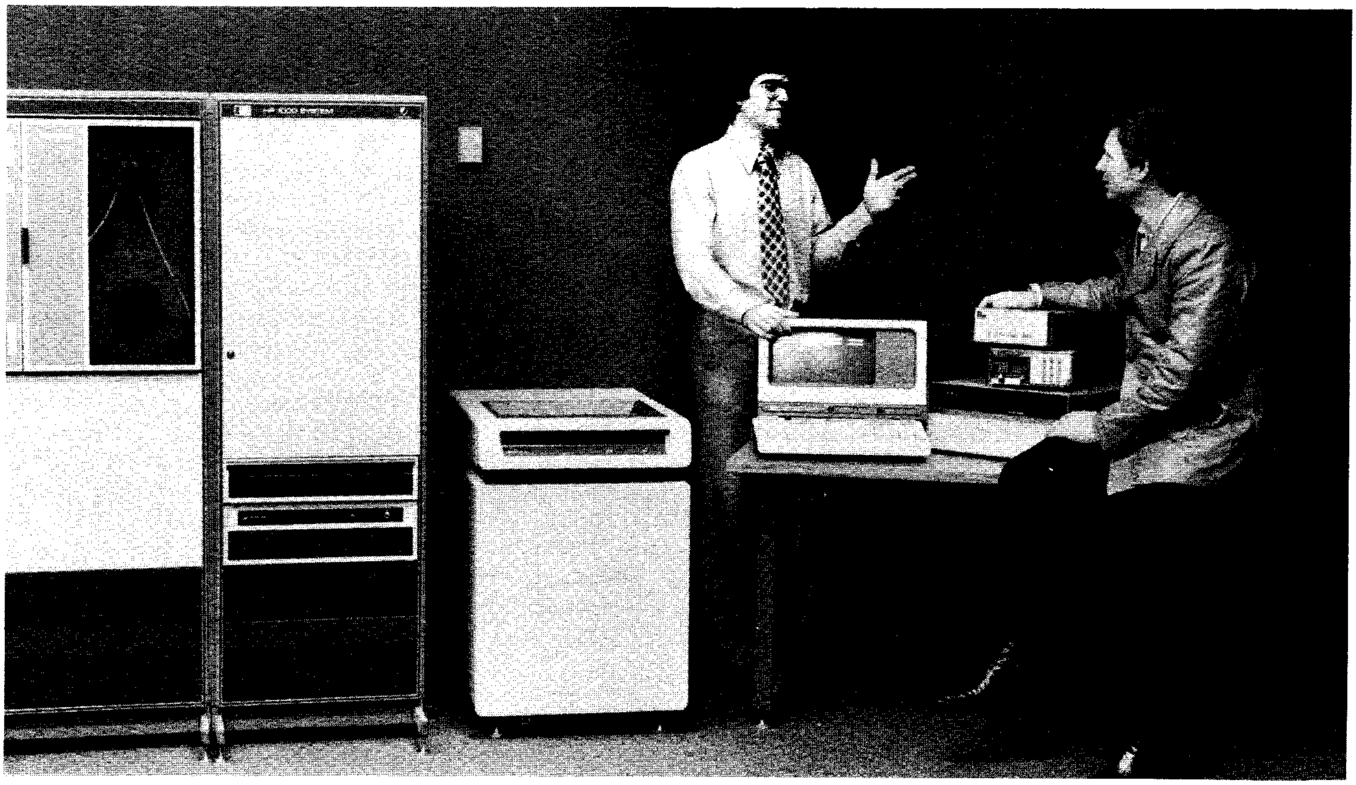
Application Note 401-1



The AN 401 Series

This Application Note 401-1 is supplemented by detailed instrument-specified programming guides. You can select the modules pertinent to your system and obtain them from your local HP sales representative. You can also obtain copies by writing to Hewlett-Packard Company, Sales Literature Center, 1820 Embarcardero Rd., Palo Alto, CA 94303.

Application Note	Content	Document Number
401-2	59307 VHF Switch/HP 1000 Computer	5953-2801
401-3	5345A Counter/HP 1000 Computer	5953-2802
401-4	5342A Microwave Counter/HP 1000 Computer	5953-2803
401-5	5328A Counter/HP 1000 Computer	5953-2804
401-6	3438A Digital Multimeter/HP 1000 Computer	5953-2805
401-7	3455A Digital Multimeter/HP 1000 Computer	5953-2806
401-8	59309A Digital Clock/HP 1000 Computer	5953-2807
401-9	6002A Power Supply/HP 1000 Computer	5953-2808
401-10	3437A Digital Voltmeter/HP 1000 Computer	5953-2809
401-11	3495A Scanner/HP 1000 Computer	5953-2810
401-12	3582A Spectrum Analyzer/HP 1000 Computer	5953-2811
401-13	3325A Function Generator/HP 1000 Computer	5953-2812
401-14	4262A Digital LCR Meter/HP 1000 Computer	5953-2813
401-15	8672A Synthesized Signal Generator/HP 1000 Computer	5953-2814
401-16	436A Microwave Power Meter/HP 1000 Computer	5953-2815
401-17	8620A Sweep Oscillator/HP 1000 Computer	5953-2816
401-18	59306A Relay Actuator/HP 1000 Computer	5953-2817
401-19	8660C Synthesized Signal Generator	5953-2818
401-20	9871A Character Impact Printer	5953-2819
401-21	6942A Multiprogrammer II	5953-2820



Welcome to the world of HP-IB. We hope you'll find that the interface bus is the most flexible, simple, standardized interface on the market today. This application note is intended to stimulate new HP-IB ideas in a typical HP-IB environment.

Table of Contents

Page

Introduction 1

Chapter 1. Getting Started 2
Adopting a Procedure 2
Device Introduction 2
Addressing 2
  Multiple Addresses 6
System Preparations 6
Programming 6
  SRQ Processing 7
Performance 7
  Session Monitor Users 7
  Bus Status and Configuration Utility Program 8
Summary 8

Chapter 2. System Preparations 9
System Considerations 9
HP-IB Device Considerations 9
System Preparations Using File Manager 10
Setting the Device to Remote 11
Configuring the Device 11
SRQ Priority Processing in DVR37 12
Clearing the Device 12
Standard Procedure Summary 13

Chapter 3. The Bus Status and Configuration Utility 14

Chapter 4. HP-IB Performance Measurements 24
Describing I/O Performance Characteristics 24
I/O Operations 26
Processing Very Fast Input 28
CPU Free Time (TF) 28
Linear Approximations 29
Obtaining Measurement Rate (RT and %UTL) 30

Chapter 5. HP-IB Performance Programs 33
Program Descriptions 33
Obtaining Operating System Overhead 34
  Program PERF 43
  System Overhead Procedure 45
Program Loading Procedures 47

Chapter 6. Assigning HP-IB Logical Unit Numbers 48
Subroutine INPRM 48
Subprogram GTDLU 51
Function GTADD 57
Function RCORD 58
Function AVAIL 58
Function ASNLU 58
Function AVEQT 60
Function GIBLU 61
Function FLASS 61
Subroutine CNVRT 63
Block Data Subprogram 63
Function ERROR 64
Program AUTLU 64
Automatic LU Assignment when Using the Session Monitor 66
Transfer File *AUTOP 66
Program GTSLU 66
SRQ Priority Processing in DVR37 68

Introduction

Generally, HP 1000 software is modular. The many different options allowed for HP-IB and other I/O subsystems provide much freedom and user program flexibility, but difficulties can arise unless procedural conventions are adopted which provide an organizational structure to this mass of paraphenalia.

Distributing intelligence to peripheral devices within the system environment has taken its toll on documentation. Two manuals are now needed where one used to be adequate. Unfortunately, the system manual proports to solve the problem from one end of the spectrum whereas the device (or instrument) manual works from the opposite end. Solving the problem requires an implementation of the products which is application-specific and user-oriented, which neither manual seems to address.

The fact remains that a basic procedural method can be adopted for the HP-IB controller and its devices. This method can make the marriage of the two, a simple one.

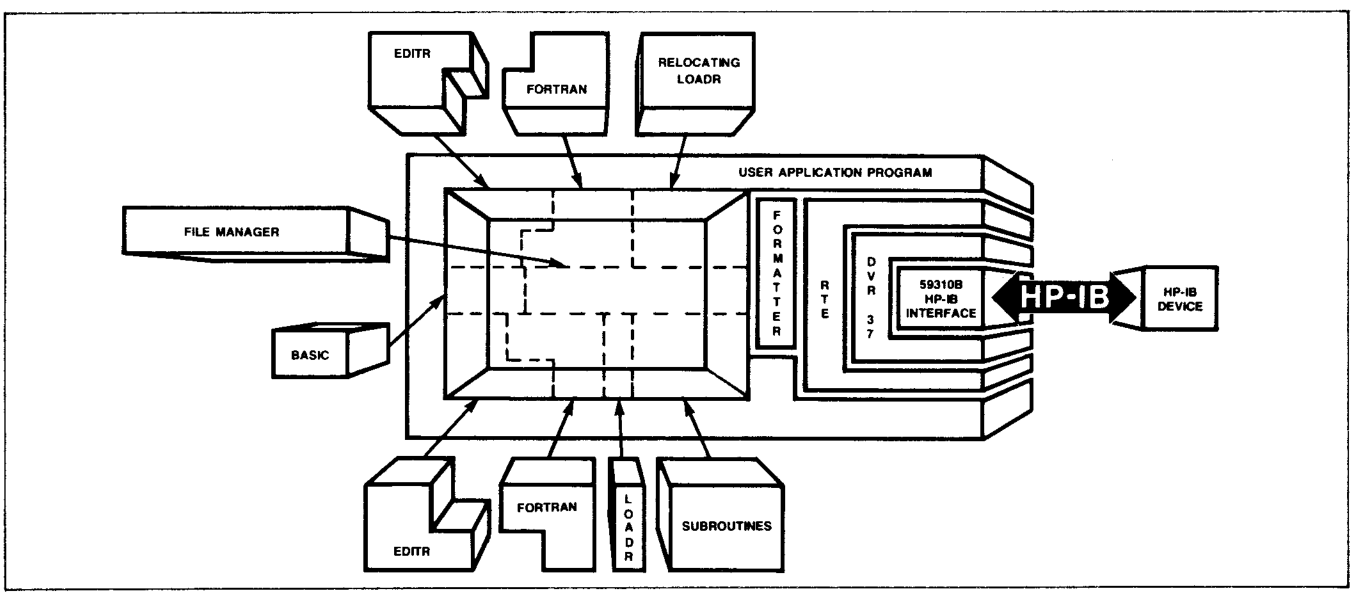
This application note has been designed to assist you in bringing up an HP-IB/HP 1000 in an effective, procedural manner. At the same time, it introduces some special capabilities of the controller and the devices which can help optimize the solution to your problem.

The basic methods have been outlined in a checklist fashion. The checklist is ideal in such a compartmentalized environment. Chapter 1 outlines the basic steps for setting up an HP-IB system from an application standpoint. In this chapter, the basic HP-IB checklist is introduced in preparation for the remaining chapters. Once the checklist has been completely defined, an abbreviated version will be seen over and over again in the remaining chapters.

Besides the HP-IB checklist, application programs and utilities have been written to clarify concepts introduced, provide programming hints, or minimize the task of reproducing identical software again and again. These progams and subroutines are available from the contributed library,[1] documented fully in this application note, or are HP products.

The procedures, application programs, and utilities encompass the whole RTE operating system and its accessories. Some of the procedures, for example, use Batch Spool Monitor commands and transfer files from the RTE File Manager program. On the other hand, all of the configuration procedures may be conducted within the user program each time it is executed. Only the user can decide which methods are optimal in his application situation. In any case, system integrity must be maintained by setting up user conventions.

Finally, the bus is a dynamically changing world. New devices and instruments are produced by HP and other companies each day. This application note should provide convicing evidence that setting a specific procedure — and sticking to it — is your assurance of success with HP-IB. It is also evidence that when you purchase an HP-IB instrument, you are investing in and receiving much more than just IEEE-488 compatability.



HP 1000 Software Compartmentalization

[1] *See the "HP 1000 Communicator" for more information about obtaining these programs.*

Getting Started

Chapter 1

Adopting a Procedure

We are constantly looking for new ways to quickly assemble and program HP-IB instrumentation. Usually, the most simple approach to making a new instrument functional on the HP-IB is the most effective one. The trick is to adopt a detailed procedure and make a checklist which can be used for most devices, then follow it carefully.[1] The procedures to be used for different instruments are basically the same, and only the application-specific areas change. We have found that the outline in figure 1-1 works very well.

Each chapter in this application note will cover the points listed in the outline. In many cases a chapter will also elaborate upon HP-IB instrument strengths which can be enhanced by the HP 1000 in application-specific areas.

The remainder of this chapter expands on the outline and discusses generalities among HP-IB instruments. Some special points concerning HP 1000 software are also mentioned. Figure 1-2 contains a short glossary of terms, abbreviations, and mnemonics, used in the following chapters.

This series of application notes assumes the user is operating outside the session monitor environment. All references to logical unit numbers refer to system logical unit numbers. In a session environment, the File Manager commands shown here would require that a user have a capability level as high as 60. More information concerning session monitor use is shown at the end of this chapter.

Device Introduction

Gather the equipment. The instruments should be turned off; however, it isn't necessary to turn off the HP 1000.

Typically, the equipment in figure 1-3 comprises an HP-IB System.

Addressing

Every HP-IB instrument, peripheral, or device to be used on the interface bus must have an identifying address, which you assign. The address in an HP-IB system is just like the phone number in the telephone system; it is a unique means of access by which a message can be sent to or from a specific device.

---

[1]*The HP-IB User Manual, part no. 59310-90064, is referenced in this application note several times and should be used to successfully control instruments with the HP 1000.*

Most HP-IB devices have small address switches on the rear panel (figure 1-4). Each binary switch is simply set to the appropriate position, 1 or 0. The combination of binary address switches corresponds to a unique decimal number (1-31) which represents the device's address. If the device address was preset at the factory, it can be left as is; however, it may be overridden simply by setting the rear panel or internal switches to a different address.

It is considerably easier to set the instrument address switches before connecting the cables to the equipment. *Make sure that no two instruments on the same bus have the same address settings.* It's a good idea to use an address assignment table like the one introduced in Appendix B of the HP-IB Users Manual.[1] The Appendix also contains information concerning the HP 1000 bus interface card switches.

NOTE

> The computer I/O bus interface card is assigned a device address of zero in the software driver. As a result, this address should not be used by other instruments.

Next, physically interconnect the cables.

Appendix E in the "HP-IB Users Manual" (concerning logical unit numbers and subchannels) should be understood before continuing in this chapter.[1]

In this text, we will discuss the instrument address in base eight (octal). This address (set into the instrument switches) satisfies the need for two addresses (the talk and the listen instrument address).

Some controllers handle device addresses differently from others. The 9830A desktop computer, for example, forces the user to differentiate between talking to a device or listening to the device by sending a corresponding talk address or listen address each time a request is made.

More sophisticated controllers like the HP 1000 perform TALK/LISTEN addressing automatically by interpreting "READ" and "WRITE" statements (in FORTRAN, for example).

Figure 1-5 demonstrates the correspondence between the switch setting of the HP-IB device and the octal address associated with the table of HP 1000 logical unit numbers. Simply set the switches on the instrument's rear panel, figure the octal value, and enter the LU command for the device assignment. (A "B" after the device address indicates the value entered is in octal.)

A. Device Introduction

- What is it?
- What does it do?
- Is it programmable?
- Does it return measurements?

B. Addressing

- Does it have a single address (both TALK and LISTEN)?
- Does it have multiple addresses?

C. System Preparations

1. Logical Unit Assignment
2. Buffering (Use or don't use?)
3. Time-out (Is time out a legal condition or an illegal condition?)
4. Device configuration (Should it be adjusted to something other than the default value, i.e., should DMA be allocated for the device? Should SRQ interrupts have a high priority? Does the device require special end of record processing?)
5. Remote (Is it necessary for programming?)

D. Programming

1. What communication mode does the device assume?
   a. Does it return measurements in ASCII?
   b. Does it return measurements in binary (in a packed type of format)?

D. Programming (Continued)

   c. Does the device lend itself to simplified programming (i.e., can it be programmed using simple FORTRAN or BASIC formatted "WRITE" or "READ" requests)?
   d. If the device returns measurements, can the computer request them in free field input (i.e., READ(5,*)A)? If not, what format should be used?
2. What measurement return capacity does the device allow (i.e., How many measurements can be returned in one request?)?
   a. Does the device return an end of record (EOR)?
   b. How many output bytes are sent from the device at one time (before an EOR terminator)?
3. Does the device have service request ability?
   a. Should processing be done using parallel or serial poll?
   b. Does it generate SRQ for error conditions only?
   c. Is SRQ generated in measurement situations?

E. Performance

1. What is the maximum number of readings at one time (between EORs)?
2. Binary input allowed?
3. Best case performance (criteria is time).
4. Worst case performance (criteria is time) (usually optimizing for simplicity).

Figure 1-1. A Systematic Approach to Analyzing HP-IB Instruments

LU	logical unit
EQT	equipment table
ILU	mnemonic for the user input LU
ILST	mnemonic for the list LU (line printer,etc.)
IDLU	mnemonic for the device LU
IBLU	mnemonic for the bus LU
INPRM	function subprogram which obtains ILU, ILST, IDLU (see Chapter 6)
SRQ	HP-IB service request
SRDIJOPE	bits used in the device configuration word.

Figure 1-2. Glossary

- HP 1000 and 59310B Bus I/O Interface Kit
- Bus Interface Cables — 0.5, 1, 2, or 4 meters in length.
- Bus Compatible Instruments and Devices (up to 14) — Each device must include all options and accessories necessary for HP-IB operation.

Figure 1-3. Needed HP-IB Equipment

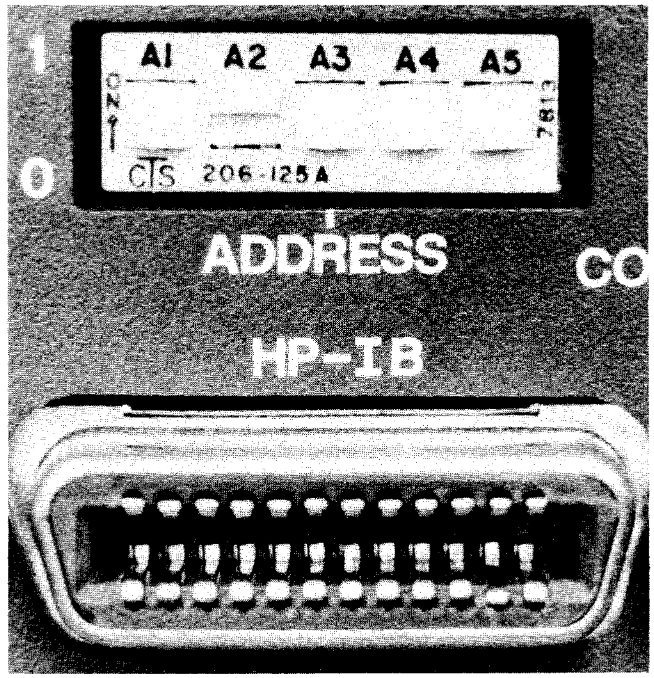


Figure 1-4. A Typical Set of Address Switches

Address Switch Numbers					Address Characters		Octal Value	Decimal Value
5	4	3	2	1	Talk	Listen		
0	0	0	0	0	@	SP	00	00
0	0	0	0	1	A	!	01	01
0	0	0	1	0	B	"	02	02
0	0	0	1	1	C	#	03	03
0	0	1	0	0	D	$	04	04
0	0	1	0	1	E	%	05	05
0	0	1	1	0	F	&	06	06
0	0	1	1	1	G	'	07	07
0	1	0	0	0	H	(	10	08
0	1	0	0	1	I	)	11	09
0	1	0	1	0	J	*	12	10
0	1	0	1	1	K	+	13	11
0	1	1	0	0	L	,	14	12
0	1	1	0	1	M	–	15	13
0	1	1	1	0	N	.	16	14
0	1	1	1	1	O	/	17	15
1	0	0	0	0	P	0	20	16
1	0	0	0	1	Q	1	21	17
1	0	0	1	0	R	2	22	18
1	0	0	1	1	S	3	23	19
1	0	1	0	0	T	4	24	20
1	0	1	0	1	U	5	25	21
1	0	1	1	0	V	6	26	22
1	0	1	1	1	W	7	27	23
1	1	0	0	0	X	8	30	24
1	1	0	0	1	Y	9	31	25
1	1	0	1	0	Z	:	32	26
1	1	0	1	1	[	;	33	27
1	1	1	0	0	\	<	34	28
1	1	1	0	1	]	=	35	29
1	1	1	1	0	^	}	36	30
1	1	1	1	1	_	?	37	31

Figure 1-5. Switch setting, device address correspondence

Assuming logical unit number 15 is available, equipment table 10 is the HP-IB bus in use, and the device address switches are set to 17 octal, an example logical unit assignment statement would appear as:

```
:SYLU,15,10,17B
```

Multiple Addresses

HP-IB devices that communicate with each other as well as with the computing controller may have two talk addresses or two listen addresses.

Multiple-address devices have a different set of address switches on the rear panel — sometimes just four switches. A single setting will determine two talk addresses and two listen addresses. The four switches control the A2 through A5 positions. For example, setting switches A2 through A5 as shown in Figure 1-6 will result in two listen addresses of "2" and "3" along with two corresponding talk addresses of "R" and "S". (See Figure 1-5.)

A5	A4	A3	A2	(Notice no A1, therefore, switches
1	0	0	1	are set to octal 22 or 23)

Figure 1-6. Example Setting with Four Address Switches

System Preparations

System preparations typically involve operations which take place only when the system is restarted. For this reason, they can sometimes be accomplished in a separate program rather than incurring this overhead each time the user program is executed. (Figure 1-7 shows how these operations may be accomplished in File Manager.)

:SYLU,40,10,0	Assign general bus LU
:SYLU,41,10,17B	Assign device LU
:SYLU,42,11,0	Assign general bus LU
:SYLU,43,11,23B	Assign device LU
:CN,40,16B	Set bus into remote
:CN,42,16B	Set bus into remote
:CN,41,25B,37000B	Configure device for DMA operation
:CN,43,25B,17400B	Errors to be handled by user program
:LL,43	Set list device
:AN,I2E8I1	Clear device
:LL,0G	Reset list device

Figure 1-7. Typical WELCOM File Procedure

Because the proper system preparations are tantamount for successful HP-IB programming, all of Chapter 2 is concerned with this subject.

Additional Comments

Some of the HP 1000/HP-IB procedures use Batch Spool Monitor commands and the transfer file capabilities from the RTE File Manager. These File Manager commands are used to configure devices and perform general I/O functions, which typically only need to be performed once during system initialization.

All of the HP-IB configuration requests, buffering, and time-out requests may be executed dynamically, on-line, using simple system commands from a user terminal (or transfer file).[2] In this situation, user programs must assume that the initialization procedure was conducted smoothly without abnormal termination, and that the HP-IB hardware and software works correctly.

On the other hand, all of the configuration procedures may be conducted within the user program each time the program is executed. Message subroutines may be applied as documented in the HP-IB User's Manual.

Several facets of error-checking are available to the programmer. Depending on how the HP-IB subsystem is configured, error-checking will be performed by the operating system or the user program. When the default condition (system error-checking) is used, the user program is either terminated abnormally or it is suspended until an operator intervenes at a terminal. User program error-checking facilitates "soft errors," but again, conventions must be adopted among user programs to maintain uniformity.

Programming

Instrument programming is usually accomplished in the user BASIC, FORTRAN, or Assembly program although it may sometimes be performed using system software such as File Manager for checking out HP-IB instruments. The other 401 Series application notes describe specific HP instruments, their applications, and programming instructions in detail.

---

[2]*Discretion must be used when exercising commands which change buffering, time-outs, and device status, etc.*

SRQ Processing

Some HP-IB instruments have the ability to spontaneously communicate back to the controller. There are some areas where this capability can be used in conjunction with the advanced SRQ handling software of the HP 1000 for measurement and error applications.[1] These are more sophisticated application situations which require some basic understanding of HP-IB device service requests (SRQ).

The structure of HP-IB is such that when an instrument spontaneously generates a service request, the controller may process it in one of two ways:

1. A parallel poll may be initiated. A device-independent command is sent on the bus, and all configured instruments[3] respond at once with one bit of information. From this bit the HP 1000 determines which instrument generated the SRQ. Afterwards, the HP 1000 may request more status by conducting a serial poll on that instrument only, or by interrogating the instrument using data messages.

2. A serial poll may be initiated. The HP 1000 sequentially polls each known instrument[4] and determines which one generated the SRQ. Included with this information is a complete device status byte which can help determine the status of the instrument without further interrogation.

The user decides ahead of time which devices are to be polled, and he must adhere to this polling sequence during an SRQ occurrence. If an HP-IB device is "known" to the HP 1000 controller but the cable is removed from it prior to a service request (caused by another HP-IB device), unpredictable results may occur. SRQ, once set up, is a device-dependent function. Even though an SRQ arrives from only one HP-IB device, all configured devices will be polled; and if a device is not physically present, the bus will "hang" or time-out before the serial poll sequence completes.

Similiarly, HP-IB devices should not be allowed to generate SRQs before SRQ configuration. When this happens, the message "illegal interrupt" is printed on the error log terminal. This only occurs, however, when other HP-IB instruments are already configured for SRQ processing .

---

[3]*A configured instrument is one which has previously been programmed to respond with a bit of status on the parallel poll command.*

[4]*Known instruments are ones which are specifically configured to schedule SRQ programs when the computer determines that an SRQ has been sent by that instrument. This configuration takes place before any SRQ occurrences.*

Each of the instrument-specific 401 application notes will discuss the ramifications of SRQ processing (when applicable) but the above basic ideas should be understood for successful operation. SRQ priority operation in Chapter 2 should also be read for SRQ operation.

Performance

Like instrument programming, performance is heavily device-dependent. Most instrument application notes in the 401 Series contain information on device performance, usually one or more graphs depicting typical performance with an HP 1000 system under different conditions. In addition, Chapter 4 of this note describes HP-IB performance theory in the HP 1000. Chapter 5 discusses the performance application programs used to obtain the performance graphs and how a user can do his own performance measurements on-line.

Session Monitor Users

The File Manager commands described in the 401 series of application notes require capability levels shown in Figure 1-8. Three commands, "SYLU", "SYEQ" and "SYTO" each require a capability level as high as 60. A typical session user may execute these commands in one of three ways:

1. Obtain a user capability level of 60 from the system manager.

2. Configure a terminal which operates outside the session environment.

3. Request that the system manager create a special File Manager transfer file on LU 2 or LU 3 and route all high capability requests through this transfer file.

In case #3 above, a File manager transfer file can be created which contains only four lines.

```
:SV,1,9,IH
:1G,2G,3G,4G
:SV,9G
:
```

COMMAND	OBSERVATION CAPABILITY	MODIFICATION CAPABILITY
:CN	NA	20
:AN	NA	20
:DU	NA	20
:TR	NA	01
:SYLU	60	60
:SYEQ	10	60

Figure 1-8. File Manager Commands and Their Capability Levels

By transferring to this transfer file on LU 2 or LU 3 and supplying the correct information, high capability commands may be accomplished with a user capability as low as 1. Suppose, for instance, that the above transfer file has the namr "*:-1:-2". The procedure file previously shown in figure 1-7 could be modified as shown in Figure 1-9. Note that non-session transfer files can easily be changed using the unconditional exchange command in the RTE EDITR. System integrety is retained by using a read and write protect security code on the file.

```
:*:-1:-2,SYLU,40,10,0
:*:-1:-2,SYLU,41,10,17B
:*:-1:-2,SYLU,42,11,0
:*:-1:-2,SYLU,43,11,23B
:*:-1:-2,CN,40,16B
:*:-1:-2,CN,42,16B
:*:-1:-2,CN,41,25B,37000B
*:-1:-2,CN,43,25B,17400B
:*:-1:-2,LL,43
:*:-1:-2,AN,I2E8I1
:*:-1:-2,LL,0G
```

Route all high capability commands through a generalized transfer file on LU 2.

Figure 1-9. Modified Session sequence from Figure 1-7

The Bus Status and Configuration Utility Program

The purpose of the utility[5] is to provide the user with dynamic, complete, up-to-date, and readable information about the HP-IB on his HP 1000. It provides information about each HP-IB logical unit, including configuration, status, and SRQ scheduling. Complete equipment table information, time-out values, buffering, and availability are also output.

Summary

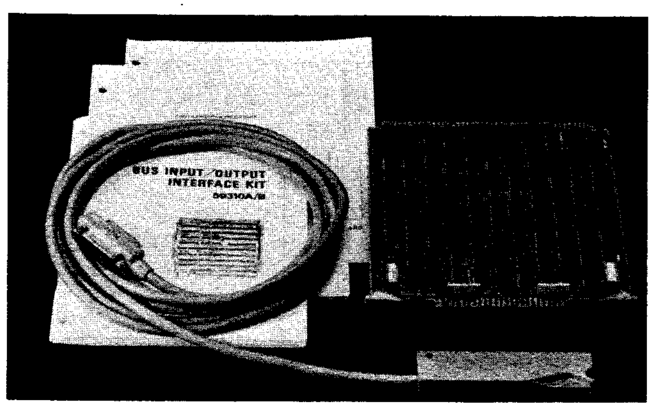
The outline and the procedural information discussed in Chapters 1 and 2 show how a new HP 1000 instrument user can get up and running with several of HP's most popular HP-IB instruments. Each of the other 401 Series application notes contains information which is pertinent to a separate HP device.

While the examples have been developed on an HP 1000 system using the RTE-IV Operating System, they apply equally well to other operating systems including RTE-M1, RTE-M2, RTE-M3, RTE-II and RTE-III. Also, any HP 1000 which has an RTE File Manager program may use the File Manager examples for operating HP-IB instruments.

The equipment used in the AN 401-1 Series consists of the 59310B Interface Kit with I/O software (which includes the general purpose HP-IB driver, DVR37 with SRQ capability).

HP-IB performance measurements were taken (except where noted) using the high-speed 350-nanosecond memory, in an F-Series HP 1000 computer. Some of the measurements perform standard FORTRAN input and output utilizing the FORTRAN Formatter.

If the reader wishes to duplicate the performance tests given, Chapters 4 and 5 will be helpful. Chapter 4 outlines the theory behind the HP-IB performance measurements and standardizes some of the definitions. Chapter 5 documents the programs, and supplies program listings needed to duplicate the measurements made in this application note. These programs may also be used to measure performance of other HP-IB devices.



[5]*The BSCU is described in Chapter 3 of this application note.*

System Preparations

Chapter 2

Because the HP-IB standard does not provide a definition for instrument operating characteristics, standard conventions should be developed and followed by HP-IB software programmers. The HP-IB system software (in the HP 1000) gives the programmer sufficient flexibility in this area.

The HP-IB software driver lets the programmer configure each HP-IB instrument so that it can operate effectively. Device specific characteristics such as end of record (EOR) processing, transfer media (DMA or Non-DMA), device status (on a serial poll), and service request handling (SRQ response) are individually configurable.

At system boot up, the driver software defaults these device characteristics to those most likely assumed by HP-IB devices, but the configuration for each device should be evaluated for proper operation. In summary, two types of configurations should take place before instrument programming is begun:

a. The HP-IB system software must be configured for each bus device.

b. The bus should be configured for remote operation (one of the basic HP-IB functions) so that the HP-IB instruments may be controlled by the computer.

System Considerations

At the computer operating system level, the following list of questions should be evaluated when interconnecting a new device to the bus.

1. Which HP-IB equipment table will be used?

2. What I/O characteristics were given to the bus at system generation:

   a. Does the bus default to buffered mode or unbuffered mode?
   b. Was DMA assigned to this bus during system generation?
   c. What is the current time-out setting?

3. Which logical unit numbers are available?

4. Have any of the above been changed since generation of the system by on-line users? Will any of these characteristics change during HP-IB operation?

HP-IB Device Considerations

The HP-IB device manual should be referenced to answer the following:

1. Should DMA be allocated for the device? (performance consideration)

2. Will the device pause for long periods of time during I/O communication (time out should be adjusted accordingly)? (error condition)

3. Does the device send an end-of-record terminator (EOR)? Will the HP-1000 default values for end-of-record recognition (one value for each device) work or should the HP-IB driver software be reconfigured for a special sequence? (performance and error conditions)

4. Should device errors cause RTE to abort a user program, or should the user program be allowed to take action on errors? (user program options for error checking)

5. Should the occurrence of an SRQ be allowed to abort any current I/O operation on the bus? (high priority SRQ processing)

These questions will arise each time a new device is added to the bus. Attention must be given to them or HP-IB interconnections can become a tedious process.[1] Answers to some of the above questions may be obtained by using system requests like:

:SYTO	time-out
:SYST	status
:SYLU	LU status
:SYEQ	EQT status

from the RTE File Manager.[2]

---

[1]*The current state of the HP-IB can be obtained by executing the Bus Status and Configuration Utility Program described in Chapter 3. The program printout answers the questions in both sections above and also some others which may be asked when using the SRQ capabilities of HP-IB.*

[2]*These requests are documented in the Batch Spool Monitor Manual, part number 92060-90013.*

Chapter 2

System Preparations Using File Manager

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 in figure 2-1):

Table 2-1. 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 device³
:DU --- Send ASCII message from source to
        destination
```

Figure 2-1. File Manager commands for HP-IB

Table 2-1 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.*

The device configuration word, as mentioned earlier, usually defaults to a predetermined setting (17000B or 37000B) but sometimes modifications are required. For example, to allocate DMA for a device, the configuration word would be changed from 17000B to 37000B (set bit 14 on). If error checking from the user program is desired, the word would look like 17400B (set bit 9 on); for DMA and user program error checking, the word looks like 37400B, etc. The format of the configuration can be visualized by executing the Bus Status and Configuration Utility and specifying the device LU. To set the device configuration word for DMA from File Manager, use the control request:

```
:CN,IDLU,25B,37000B
```

Study thoroughly the section "Controller Configuration" in the HP-IB User's Manual.[6] This section describes the format of the device configuration word, and the details for end of record processing.

SRQ Priority Processing in DVR37

Discussion of the S bit in the device configuration word is accurate on page 2-23 of the Programming and Operating Manual for DVR37 (59310-90063). However, the real implications of high priority SRQ processing are not discussed.

When multiple instruments reside on the same bus, the "current I/O request" may be any of these devices. Therefore, high priority SRQ processing affects all devices currently configured on HP-IB.

When the S bit for a device (device A) is set to zero, it indicates the following:

If an SRQ arrives while I/O processing is in effect for **this** device (device A), the SRQ will be held off until this I/O process is complete. This, however, may not always satisfy the user's needs. Therefore when the S bit is set to one (for device A), the I/O request will be aborted so that the SRQ can be evaluated. Note that where the SRQ came from (device A, B, C, etc,) is unclear at this point. Setting the S bit for the device that generated the SRQ is no assurance that the I/O request will be aborted.

To insure that a high priority SRQ receives immediate processing, the S bit must be set for every device configured on the bus. (Configuring the S bit to 1 for the device generating the SRQ can only be decided by the user.)

High priority SRQ processing can have grave effects on HP-IB I/O programs in the system. Very few HP-IB instruments are capable of recovering from an aborted I/O request, let alone the re-issuance of the request later. Use high priority SRQ processing only when the SRQ must be processed immediately, at all costs.

Clearing the Device

Some devices need to be initialized before instrument programming can take place. This can occur in one of two ways: If the instrument recognizes the selected device clear command (SDC) described in the HP-IB User's Manual, then the file manager command,

```
:CN,IDLU,0
```

may be used. However, if the device accepts commands in ASCII, but it does not recognize the SDC command, there usually is a device dependent ASCII string which may be sent to clear the device, for example (figure 2-2).

Figure 2-3 shows an exerpt from a WELCOM file to configure two HP-IB's on system boot-up.

---

[6] *The HP-IB User Manual, part no. 59310-90064.*

Standard Procedure Summary

1. Load and execute the Bus Status and Configuration Utility in Chapter 3, or use system requests to obtain answers to the two lists in figures 2-2 and 2-3.

2. For all new instruments on HP-IB, set the EQT to the unbuffered mode (until the instrument has been checked out and is understood).

3. Remember that there is one time-out for each bus, not for each device. The time-out must be a compromise for all the devices on a bus and not set specifically for one device.

4. Modify the WELCOM file, write a user program, or write a transfer file to perform the above; assign the proper LU's, remote enable the bus, and configure each device.

```
:LL,43 -------- Set HP-IB list device
:AN,I2E8I1 ---- Send reset command
:LL,0G -------- Reset list device to input terminal
```

Figure 2-2. Example File Manager sequence

```
.
.
.
:SYLU,40,10,0. . . . . . . . .Assign general bus LU
:SYLU,41,10,17B. . . . . . . .Assign device LU
:SYLU,42,11,0. . . . . . . . .Assign general bus LU
:SYLU,43,11,23B. . . . . . . .Assign device LU
:CN,40,16B . . . . . . . . . .Set bus into remote
:CN,42,16B . . . . . . . . . .Set bus into remote
:CN,41,25B,37000B. . . . . . .Configure device for DMA operation
:CN,43,25B,17400B. . . . . . .Errors to be handled by user program
:LL,43 . . . . . . . . . . . .Set list device
:AN,I2E8I1 . . . . . . . . . .Clear device
:LL,0G . . . . . . . . . . . .Reset list device
.
.
.
```

Figure 2-3. WELCOM file example

The Bus Status and Configuration Utility[1]

Chapter 3

The interface card described in this application note is the 59310B HP-IB card, but it looks just the same as any other hardware interface to the Real Time Executive (RTE). The RTE builds a layer of software and a pair of tables around each physical I/O card (and in some cases a pair of I/O cards) in an HP 1000 system. The layer of software is called the driver, and the two tables are the device reference table (DRT) and the equipment table (EQT).

In the early days of RTE, the equipment table for each hardware interface had 15 entries; but as I/O devices progressed, extra entries (EQT extensions) were needed to support more sophisticated peripherals. In general, however, only one peripheral per I/O card was implemented.

HP-IB was one of the first concepts to allow multiple peripherals connected to one interface card. However, this complicated driver software and the EQT had to be extended significantly. Because virtually any type of peripheral may be connected, certain information must be recorded and kept for each device. Specifically, for the HP-IB in the HP 1000, 18 extra words are needed for the EQT plus 7 words for each device which will be used on a bus (see figure 3-1). A detailed version of the HP-IB EQT block diagram may be found in table 3-1 and table 3-2. These tables represent the format of the REV. 1940 EQT.

Remember that each bus has its own EQT and the formula,

#EXTNTS = 18 + 7n

where n is the number of devices to be attached, must be applied to each one. Each bus is completely independent of the other.

The device reference table is simply the table of logical unit numbers in the RTE system. The DRT and EQT work together, controlled by the operating system to provide the mapping scheme shown in figure 3-2. The map is,

LU		EQT		DEVICE
n	→	1	→	n

where n logical units are mapped through one EQT to n device addresses. The map is dynamic. Changes are easily managed from an operator terminal or user program.

```
:SYLU,10,11,23B
:SYLU,15,11,3B
:SYLU,20,11,13B
```

are examples which dynamically reassign LUs and device relationships.

Because these changes are so easily managed, it is often helpful to observe the current configuration of HP-IB at any time. The Bus Status and Configuration Utility (BSCU) supplies this information.

Some of the status information is common to all 'n' devices associated with the EQT. The statement,

```
:RU,BS,,11
```

supplies this general information for EQT 11 as shown in figure 3-3.

The select code (for EQ 11) described is the actual slot location of the 59310B card inside the computer. The bus described in figure 3-3 is currently available for use, although it could have been down, busy, or waiting for a DMA channel.

One time-out value is used for all of the HP-IB devices on the bus. This time-out is set for 10 seconds.

The BSCU distinguishes between the "bus logical unit" and "device logical units". The bus logical unit has a device address (subchannel) of 0. Device logical units have non-zero addresses less than 32. The BSCU obtains the maximum number of device logical units which may be used on the bus by examining the number of EQT extensions which were allocated at system generation. The IEEE-488 standard indicates that the maximum number of device logical units allowed is 14. However, the BSCU will indicate a larger value if more EQT extension area was allocated at system generation.

The BSCU next indicates the number of five-word blocks currently available for new device logical units.

NOTE

> The BSCU assumes that if a matching LU cannot be found for a currently active five-word extension entry, the LU was inadvertantly reassigned. The BSCU will then automatically unconfigure the five-word entry and report the cleanup to the BSCU input terminal. This procedure can be suppressed by specifying a non-zero value in parameter four of the run statement.

---

[1] *The BSCU can be obtained from the Contributed Library (part number, 22682-13397). See the Computer Systems "Communicator 1000" for an order form.*

"Available logical units" indicates all of the LUs currently unassigned in the RTE system. These LUs are available for on line HP-IB assignment. "HP-IB logical units" indicate both the "bus" logical unit and "device" logical units currently assigned to EQT 11.

The statement,

```
:RU,BS,,,16
```

schedules the BSCU to supply user information about a particular HP-IB system logical unit. As shown in figure 3-4, it can be seen that LU 16 is a bus logical unit because it has a device address of zero.

```
*******************************************************
*                                                     *
*                                                     *
*           The standard 15 entry EQT                 *
*                                                     *
*                                                     *
*                                                     *
*                                                     *
*                                                     *
*                                                     *
*                                                     *
*                                                     *
*                                                     *
*                                                     *
*                                                     *
*                                                     *
*******************************************************
                          :
*******************************************************
*                                                     *
*           Plus 18 added entries                     *
*                                                     *
*                                                     *
*                                                     *
*                                                     *
*                                                     *
*******************************************************
                          :
*******************************************************
*                                                     *
*          Seven entries for device 1                 *   LU(a)
*                                                     *
*******************************************************
                          :
*******************************************************
*                                                     *     .
*          Seven entries for device 2                 *     .
*                                                     *     .
*******************************************************
                          :
*******************************************************
*                                                     *
*          Seven entries for device n                 *   LU(n)
*                                                     *
*******************************************************
```

Figure 3-1. Extension of the Original EQT for HP-IB

Table 3-1. HP-IB EQT Format

The Standard EQT:

```
EQT 1    DEVICE SUSPEND LIST POINTER
EQT 2    DRIVER INITIATION SECTION ADDRESS
EQT 3    DRIVER COMPLETION SECTION ADDRESS
EQT 4    DRIVER I/O ASSIGNMENTS
EQT 5    DRIVER STATUS INFORMATION
EQT 6    CURRENT I/O REQUEST
EQT 7    DATA BUFFER ADDRESS/CONTROL PARM
EQT 8    DATA BUFFER LNG/CONTROL PARM
EQT 9    CONTROL BUFFER ADDRESS
EQT10    CONTROL BUFFER LNG
EQT11    DRIVER CONTROL WORD
EQT12    EQT ENTRY COUNT
EQT13    EQT EXTENSION ADDRESS
EQT14    DEVICE TIME OUT RESET VALUE
EQT15    DEVICE TIME OUT CLOCK
```

Fixed Extensions (18 Entries)

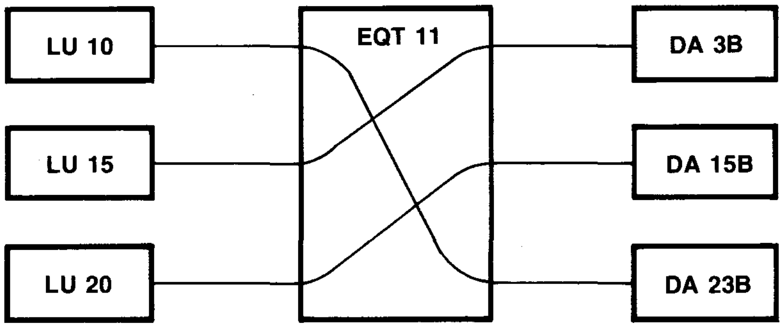
```
EQTX 1   CURRENT I/O CHARACTER LENGTH
EQTX 2   CURRENT I/O RUNNING BUFFER ADDRESS
EQTX 3   CURRENT I/O RUNNING CHAR COUNT
EQTX 4   CURRENT I/O TRANSMISSION LOG, CHARS
EQTX 5   PENDING STATUS BYTE
EQTX 6   SRQ SERVICE, PENDING BEQT ADDR
EQTX 7   SRG SERVICE, PENDING BEQT COUNT
EQTX 8   CURRENT OPERATION I/O RESUME ADDR
EQTX 9   DUMMY TIMEOUT = 0
EQTX10   AUTO ADDRESSING COMMAND BUFFER WORD 1
EQTX11   AUTO ADDRESSING COMMAND BUFFER WORD 2
EQTX12   BUS CONFIGURATION WORD (BEQT1 FOR BUS)
EQTX13-18   BEQT2-7 FOR SUBCHANNEL 0 (BUS ITSELF)
```

Variable length extensions (increments of 7 words)

```
BEQT 1  Device configuration word
BEQT 2  Alarm Program (first two characters)                      LU(a)
BEQT 3  Alarm Program (second two characters)
BEQT 4  Alarm Program (fifth character)
BEQT 5  SRQ Status Byte (lower byte)
BEQT 6  Arbitrary value to be passed to SRQ Program
BEQT 7  Error status of last operation or transmission log of last operation
BEQT 1             :
BEQT 2             :
BEQT 3             :                                              LU(n)
BEQT 4             :
BEQT 5             :
BEQT 6             :
BEQT 7             :
                   :
```

Table 3-2. HP-IB EQT Table Details

```
EQT4   - Format: D BPS TUU UUU CCC CCC
                 D = DMA ASSIGNED, 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 occurance, 1 = Yes
                 U = Unit or subchan, this request
                 C = I/O Channel, this req.

EQT5   - Format: A ATT TTT TSS SSS SSS
                 A = Availability
                 T = Device Type, 37
                 S = Status Byte

EQT6   - Format: C COZ OFF FFF 000 ORR
                 C = REQUEST TYPE, 0/1/2/3:STANDARD/BUFFERED/SYSTEM/CLASS
                 F = Subfunction
                 R = I/O Request, 1/2/3:READ/WRITE/CNTRL
                 Z = 0/1 SINGLE/DOUBLE BUFR REQUEST

EQT11  - Format: S A0E B00 HL0 00C MDI
                 S = SRQ Service in progress, 1 = Yes
                 A = I/O Request aborted to service SRQ, 1 = Yes
                 E = Expect/Issue EOR with I/O, 1 = Yes
                 B = Expect/Issue EOR with last data byte, 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 CRLF 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

EQT12 - Format: S PAB BBB BFE EEE EEE
                 S = SRQ Pending Flag
                 P = Alarm Prog Scheduling Active
                 A = SRQ Interrupt Arming Flag
                 B = # Active BEQT Entries, 0 = 31
                 F = First Direct I/O Request Flag
                 E = # EQT Extension Words, 18-255

EQT13 -Format: I AAA AAA AAA AAA AAA
                 I = Initiator/Continuator Flag
                 A = EQT Extension Address
```



LU 10 → EQT 11 → DA 23B

LU 15 → EQT 11 → DA 3B

LU 20 → EQT 11 → DA 13B

Figure 3-2. DRT — EQT — DA Mapping Scheme

```
EQ 11  SELECT CODE 21, IS AVAILABLE FOR USE.

       TIME OUT WILL OCCUR AFTER   10.00 SECONDS (    .1667 MINUTES).

       14 DEVICE LOGICAL UNITS MAY BE USED ON THIS BUS.
       14 DEVICE LOGICAL UNITS ARE YET UNKNOWN BY THE EQT.

       AVAILABLE LOGICAL UNITS:    17, 18, 19, 32, 33, 34, 51, 52,
                                   53, 54, 55, 56, 63,
       HP-IB LOGICAL UNITS:        15,
```

Figure 3-3. BSCU Listing from ':RU,BS,,11'

```
        TIME OUT WILL OCCUR AFTER   10.00 SECONDS (    .1667 MINUTES).

LU 16      THIS IS A DEVICE INDEPENDENT, GENERAL BUS LOGICAL UNIT       DA 000B

              15 14 13 12 11 10  9  8  7  6  5  4  3  2  1  0
               S  R  D  I  J  O  P  E  X  X  X  X  X  X  X  X
              -----------------------------------------------
               0  0  0  1  1  1  1  0  0  0  0  0  0  0  0  0
              -----------------------------------------------
               0        1        7        0        0        0
              -----------------------------------------------

        S=0  I/O REQUEST NOT ABORTED ON AN SRQ.
        R=0  NO I/O RESTART ATTEMPT AFTER SRQ.
        D=0  DMA IS NOT ALLOCATED FOR THIS DEVICE.
        I=1  REQUIRE AN EOI FROM DEVICE WITH THE LAST BYTE.
        J=1
        O=1  ISSUE AN EOI WITH THE LAST BYTE.
        P=1
        E=0  HP-IB ERRORS WILL ABORT THE PROGRAM.

        BUS STATUS:       REN    REMOTE
                          ATN    DATA
                          LSTN   NT ADDRS
                          TALK   ADDRSSED
                          CNTLR  ACTIVE
                          DAV    HI
                          NRFD   HI
                          NDAC   LO
```

Figure 3-4. "RU,BS,,16" Gives Information About System LU 16

The complete details concerning how the bus is configured may be obtained from the table which follows. The table is straightforward. Some of the information is condensed because certain configuration bits work together in defining the end of record requirements (I and J, O and P).[2]

When a bus logical unit is found, the BSCU will supply user information about the five HP-IB management lines and the three handshake lines.

---

[2] *See the HP-IB Users Manual (59310-90064) for device configuration information.*

Figure 3-5 shows the BSCU output when a device logical unit is specified in the RUN statement,

```
:RU,BS,,,9
```

Each device logical unit may be configured separately. If left unmodified, this configuration defaults to that defined for the bus logical unit. More information is supplied for a device logical unit. The status byte obtained at any time during a serial poll is device dependent. The device will be polled only if SRQ program scheduling was set up previously for the device. The current value for device status is shown.

These examples show only some of the information which the BSCU is capable of supplying. The BSCU outputs only the information applicable to the situation. The outline in figure 3-6 gives a complete description of BSCU characteristics.

The FORTRAN program describes in an easy to read format, the current configuration and status of HP-IB in RTE-M, RTE II, RTE III, and RTE IV when using driver DVR37 (REV. 1926) with SRQ program scheduling.

```
        TIME OUT WILL OCCUR AFTER   10.00 SECONDS (    .1667 MINUTES).

   LU  9                  (DEVICE ADDRESS    DECIMAL)                  DA 001B

                15 14 13 12 11 10  9  8  7  6  5  4  3  2  1  0
                 S  R  D  I  J  O  P  E  X  X  X  X  X  X  X  X
                -----------------------------------------------
                 0  0  0  1  1  1  1  0  0  0  0  0  0  0  0  1
                -----------------------------------------------
                 0        1        7        0        0        1
                -----------------------------------------------

        S=0  I/O REQUEST NOT ABORTED ON AN SRQ.
        R=0  NO I/O RESTART ATTEMPT AFTER SRQ.
        D=0  DMA IS NOT ALLOCATED FOR THIS DEVICE.
        I=1  REQUIRE AN EOI FROM DEVICE WITH THE LAST BYTE.
        J=1
        O=1  ISSUE AN EOI WITH THE LAST BYTE.
        P=1
        E=0  HP-IB ERRORS WILL ABORT THE PROGRAM.

                        15 14 13 12 11 10  9  8  7  6  5  4  3  2  1  0
        DEVICE STATUS:   0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
```

Figure 3-5. "RU,BS,,,9" Will Give Device LU Information

A. HP-IB equipment table information.

1. Time-out value in milliseconds and seconds

2. Availability

   a. available
   b. down
   c. busy
   d. waiting for a DMA channel

3. DMA and output buffering

4. Maximum number of devices which may be used on the bus

5. Maximum number of devices which may yet be assigned

6. Available logical units in the system

7. The logical units which are assigned to the particular bus

B. The HP-IB bus logical unit (when one is assigned)

1. Bus configuration

2. I/O card status information

   a. REN
   b. ATN
   c. LSTN
   d. TALK
   e. CNTLR
   f. DAV
   g. NRFD
   h. NDAC

C. The HP-IB device logical unit (when one or more are assigned)

1. Time-out (if only the device logical unit is requested)

2. Device configuration (see program output)

3. Device status

4. SRQ program scheduling (when applicable)

   a. the current program state

      a1. dormant
      a2. scheduled
      a3. I/O suspend
      a4. general wait
      a5. memory suspend
      a6. disc suspend
      a7. operator suspend

Figure 3-6. Outline of the BSC Utility

The information shown may be obtained in several different ways, in whole or in part depending in the format of the 'RUN' statement for the program. Current statistics may be obtained for:

a. All HP-IB logical units and EQTS.

b. One HP-IB EQT.

c. One HP-IB EQT and all associated logical units.

d. One HP-IB EQT and one associated logical unit.

e. One logical unit.

The 'RUN' statement can be entered from file manager as shown:

```
:RU,BS,ILST,EQT,LU
```

Figure 3-7 gives some examples of how the program might be executed.

The program requires ten pages of memory and the following modules shown in figure 3-8.

The BSCU obtains information from the EQT by making control requests (CALL EXEC(3,...) to DVR37 or references the actual addresses in memory to obtain the information directly from the EQT. Typically, control requests are made when DVR37 must determine the status of the bus or configure the solution to the request as a real-time operation.[3]

All corresponding LUs in the DRT are checked against the variable length EQT extenson area for valid device address identification. When a configured device address is found in the EQT with no corresponding LU in the DRT, the five device words are cleared out so that a legal LU may later use the space.

Each time an LU subchannel entry from the DRT matches a device address from an entry in the EQT, the five-word entry is evaluated in the EQT and output: first, the configuration for the device, then the status word as last received from the device, and then SRQ program scheduling.

When an HP-IB device logical unit is set up for SRQ program scheduling, the BSCU obtains the name of the program from the EQT and then searches the ID segment list[4] for the program to be scheduled and its current status. This information, or a message stating that the program cannot be found is then output.

```
RU,BS              GIVES ALL HP-IB INFORMATION
RU,BS,,11          GIVES INFO FOR HP-IB EQT 11
RU,BS,,11,-1       GIVES INFO FOR EQT11 AND ALL ASSOC. LUS
RU,BS,,11,19       GIVES INFO FOR EQT11 AND LU19
RU,BS,,,19         GIVES INFO FOR LU19
RU,BS,,,-18        GIVES INFO FOR SESSION LU18
RU,BS,,11,-18      GIVES INFO FOR EQT 11 AND SESSION LU 18
RU,BS,,,,1         NON-ZERO PARAMETER 4 SUPPRESSES AUTO LU CLEANUP.
```

Figure 3-7. Run Statement Examples

---

[3] *The BSCU avoids making control requests to the driver when the EQT or the LU is down (which would cause the utility to suspend). Instead, it prints out a message stating that the information is not obtainable when the EQT is in the down state.*

[4] *For this reason, the BSCU must be declared a type 3 program.*

The information shown may be obtained in several different ways, in whole or in part depending in the format of the 'RUN' statement for the program. Current statistics may be obtained for:

a. All HP-IB logical units and EQTS.

b. One HP-IB EQT.

c. One HP-IB EQT and all associated logical units.

d. One HP-IB EQT and one associated logical unit.

e. One logical unit.

The 'RUN' statement can be entered from file manager as shown:

```
:RU,BS,ILST,EQT,LU
```

Figure 3-7 gives some examples of how the program might be executed.

The program requires ten pages of memory and the following modules shown in figure 3-8.

The BSCU obtains information from the EQT by making control requests (CALL EXEC(3,...) to DVR37 or references the actual addresses in memory to obtain the information directly from the EQT. Typically, control requests are made when DVR37 must determine the status of the bus or configure the solution to the request as a real-time operation.[3]

All corresponding LUs in the DRT are checked against the variable length EQT extenson area for valid device address identification. When a configured device address is found in the EQT with no corresponding LU in the DRT, the five device words are cleared out so that a legal LU may later use the space.

Each time an LU subchannel entry from the DRT matches a device address from an entry in the EQT, the five-word entry is evaluated in the EQT and output: first, the configuration for the device, then the status word as last received from the device, and then SRQ program scheduling.

When an HP-IB device logical unit is set up for SRQ program scheduling, the BSCU obtains the name of the program from the EQT and then searches the ID segment list[4] for the program to be scheduled and its current status. This information, or a message stating that the program cannot be found is then output.

```
RU,BS               GIVES ALL HP-IB INFORMATION
RU,BS,,11           GIVES INFO FOR HP-IB EQT 11
RU,BS,,11,-1        GIVES INFO FOR EQT11 AND ALL ASSOC. LUS
RU,BS,,11,19        GIVES INFO FOR EQT11 AND LU19
RU,BS,,,19          GIVES INFO FOR LU19
RU,BS,,,-18         GIVES INFO FOR SESSION LU18
RU,BS,,11,-18       GIVES INFO FOR EQT 11 AND SESSION LU 18
RU,BS,,,,1          NON-ZERO PARAMETER 4 SUPPRESSES AUTO LU CLEANUP.
```

Figure 3-7. Run Statement Examples

---

[3] *The BSCU avoids making control requests to the driver when the EQT or the LU is down (which would cause the utility to suspend). Instead, it prints out a message stating that the information is not obtainable when the EQT is in the down state.*

[4] *For this reason, the BSCU must be declared a type 3 program.*

```
COM     40042 40066
BS      40067 45352  06-13-79 <GWG> CURRENT HP-IB CONFIGURATION
GTLU    45353 45464  06-13-79 <GWG> GET TRUE LU
OUTPT   45463 45527  04-27-78 <GWG> OUTPUT AND CLEAR BUFFER
DFIG    45530 47274  05-23-78 <GWG> WRITE DEVICE CONF.
BFIG    47275 50203  10-02-78 <GWG> BUS BS. AND STATUS
WEHT    50204 50345  07-11-78 <GWG> PAGE EJECT?
HDVR    50346 50361  09-27-78 <GWG> CHECK FOR SPOOL EQT

PAUSE   50362 50462  771122  24998-16001
IGET    50463 50471   750701  24998-16001
LURQ    50472 51054  92067-16268 REV.1903 790223
LOGLU   51055 51132  92067-16268 REV.1903 790228
RMPAR   51133 51175  781106  24998-16001
LUTRU   51176 51307  92067-16268 REV.1903 790223
FMTIO   51310 52603   24998-16002 REV.1901  781107
FMT.E   52604 52604   24998-16002 REV.1901  781107
PNAME   52605 52652   771121  24998-16001
REIO    52653 52777  92067-16268 REV.1903 790316
ERR0    53000 53067   771122  24998-16001
.ITOI   53070 53201  750701  24998-16001
IFBRK   53202 53235  92067-16268 REV.1913 790124
PAU.E   53236 53236  750701  24998-16001
ER0.E   53237 53237  750701  24998-16001
$ALRN   53240 53355  92067-16268 REV.1903 770715
CNUMD   53356 53375  92067-16268 REV.1903 770621
FTIME   53376 53667  92067-16268 REV.1903 780731
DTACH   53670 53750  92067-16268 REV.1903 781202
MESSS   53751 54304  92067-16261 REV.1903 790420
VSCBA   54305 54352  92067-16261 REV.1903 790202
$CVT3   54353 54440  92067-16268 REV.1903 770621
IDGET   54441 54523  92067-16268 REV.1903 790314
FRMTR   54524 60150   24998-16002 REV.1901  781107
CAPCK   60151 60502  92067-16268 REV.1903 790201
$SMVE   60503 60571  92067-16268 REV.1903 790202
$ESTB   60572 60606  92067-16268 REV.1903 790202

10 PAGES RELOCATED    10 PAGES REQ'D    NO PAGES EMA      NO PAGES MSEG
/LOADR:BS      READY AT 10:04 AM  THU.,  14  JUNE, 1979

/LOADR:$END
```

Figure 3-8. Modules Needed for the BSCU

HP-IB Performance Measurements

Chapter 4

This chapter will introduce several equations that may be applied to describe some facets of HP-IB performance. These equations relate HP-IB performance with the HP 1000 using an RTE IV operating system only. This is meant to be an aid in understanding HP-IB, more than a method to analytically describe HP-IB performance. However limited, the model can be used to approximate simple HP-IB performance problems, analytically. The equations become too cumbersome for more complex problems.

These equations describe HP-IB performance and system utilization when I/O is via the interrupt system only.[1] I/O requests, when DMA is active, require an evaluation at another level which is not covered in this section. When DMA requests must be evaluated, the programs in Chapter 5 can be used to do the HP-IB measurements experimentally. For most cases, this gives satisfactory results.

Describing I/O Performance Characteristics

In instrumentation applications, we often wish to determine the rate (RT) at which measurements can be taken from a device or devices in a system. To do this, one must understand the operations involved in obtaining the measurements. In the simplest case, the procedure may involve one I/O request. In FORTRAN or a similiar language, this is one "WRITE" or "READ" statement to or from the HP-IB instrument. When one or more I/O requests are used for the operation, the combination is called a "task-in-process."

From a computer user standpoint, it is important to know the rate (RT) during a task-in-process and to understand how much (as an average) the computer is utilized (%UTL) during the task (task-in-process will be abbreviated task). These phenomena can be described by the variables shown in figure 4-1.

For example, the value for RT can be used to determine the number of measurements a digital voltmeter can obtain per second. The variable %UTL can give us a feel for the number of digital voltmeters that can be connected to the HP-IB interface and run before the central processing unit becomes saturated.

VARIABLE	DESCRIPTION
RT	The rate at which measurements can be taken.
%UTL	The average CPU utilization during a particular I/O operation or task.
TT	Time required for an I/O request.
TF	Time the CPU is free during TT.
TB	Time required by the HP-IB device during TT.
TC	Time used by the CPU for the I/O request during TT.
TCT	Time used for moving data by the CPU during TC.
TCFM	Time used for formatting data by the CPU during TC.
TCI	Time required by the CPU to initiate an I/O request.
TCIO	Time required by the CPU to continue an I/O request.
TCIO1	Discrete time to describe TCIO with dispatching.
TCIO2	Discrete time to describe TCIO without dispatching.
TIDL	Time required by the CPU while in the idle loop.
TDISP	Time required by the CPU while in the dispatching loop.
TP	Time used by the CPU for moving data into memory.
TP2	Time used by the CPU for moving data and switching memory maps.
TP3	Time used by the CPU for moving data without leaving the HP-IB driver.
TFPi	Discrete quantums of time which compose TF.
TS	The approximate setup time required for the I/O request assuming the I/O curve is perfectly linear.
TBn	Time required per byte by the HP-IB instrument.
UT	The average system utilization during an I/O request.
TDVM	Time required for the HP-IB device to obtain a measurement and make it ready for the return trip to the controller.

Figure 4-1. HP-IB Performance Variables

---

[1]*See Application Note AN201-4, "Performance Evaluation of HP-IB Using RTE Operating Systems." Also see the HP-IB User's Manual, especially Chapter 4 (DMA usage) and Chapter 3.*

The request to be measured can be broken down into two general catagories. These are the times used by the CPU and by the HP-IB device. Generally, the time required to complete an I/O request can be described by the time the computer is busy plus the time the computer is free (figure 4-2):

TT = TC + TF

In the HP 1000, an I/O request may take several different forms, from the simplest "CALL EXEC" request for ASCII input:

```
CALL EXEC (1,IDLU,IBF,INUM)
```

to the most sophisticated, free field, formatted input (which converts ASCII to binary):

```
READ (IDLU,*)A,B,C(10)
```

In each case, TT takes on a different meaning (depending on the user's requirements), but it always represents the total time required to complete an I/O task.

NOTE

In a real-time computer like the HP 1000, a system clock is available which allows the time to be recorded before the I/O task is begun and again after the task has completed. Subtraction of these two times allows TT to be determined.

Practically speaking, the total time required during an I/O task may involve more than just returning measurements from an HP-IB device. Most HP-IB devices return data in ASCII format which must be converted to binary either to reduce the memory space required to store the data, or to perform computations on the data, or both. Because this formatting requires 100% of the CPU's computing ability, the time used by the computer during the task (TC) may be described as,

TC = TCT + TCFM

where TCT represents the summation of the individual TCT times in the picture (see figure 4-3). The individual TCT times represent the time required for I/O transfers by the CPU, and TCFM represents the time required for formatting the data.

TT = TC + TF

where,

TC = The time used by the CPU during TT.

TT = The time required to transfer the measurement data.

TF = The time the CPU is free during TT. It is also the time required by the HP-IB instrument during TT.

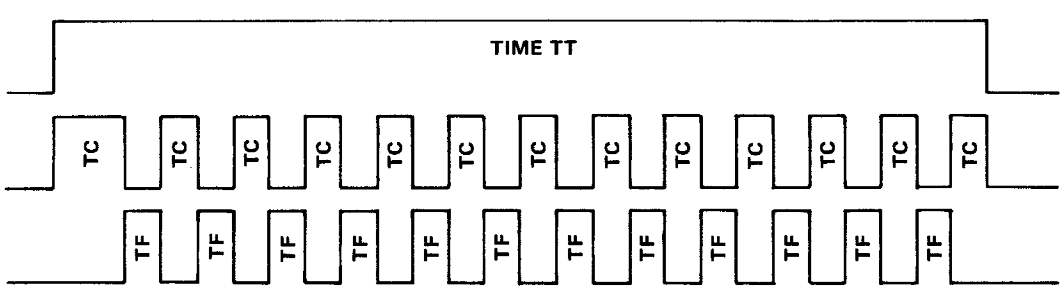


Figure 4-2. Detailed List of Performance Variables

$$TC = TCT + TCFM$$

where,

TCT = CPU time required during I/O.

TCFM = CPU time required during formatting.

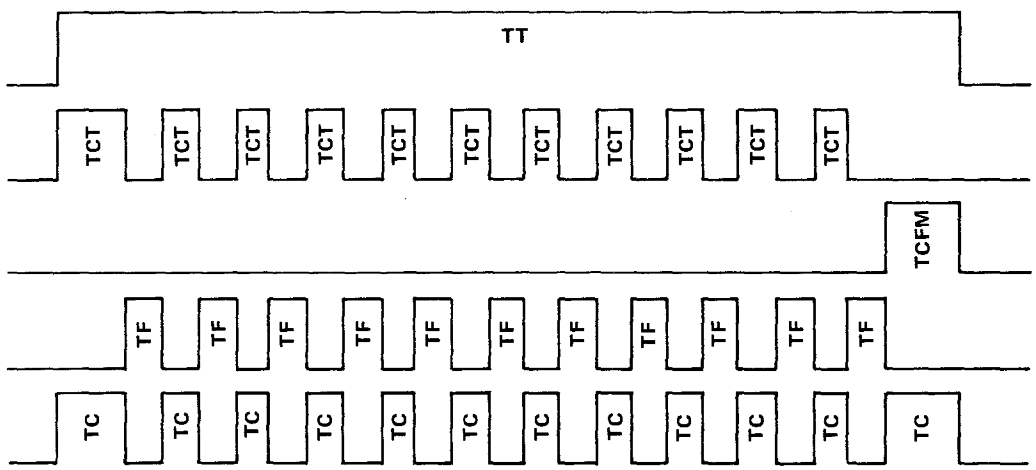


Figure 4-3. Breakdown of TC

When the input data from the HP-IB device will simply be displayed, no formatting is required and TCFM = 0. Therefore, TC (the CPU time used for the I/O request) is equal to TCT.

$$TC = TCT$$

when TCFM=0.

I/O Operations (TCIO)

The RTE operating system remains active during an I/O request. For this discussion, assume that TCFM = 0. Exploring TCT shows that,

$$TCT = TCI + TCIO$$

where,

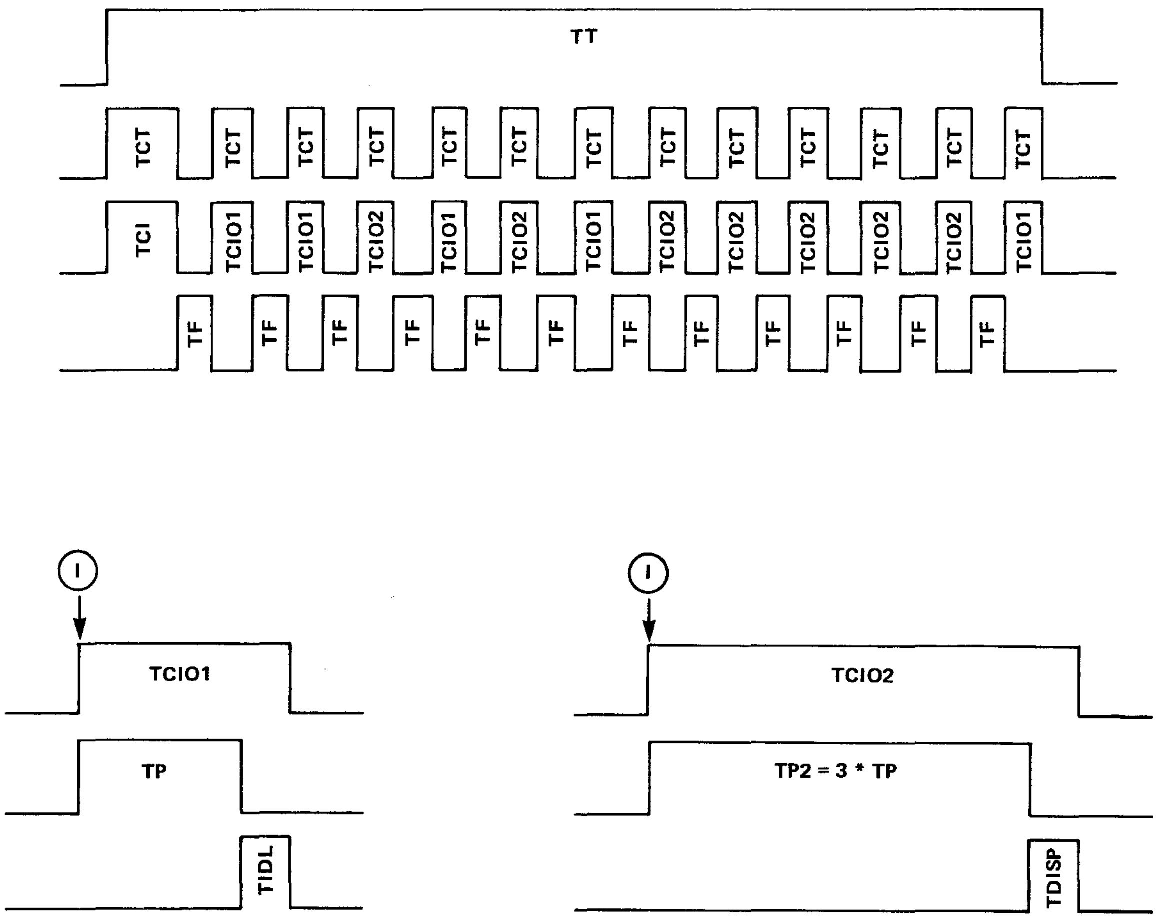
$$TCIO = \sum_{\substack{i = 1,n \\ j = 1 \text{ or } 2}} TCIO_{ji}$$

$$= TCIO_{11} + TCIO_{22} + TCIO_{13} + \ldots + TCIO_{2i}$$

The value TCI (figure 4-4) represents the time required to initiate an I/O request and to set up all of the required pointers, flags, etc. $TCIO_{ji}$ is the time required by the CPU to continue an I/O request after each interrupt until all the data has been transferred. TCIO is simply the sum of these discrete quantums of time (see figure 4-4).

After each interrupt the RTE operating system may be in one of two states ($TCIO_{1i}$ or $TCIO_{2i}$). Figure 4-4 shows that the quantum $TCIO_1$ or $TCIO_2$ may occur during the continuation of I/O. Which state depends on whether other programs can be dispatched[2] during the computer's free time (TF).

[2]*A program must be in the scheduled state (state 1) before it can be dispatched. Many programs may be in the scheduled list (state 1), but only one program can be dispatched at a time.*



(I) indicates an interrupt was generated

$TCIO_1$ — The operating system is idle and no other programs are being dispatched during the I/O request.

$TCIO_2$ — If other programs are scheduled, the Real Time Executive will attempt to dispatch programs during CPU free time (TF).

Figure 4-4. Visual Description of $TCIO_1$ and $TCIO_2$

The term TP enters the equations for $TCIO_1$ and $TCIO_2$ as shown in the flowchart in figure 4-5.

$$TCIO_{1i} = TP + TIDL$$
$$TCIO_{2i} = 3 * TP + TDISP$$

In RTE-IV the HP-IB driver is mapped into the user's partition. When the operating system is idle ($TCIO_1$), the user's map will remain active during the I/O request. After each interrupt a given amount of time (TP) will be required to move the data in or out of memory.

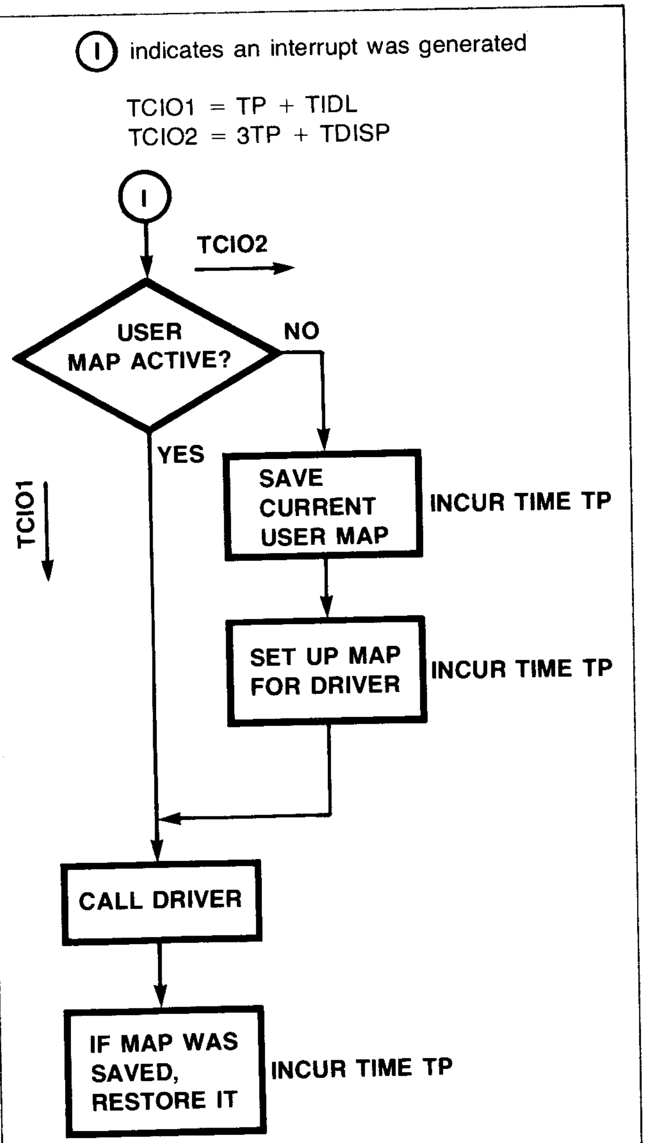


Figure 4-5. Flowchart of Map Switching Time

When other programs are active in the system, TP2 will be incurred which is longer than TP because user maps must be switched and saved.

Notice that the variables $TCIO_1$ or $TCIO_2$ also contain the factors TIDL or TDISP respectively (figure 4-4). They are constants and for most situations,

$$TIDL \approx TDISP$$

Processing Very Fast Input

In some cases the maximum transfer rate is faster for the HP-IB instrument than it is for the HP 1000 which reduces the equations to,

$$\begin{aligned} TT &= TCT \\ &= TCI + TCIO \\ &= TCI + n * TP3 \end{aligned}$$

where TP3 is a quantum of time required to process incoming or outgoing data within the HP-IB driver (in this mode the driver operates with the interrupt system turned off). The value "n" is the number of words transmitted. Basically, the HP-IB driver initiates an I/O transfer and then just a short time later (several microseconds) checks to see if the operation has already completed. If it has, it loops around and starts the next one.

NOTE

Unless it is acceptable to dedicate the entire system usage (100 percent utilization) to the operation, this method is not suggested.[3] It may be disabled by allocating DMA to the device.

CPU Free Time (TF)

One quantity of interest for the HP 1000 system user is TF (the time left for another program to execute during a previously started I/O request) Note that,

$$TF = \sum_{i=1,n} TFP_i =$$

$$TFP_1 + TFP_2 + TFP_3 + \ldots + TFP_i$$

is HP-IB instrument dependent and will vary from device to device and task to task as shown in figure 4-6.

---

[3] *Some HP instruments such as the 3437A Digital Voltmeter, the 2240A Measurement and Control Processor, and the 5345A Counter are capable of operating in this mode. DMA is suggested for these devices.*

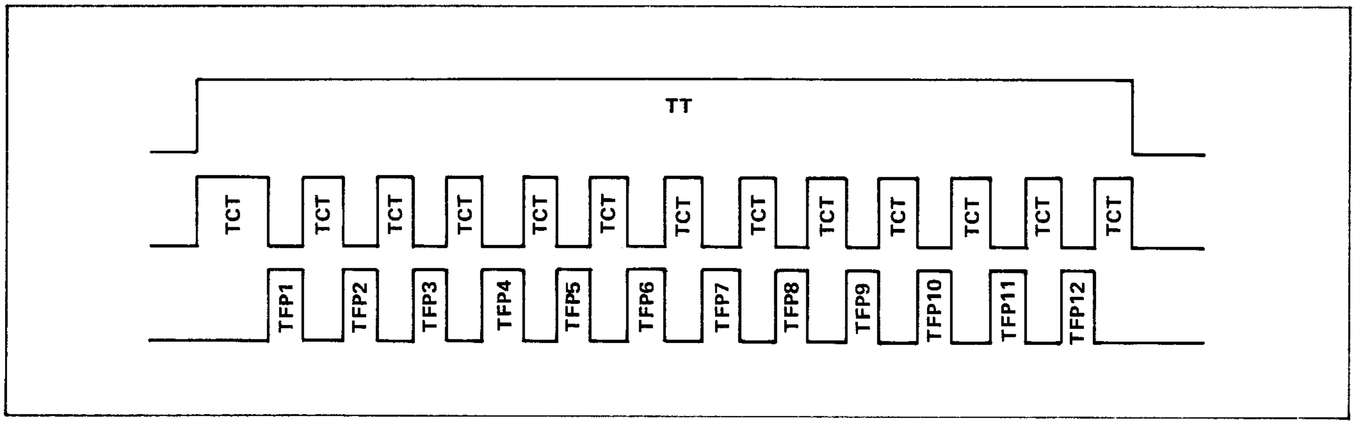


Figure 4-6. Timing Diagram of TF

NOTE

The quantity TF may be determined empericaly using software. Dispatching a lower priority program during the I/O request idle time provides this information. Similiarly TT may be determined.

This value (TF) supplies some information about how the HP 1000 is utilized (as a percentage) during the I/O task (figure 4-7).

Note that as,

$$TF \rightarrow 0$$
$$UT \rightarrow 100\%$$

and,

$$TCIO \rightarrow n * TP3$$

$$UT = 100 - \frac{TF}{TT} * 100 \qquad \text{(in percent)}$$

Figure 4-7. The System Utilization Equation

Linear Approximations

A graphical representation can be used to help visualize the above concepts when applied to a real situation.

An output from the computer to the HP-IB device under a given set of conditions, for example, may be graphed as the number of bytes output vs time. An approximately linear curve will result.

An approximated version of the curve will have a positive y-intercept and a positive slope (as shown in figure 4-8).

$$TT = TS + n * TB_i$$

Note that this is anologous to:

$$TT = TC + TF$$

$$= (TCT + \cancelto{0}{TCFM}) + TF$$

$$= TCI + \sum_{\substack{i=1,n \\ j=1 \text{ or } 2}} TCIO_{ji} + \sum_{i=1,n} TFP_i$$

Under most conditions:

$$TS \approx TCI$$

$$TCIO_{1i} \approx TCIO_{2i}$$

$$TFP_1 \approx TFP_2 \approx TFP_3 \ldots . \approx TFP_n$$

$$TB_i \approx TCIO_{ji} + TFP_i$$

$$TB_1 \approx TB_2 \approx \ldots . TB_n$$

The quantity TS (the y-intercept) is a generalized approximation of the time required to get an I/O request initiated before any bytes are transmitted or received over the bus. Once this setup time has finished, n bytes will be transferred at time TB_n per byte.[4]

---

[4]*This is only a linear approximation to the curve which represents actual transfer times observed via HP-IB in the test system. The emperical curve is* not *linear especially in the lower region (when less than two bytes have been transferred). This equation was used in AN 401-4. More information can be obtained there.*

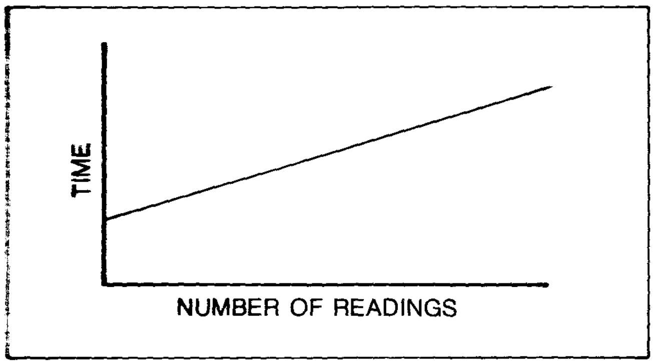


Figure 4-8. Example Linear Performance Graph

Obtaining Measurement Rate RT and %UTL

So far, only single I/O requests have been discussed. These I/O request equations can now be used to construct a conceptual model of HP-IB system performance for a task. The final objective is to calculate the rate (RT) at which measurements can be obtained from an instrument and the average system utilization during the task (%UTL).

While constructing these models it is helpful to draw a picture showing the physical timing of the I/O requests, and their relative "time lengths" to one another. In some cases these times overlap (see the following example), causing the total time TT to be shorter than one would initially suspect. The time axis can be broken up into separate blocks, each one specifying the controller or one or more instruments on the bus.

Most simple HP-IB devices go through the same basic steps to obtain a measurement. Figure 4-9 shows a breakdown of the times required to program, trigger, and read a measurement from an HP-IB voltmeter.[5]

Naturally, the sequence won't always occur in this order, but the conceptual model will obtain a ball park figure, depending on the assumptions made concerning how the measurements are taken. A description of the performance variables involved are shown in figure 4-10.

The quantities which determine UT and TT for each of the three operations during the task are shown in figures 4-7 and 4-11. The quantity "n" is the number of bytes transmitted or received.

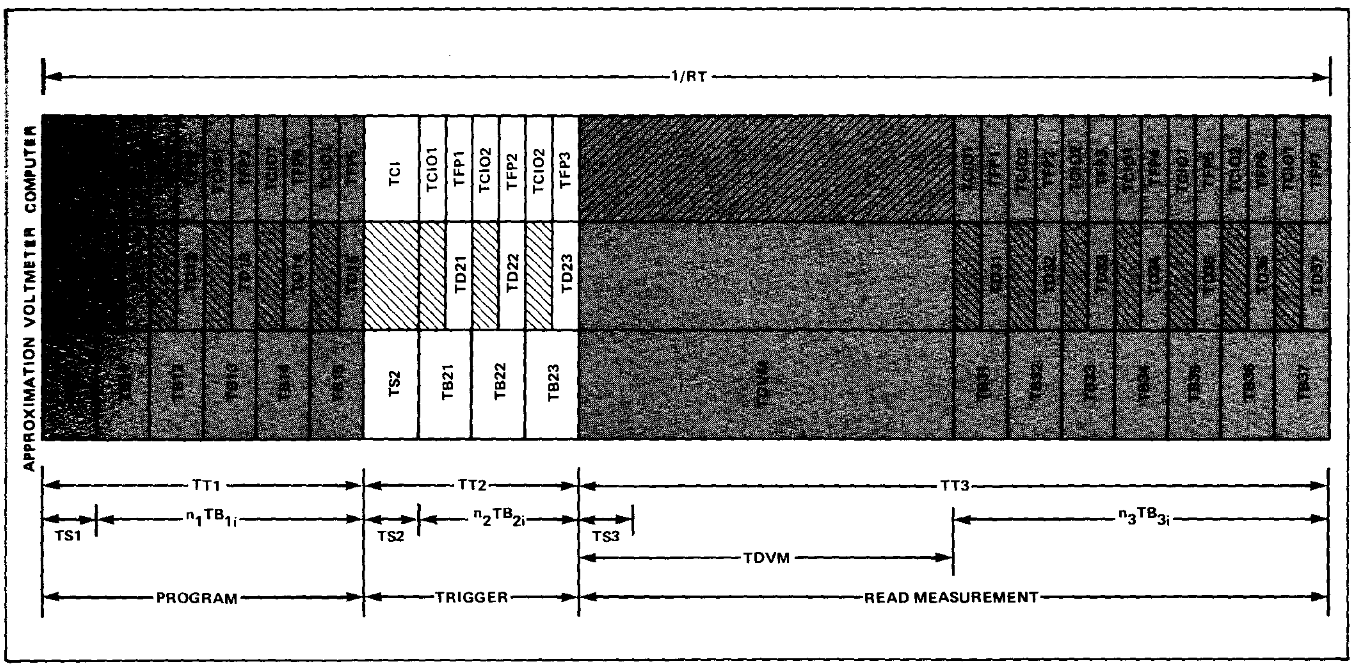


Figure 4-9. "Time Axis" of HP-IB Performance

[5] *The model developed here is the same as that in AN 401-4 but the actual times will be replaced with the performance variables developed thus far in this chapter.*

1. TIME 'TT1'

   The computer talks while the instrument listens. Programming information is then sent to the device.

2. TIME 'TT2'

   The computer talks while the instrument listens. The device is activated to take the measurement.

3. TIME 'TT3'

   The computer listens while the instrument talks. The instrument returns the measurement to the computer.

Figure 4-10. Typical Measurement Sequence

TT	=	The total time required for the I/O operation.
TS	=	The approximated setup time required for the operation, assuming the I/O curve is perfectly linear.
TB_i	=	The time per byte required by the HP-IB instrument.
TCI	=	The CPU time required to initiate an I/O operation.
TCIO	=	The CPU time required to continue an I/O operation.
TF	=	The free time available to another program during a previously started I/O request.

Figure 4-11. Quantities Which Determine UT and TT

Figure 4-9 shows that the voltmeter requires a quantum of time for each byte transferred.

This can be represented as,

$$TD = TD1 + TD2 + \ldots TDn$$

In the general sense, as,

$$TD \rightarrow 0$$
$$UT \rightarrow 100\%$$

From this discussion it can be seen that performance can be approximated analytically if certain conditions can be reasonably estimated (figure 4-12).

TS	=	Is a psuedo setup time derived as a y-intercept from an approximately linear curve.
TBn	=	The transfer time per byte for the HP-IB device (used when TDn > TP3). Note, in most cases the assumption TD1 = TD2 = . . . = TDn is used.
TDVM	=	The time required for the HP-IB device to obtain a measurement and make it ready for the return trip to the controller.

$TCIO_{ji} << TFP_i$ or TD_i IE. $TCIO_{ji}$ is negligible.

Figure 4-12. Analytical Performance Criteria

The controller can only perform one operation at a time (it cannot process incoming data at the exact same moment that the data is arriving from the HP-IB device, for example).

TS is approximated using worst case figures. This can be done by experimentally finding the maximum rate performance curve for the HP 1000. By extrapolating a line back to the y-intercept (zero bytes transmitted) a worst case figure for TS can be found.

The transfer time per byte (TD_i) for the HP-IB instrument and the measurement time required by the instrument (TDVM) are device dependent. Once determined (from the HP-IB device manual or data sheet) they are fixed quantities. TD_i maximum, for the HP 1000, is given in AN 201-4. If TD_i for the HP 1000 is greater than TD_i for the device, the HP 1000 value should be used. Remember that using TD_i for the HP 1000 assumes that TFP_i is zero (during the actual byte transfer) and the HP 1000 is being utilized 100%.

TT, TB_i, and TDVM determine the system utilization (%UTL) during the task (as shown in figure 4-13).

TT is determined by using TS, TB_i, TDVM, and n. See figure 4-14.

This chapter has summarized a complete set of equations which approximate HP-IB performance characteristics. Although the equations can be cumbersome, the characteristics are definable and preliminary estimates can be made. See Chapter 5 for a description of how to do performance measurements using HP 1000 software and specific HP-IB devices.

$$TF1 = (n1) * TB11$$

$$TF2 = (n2) * TB21$$

$$TF3 = (n3) * TB31 + TDVM - TS3$$

$$TF = TF1 + TF2 + TF3$$

$$\%UTL = 100 - \frac{TF}{TT} * 100$$

Figure 4-13. The Final Performance Utilization Equation

$$TT1 = TS1 + (n1) * TB11$$

$$TT2 = TS2 + (n2) * TB21$$

$$TT3 = TDVM + (n3) * TB31$$

$$TT = TT1 + TT2 + TT3$$

$$RT = 1/TT$$

Figure 4-14. The Final Rate Equation RT

HP-IB Performance Programs

Chapter 5

This chapter will show how the theoretical equations discussed in Chapter 4 may be applied to determine HP-IB performance in an HP 1000 system. These performance equations have been duplicated in figure 5-1.

Several simple programs will be introduced which were used to graph the performance of HP-IB devices in earlier chapters. The programs can be used to obtain measurements via the interrupt system, or using DMA. The performance results of these programs are a function of:

- The type of CPU and memory being used.
- The number of EQTs in the system.
- The number of programs in the time list.
- The priority and type of program interacting with the HP-IB device.
- Whether output buffering has been implemented.
- Whether DMA is used.
- The capability and speed of the HP-IB device.
- Whether the measurements are saved in memory or output to another peripheral.

Program Descriptions

A series of four programs can be used to characterize HP-IB performance:

MEMAA
FIGUR
PERF
*PERF

Two assembly subroutines are used by FIGUR and PERF:

CMPUT
TIMEX

The subroutine CMPUT computes the difference between two octal double words, and subroutine TIMEX computes the total elapsed time. The tests may be run on any RTE system — RTE-II, RTE-III, or RTE-M. Any CPU may be used — the HP 1000 M-series, E-series, or F-series. At least four words of system common must be allocated at generation time for the measurements.

$$TT = TC + TF \qquad (a)$$

$$TT = TS + nTB \qquad (b)$$

$$\%UTL = \left[1 - \frac{TF}{TT}\right] * 100$$

TT	=	Time required to transfer the measurement data.
TF	=	Time the CPU is free during TT.
TC	=	Time used by the CPU during TT.
TS	=	Approximated setup time required to start the I/O operation.
n	=	Number of bytes transmitted or received.
TB	=	Time per byte during the transfer.
RT	=	Rate at which measurements can be taken.
%UTL	=	Average CPU utilization during a particular task.

Figure 5-1. Performance Equations from Chapter 4

The four programs enable the user to solve the equations in figure 5-1 for an HP-IB "task in process."[1]. The task will vary from device to device but basically revolves around those statements necessary to perform a repetitive measurement operation of some kind, or a repetitive programming operation.

To backtrack a bit, the meaning of "HP-IB system response time" has to do with the time required for the CPU to output, input, and/or perform a series of operations on the HP-IB. In the simplest case, this would be the total time from start to finish to perform a particular task. However, RTE-M, RTE II, and RTE IV, are somewhat more sophisticated because they are designed for multiprogramming (i.e., they are capable of performing other tasks before the particular task in process is finished).

For the eight given conditions listed on the previous page, and as long as system utilization (%UTL described in Chapter 4) is less than 100%, the total time required to perform a given task will always remain the same. If the CPU is not being utilized 100% by the task in process, it spends its extra time doing other constructive work (if there is constructive work to be done). The percentage amount of time (%UTL) which is being used by the task in process is denoted as system utilization. Note that if the system is being utilized 100%, the time required for the task in process will vary each time the task is run (assuming that all of the tasks in the system have equal priorities).

The multiprogramming capability of RTE can be used to determine system utilization by executing a lower-priority program during the time which the task program (performance test) is suspended. The lower-priority program counts at a known rate from which TF (the CPU free time, equation a in figure 5-1) can be found.

In a real-time operating system like RTE, the real-time clock causes the proper system tables to be updated every ten milliseconds. This process incurs a basic "operating system overhead" which can be described as a percentage (of time) and normalized out of the CPU's available time to do user oriented tasks. This percentage of time can only be regained by turning off the interrupt system or using privledged I/O.

Obtaining Operating System Overhead

Before HP-IB performance measurements can be started, the operating system overhead must be determined by using the two programs shown in figures 5-2, 5-3, and 5-4. Operating system overhead usually falls between 1% and 10%.

NOTE

> When running the overhead programs, the system should be quiescent. That is, no programs should be in the time list unless this is the condition which generally prevails. No other programs should be scheduled or actually running.
>
> The interrupt system is turned completely off for five minutes during the overhead procedure.

The program MEMAA shown in figure 5-2 is executed in a special mode to perform two functions. The first function is to determine the number of counts per second it can obtain with the operating system turned on. The second function is to determine the number of counts per second it can obtain with interrupt system off.

Once these values are recorded, MEMAA schedules program FIGUR (figure 5-3) which computes the ratio of the values and thus determines the operating system overhead. The number of counts/sec MEMAA obtains without the operating system, together with the operating system overhead value, are then saved in a file labeled "HPIB" on system disc LU 2. Later, the performance measurement programs will access the file to determine normalized performance times and operating system utilization. (See the "Procedure" section at the end of this chapter for complete system requirements and procedures for relocating and executing programs MEMAA and FIGUR.)

Once the operating system overhead programs have been run successfully, the actual HP-IB measurements can be conducted using the FORTRAN program shown in Figures 5-5 and 5-6.

---

[1] *In its simplest form, the "task in process" is one executable FORTRAN statement; however, it may be multiple statements or involve transferring control to another program as long as control is transferred back before the task completes.*

```
ASMB,L
      NAM  MEMAA,1,32767   01-04-77 (GWG) HP-IB COUNTER
      EXT  EXEC,$LIBR,$LIBX,RMPAR
      ENT  MEMA
      COM  UTIL2(1),UTIL1(1)
A     EQU  0
B     EQU  1
SCHED DEC  10
NPRV1 NOP
NPRV2 NOP
PRV1  NOP
PRV2  NOP
INAM  ASC  3,FIGUR
ICOD  DEC  6
NAM   DEC  0
************************************************
ONF   JSB  EXEC
      DEF  TERM
      DEF  SCHED
      DEF  INAM
      DEF  NPRV1
      DEF  NPRV2
      DEF  PRV1
      DEF  PRV2
*****************************************************
TERM  JSB  EXEC        TERMINATE PROGRAM
      DEF  WAT2
      DEF  ICOD
      DEF  NAM
      DEF  NAM
WAT2  JMP  *
*****************************************************
PRAM  BSS  5
MEMA  NOP
      JSB  RMPAR
      DEF  RT1
      DEF  PRAM
RT1   LDA  PRAM+1
      SZA,RSS
      JSB  LOP1
      JSB  LOOP        DETERMINE COUNTS IN OP-SYSTEM
      JSB  LOP1
      STA  NPRV1
      STB  NPRV2
      JSB  WAIT
      JSB  WAIT
      JSB  WAIT
      JSB  WAIT
      JSB  WAIT
      JSB  WAIT
      JSB  PRV
      JMP  ONF
```

Figure 5-2. MEMAA, the HP-IB Performance Counter

```
*********************************************************
PRV    NOP                 DETERMINE COUNTS OUTSIDE OF OP-SYSTEM
       JSB  $LIBR
       NOP
       JSB  LOOP
       JSB  LOP1
       STA  PRV1
       STB  PRV2
       JSB  $LIBX
       DEF  PRV
********************************************************
* THE NEXT SECTION JUST UPDATES TWO LOCATIONS IN
* COMMON.
********************************************************
D.0    NOP
LOP1   NOP
       LDA  D.0
       OTA  1
       STA  UTIL2
       STA  UTIL1
REPT   ISZ  UTIL1
       RSS
       ISZ  UTIL2
       LDB  UTIL2
       OTB  1
       LIA  1
       SSA
       JMP  CNT
       JMP  REPT
CNT    LDA  UTIL2          YES, OUTPUT COMMON
       LDB  UTIL1          READY TO OUTPUT THE NEXT COMMON LOCATION.
       JMP  LOP1,I         FINISHED!
SEVNS  OCT  77777
LOOP   NOP
       LIA  1
       AND  SEVNS          CLEAR BIT 15
       OTA  1
RTN    CLE
       JSB  WAIT
       CME
       JSB  WAIT
       LIA  1              GET BIT 15 OF THE SWR.
       SSA,RSS             IS IT SET?
       JMP  RTN
       JSB  WAIT
       JMP  LOOP,I         YES, WAIT FOR INSTRUCTIONS.
CNTR   NOP
CNT1   NOP
LARG   OCT  32767
WAIT   NOP
WATE   ISZ  CNTR
       JMP  WATE
       CLB
       STB  CNTR
       STB  CNT1
       JMP  WAIT,I
       END  MEMA
```

Figure 5-2. MEMAA, the HP-IB Performance Counter (Continued)

```
FTN4,L
      PROGRAM FIGUR(3),03-28-78 (GWG) SAVES INFO IN HPIB
C
C
C  FIGUR CALLS 'CMPUT' TO COMPUTE THE DIFFERENCE
C  BETWEEN TWO OCTAL DOUBLE WORDS AND DIVIDES THE
C  THE RESULT BY 300 SEC (5 MINUTES).
C  IT THEN PUTS THE RESULT INTO A FILE: HPIB:RT:-2:3:1
C
C
      DOUBLE PRECISION UTIL,PCNT
      DIMENSION IBUF(60),IPRAM(5),NAM(3)
      DIMENSION IDCB(144),IHI(2),ILO(2),IFHI(2),IFLO(2),UTL(2)
      EQUIVALENCE (IREG,REG,IA),(IREG(2),IB)
      DIMENSION IREG(2),IXH(2),IXL(2)
      INTEGER CREAT,CLOSE,WRITF
      DATA NAM/2HHP,2HIB,2H  /
      CALL RMPAR(IPRAM)
      IFHI(1)=IPRAM(3)
      IFLO(1)=IPRAM(4)
      IFHI(2)=IPRAM(1)
      IFLO(2)=IPRAM(2)
      ILU=1
C  CREATE A FILE FOR THE OUTPUT DATA
      IF(CREAT(IDCB,IERR,NAM,1,3,2HRT,-2).LT.0) GO TO 800
      WRITE(ILU,1900)
1900  FORMAT(/" FIGUR: DATA FILE 'NAMR' IS       HPIB:RT:-2:3:1")
C READ IN DATA
C FIRST READ 'MEMAA' COUNTS OUTSIDE OF OPERATING SYSTEM
      DO 685 I=1,2
      IHI(I)=0
      ILO(I)=0
C CALL CMPUT TO ACTUALLY COMPUTE THE DIFFERENCE
      CALL CMPUT(IHI(I),ILO(I),IFHI(I),IFLO(I),IXH(I),IXL(I))
      UTL(I)=IXH(I)*32768.+IXL(I)
      UTL(I)=UTL(I)/300.
685   CONTINUE
      PCNT=(UTL(1)-UTL(2))/UTL(1)
      CALL CODE
      WRITE(IBUF,601)UTL(1),PCNT
601   FORMAT(D15.12,", "D15.12", ")
C WRITE THE RECORD
      IF(WRITF(IDCB,IERR,IBUF,17).LT.0)GO TO 800
C WRITE AN EOF
81    IF(WRITF(IDCB,IERR,IBUF,-1).LT.0)GO TO 800
C CLOSE THE FILE
85    IF(CLOSE(IDCB,IERR).LT.0)GO TO 800
      GO TO 84
800   WRITE(ILU,908) IERR
908   FORMAT(" FMP ERROR "I6" ON FILE 'HPIB' -- FIGUR ABORTED.")
      GO TO 900
84    WRITE(ILU,189)
189   FORMAT(/"FIGUR: TASK COMPLETED.")
900   END
```

Figure 5-3. FIGUR, Computes and Saves System Overhead

```
ASMB,L
      NAM CMPUT,8     11-29-76 (GWG) COMPUTE DOUBLE OCTAL
      EXT  .ENTR,EXEC
      ENT  CMPUT
*********************************************************
* THIS IS A GENERAL PURPOSE ROUTINE WHICH PICKS UP
* TWO OCTAL DOUBLE WORDS,
* COMPUTES THE DIFFERENCE BETWEEN THEM AND
* RETURNS THE RESULT IN TWO DIFFERENT PLACES.
* CALL SEQUENCE IS AS FOLLOWS:
*       CALL CMPUT(IPRM1,IPRM2,IPRM3,IPRM4,IPRM5,IPRM6)
*                   IPRM1 = HIGH BITS INITIAL UTIL
*                   IPRM2 = LOW  BITS INITIAL UTIL
*                   IPRM3 = HIGH BITS FINAL   UTIL
*                   IPRM4 = LOW  BITS FINAL   UTIL
*                   IPRM5 = HIGH BITS OF RESULT
*                   IPRM6 = LOW  BITS OF RESULT
*
*  NOTE THE FORMAT OF THE HI AND LO RESULT BITS:
*  THE LOW BITS ARE FIFTEEN SIGNIFICANT (THE SIGN
*  BIT IS ALWAYS ZERO. THE HIGH BITS ARE A CONTINUATION
*  OF THE LOW BITS (THE LOW REGISTER SIGN BIT HAS BEEN
*  SHIFTED INTO THE HI REGISTER LSB).
*  THIS WAY THE CALLING PROGRAM CAN PICK UP THE TWO
*  REGISTERS (FROM COMMON OR THE CALL PARAMETERS)
*  AND COMPUT THE COMPLETE NUMBER AS FOLLOWS:
*  X = IPRM5 * 32767.+ IPRM6
*
*********************************************************
*
*
A     EQU  0
B     EQU  1
LIST  OCT  1
FRMT  ASC  4,(2(I6))
PASS  DEC  14
WRTF  DEC  2
DQLT  OCT  2
LNTH  DEC  2
ARGS  BSS  6
CMPUT NOP
      JSB  .ENTR
GET   DEF  ARGS
      LDB  ARGS,I       * B HAS HI INITIAL BITS
      LDA  ARGS+1,I     * A HAS LO INITIAL BITS
      CMB               * ONE'S COMPLEMENT ON HI INIT. BITS
      CLE               *
      CMA,INA           * TWO'S COMP. ON LOW INIT. BITS
      SEZ               * CARRY INTO B?
      INB               * YES, INCREMENT B
      CLE               *
      ADA  ARGS+3,I     + ADD TO A THE LO FINAL BITS
      SEZ               + CARRY OUT OF A?
      INB               + YES!
      ADB  ARGS+2,I     + ADD TO B THE HI FINAL BITS
```

Figure 5-4. CMPUT, Subroutine used by FIGUR and PERF

```
*  HIGHER WORD IN B AND LOWER WORD IN A
*
        SSA                  SIGN BIT SET?
        JMP   INC            +YES!
        ELB                  +NO! JUST SHIFT B.
        JMP   GO
INC     ELB                  SHIFT AND
        INB                  INCREMENT B.
        ELA,CLE,ERA          CLEAR SET SIGN IN A!
GO      STB   ARGS+4,I
        STA   ARGS+5,I
RTN     JMP   CMPUT,I        *TERMINATE SUBROUTINE
        END   CMPUT
```

Figure 5-4. CMPUT, Subroutine used by FIGUR and PERF (Continued)

```
FTN4,L
      PROGRAM PERF(3),01-11-79 (GWG) MEASURE TASK PERFORMANCE,UTIL
C
C RU,PERF,INPUT,IDLU,ILN,ITER,IUTF
C
C INPUT= INPUT LOGICAL UNIT
C IDLU = DEVICE LOGICAL UNIT
C ILN  = NUMBER OF MEASUREMENTS TO BE TAKEN
C ITER = # OF ITERATIONS (DETERMINES ACCURACY)
C IUTF = UTILIZATION FLAG 1 MEANS NO UTILIZATION
C
      COMMON IUT2,IUT1
      DOUBLE PRECISION COUT,OHED,UTL
      DIMENSION IPRAM(5),ISTRT(5),IENDT(5),IDCB(144),NAM2(3)
      DIMENSION IDFT(5),IREG(2),ISEC(5),IBFR(60),MEMA(3),IUTL(2)
C
      INTEGER   IBUF(100),ISTAT(2)
      REAL      OUTBUF(100)
C
      INTEGER OPEN,CLOSE,READF
      EQUIVALENCE (REG,IREG,IA),(ISEC,SEC),(IREG(2),IB)
      EQUIVALENCE (ILU,IPRAM),(IDLU,IPRAM(2)),(ILN,IPRAM(3))
      EQUIVALENCE (ITER,IPRAM(4)),(IUTF,IPRAM(5))
      DATA NAM2/2HHP,2HIB,2H  /
      DATA MEMA/2HME,2HMA,2HA /
      CALL RMPAR(IPRAM)
      IF(IDLU.NE.0)GO TO 1010
      WRITE(ILU,1011)
1011  FORMAT(" :RU,PERF,INPUT,IDLU,#MEAS,ITER,IUTF")
      GO TO 900
1010  IF(ITER.EQ.0)ITER=100
C OPEN HPIB FILE AND GET OVERHEAD DATA
      IF(OPEN(IDCB,IERR,NAM2).LT.0) GO TO 800
      IF(READF(IDCB,IERR,IBFR,60,LEN).LT.0)GO TO 800
      CALL CODE
```

Figure 5-5. PERF, A User Program for HP-IB Performance Measurements

```
      READ(IBFR,*)COUT,OHED
      IF(CLOSE(IDCB,IERR).LT.0) GO TO 800
C INITIALIZE LOOP COUNT
      IUT2=0
      IUT1=0
C SCHEDULE MEMA
      IF(IUTF.EQ.0)CALL EXEC(10,MEMA)
C
C==================================================
C ENTER USER STATEMENTS OUT OF TEST HERE.
C
      WRITE(IDLU,1111)
 1111 FORMAT("D.00001SR3T1F2")
      WRITE(IDLU,1110)ILN
 1110 FORMAT("N"I3"S")
C
C
C=====================================================
C  GET INITIAL UTILIZATION
      ISTR2=IUT2
      ISTR1=IUT1
C  GET THE START TIME
      CALL EXEC(11,ISTRT)
      DO 100 J=1,ITER
C
C======================================================
C ENTER USER STATEMENTS FOR TEST HERE.
C     DO 100 IJ=1,ILN
C     READ(IDLU,*)A
      REG= EXEC(1,IDLU+100B,IBUF,ILN)
      IF(CNVRT(IBUF,IB,OUTBUF).LT.0) GO TO 800
C USER STATEMENTS FOR TEST END HERE.
C=====================================================
C
 100  CONTINUE
C
C  GET FINISH COUNT
C
      IFIN2=IUT2
      IFIN1=IUT1
C
C GET FINISH TIME
C
      CALL EXEC(11,IENDT)
C TURN OF MEMA
      IF(IUTF.EQ.0)CALL EXEC(6,MEMA,0)
C
C  COMPUTE THE DOUBLE WORD COUNT
C
      CALL CMPUT(ISTR2,ISTR1,IFIN2,IFIN1,IUTL(1),IUTL(2))
C
C  COMPUTE THE ELAPSED TIME
C
      CALL TIMEX(ISTRT,IENDT,IDFT)
      SEC=IDFT(3)*60.+IDFT(2)+IDFT(1)*1.E-02
```

Figure 5-5. PERF, A User Program for HP-IB Performance Measurements (Continued)

```
          WRITE(ILU,111)SEC
    111   FORMAT("SEC= "E14.6)
C DETERMINE TOTAL COUNTS FROM MEMAA
          UTL=IUTL(1)*32768.+IUTL(2)
          TNORM=(SEC-SEC*OHED)/ITER
          TFREE=1./COUT*UTL/ITER
          TPER=TNORM-TFREE
          UTIL=(TNORM-TFREE)/TNORM*100.
          IF(IUTF.EQ.0) WRITE(ILU,300)TNORM,UTIL
    300   FORMAT(" PERF: TASK TIME = "F7.5" SEC.  UTILIZATION = "F7.2,
         &" %")
          IF(IUTF.NE.0)WRITE(ILU,301)TNORM
    301   FORMAT("PERF: TASK TIME = "F7.5" SEC.")
          GO TO 900
    800   WRITE(ILU,200)IERR
    200   FORMAT(" PERF: FMP ERROR ",I6," ON FILE 'HPIB'.")
     900  END
```

Figure 5-5. PERF, A User Program for HP-IB Performance Measurements (Continued)

```
ASMB,L
      NAM  TIMEX,7  COMPUTE ELAPSED TIME-LARRY SMITH 770223-G.GROSS
      EXT  .ENTR
      ENT  TIMEX
*
*
*
*  ELAPSED TIME
*
*  THIS ROUTINE COMPUTES THE ABSOLUTE ELAPSED TIME BETWEEN
*  TWO SYSTEM 'EXEC' TIME CALLS.  IT MUST BE CALLED AS
*  FOLLOWS:
*
*     CALL EXEC(11,ISTRT) --- RETRIEVE FIRST TIME OF DAY
*          .
*          .
*          .
*    CALL EXEC(11,IENDT) --- RETRIEVE SECOND TIME OF DAY
*    CALL TIMEX(ISTRT,IENDT,IDIFT) --  COMPUTE ELAPSED
*                                      TIME IN IDIFT
*
*    WHERE:  ISTRT -  5 ELEMENT ARRAY FOR INITIAL TIME VALUES
*            IENDT -  5 ELEMENT ARRAY FOR TINAL TIME VALUES
*            IDIFT -  5 ELEMENT ARRAY FOR COMPUTED ELAPSED TIME
*    EACH 5 ELEMENT ARRAY IS PARTITIONED AS FOLLOWS:
*
*  ELEMENT   CONTENTS
*  -------   --------
*     1      NUMBER OF TENS OF MILLISECOND PERIODS
*     2      NUMBER OF SECONDS
*     3      NUMBER OF MINUTES
*     4      NUMBER OF HOURS
*     5      NUMERICAL DAY OF THE YEAR
```

Figure 5-6. TIMEX, Compute Time Difference

```
*
*
ISTRT NOP
IENDT NOP
IDIFT NOP
TIMEX NOP                 TIMEX ENTRY POINT
      JSB  .ENTR
      DEF  ISTRT

*     ....... COMPUTE 10'S OF MILLI-SECONDS ELAPSED TIME ...

      JSB  COMPT          COMPUTE THE RELATIVE DIFFERENCE.
      SSA,RSS             ADJUSTMENT NEEDED?
      JMP  SECS           NO.
      ADA  =D100          ADJUST.
      JSB  ADJ            PUT ADJUSTMENT IN IENDT ARRAY.

*     ....... COMPUTE SECONDS ELAPSED TIME ...

SECS  JSB  INC            BUMP ARRAY ADDRESS POINTERS.
      JSB  COMPT          COMPUTE THE RELATIVE DIFFERENCE.
      SSA,RSS             ADJUSTMENT NEEDED?
      JMP  MINS           NO.
      ADA  =D60           ADJUST.
      JSB  ADJ            PUT ADJUSTMENT IN IENDT ARRAY.

*     ....... COMPUTE MINUTES ELAPSED TIME ..

MINS  JSB  INC            BUMP ARRAY ADDRESS POINTERS.
      JSB  COMPT          COMPUTE THE RELATIVE DIFFERENCE.
      SSA,RSS             ADJUSTMENT NEEDED?
      JMP  HOURS          NO.
      ADA  =D60
      JSB  ADJ            PUT ADJUSTMENT IN IENDT ARRAY.

*     ....... COMPUTE HOURS ELAPSED TIME ...

HOURS JSB  INC            BUMP ARRAY ADDRESS POINTERS.
      JSB  COMPT          COMPUTE THE RELATIVE DIFFERENCE.
      SSA,RSS             ADJUSTMENT NEEDED?
      JMP  TIMEX,I        NO. RETURN TO CALLER.
      ADA  =D24           ADJUST.
      JSB  ADJ            PUT ADJUSTMENT IN IENDT ARRAY.
      JMP  TIMEX,I        RETURN TO CALLER

*     ....... UTILITY ROUTINES
ADJ   NOP
      STA  IDIFT,I        PUT ADJUSTED TUNIME BACK IN BUFFER
      LDA  IENDT
      INA
      STA  B              ADJUSTED ADDRESS.
      LDA  A,I            ADJUST THE TIME.
      ADA  =D-1
      STA  B,I            PUT BACK IN IENDT BUFFER.
      JMP  ADJ,I          RETURN.
COMPT NOP
      LDA  IENDT,I        GET ENDING TIME VALUE.
```

Figure 5-6. TIMEX, Compute Time Difference (Continued)

```
             LDB  ISTRT,I      GET STARTING TIME VALUE.
             CMB,INB
             ADA  B            COMPUTE RELATIVE DIFFERENCE.
             STA  IDIFT,I      PUT BACK IN DIFFERENCE ARRAY.
             JMP  COMPT,I      RETURN.
       INC   NOP
             ISZ  ISTRT        MOVE START TIME ARRAY ADDRESS POINTER
             ISZ  IENDT        MOVE ENDING TIME ARRAY ADDRESS POINTER
             ISZ  IDIFT        MOVE DIFF. TIME ARRAY ADDRESS POINTER
             JMP  INC,I        RETURN.

     *  ....... CONSTANT AND TEMPORARY STORAGE AREA ...

       A     EQU  0
       B     EQU  1

     *  ......... LITERALS ...
             END
```

Figure 5-6. TIMEX, Compute Time Difference (Continued)

Program PERF

Program PERF performs four operations.

1. It opens the file "HPIB" on LU 2 and obtains the system overhead values.

2. It obtains the initial and final times, and computes the difference using subroutine TIMEX shown in figure 5-6.

3. It obtains the initial and final counts, and computes the difference.

4. It calls the subroutine CMPUT to do some required computations so the results may be printed.

The user inserts FORTRAN statements into PERF to comprise the task in process to be measured.

In some situations, one user task may take much less than 10 milliseconds (the best time resolution available from the real-time system clock). For this reason, PERF gives the user an option of repeating the single task many times (then dividing the total time by the number of times the task was repeated) so that good resolution may be obtained. Figure 5-7 demonstrates this idea.

In the run statement,

```
:RU,PERF,A,B,C,D,E
```

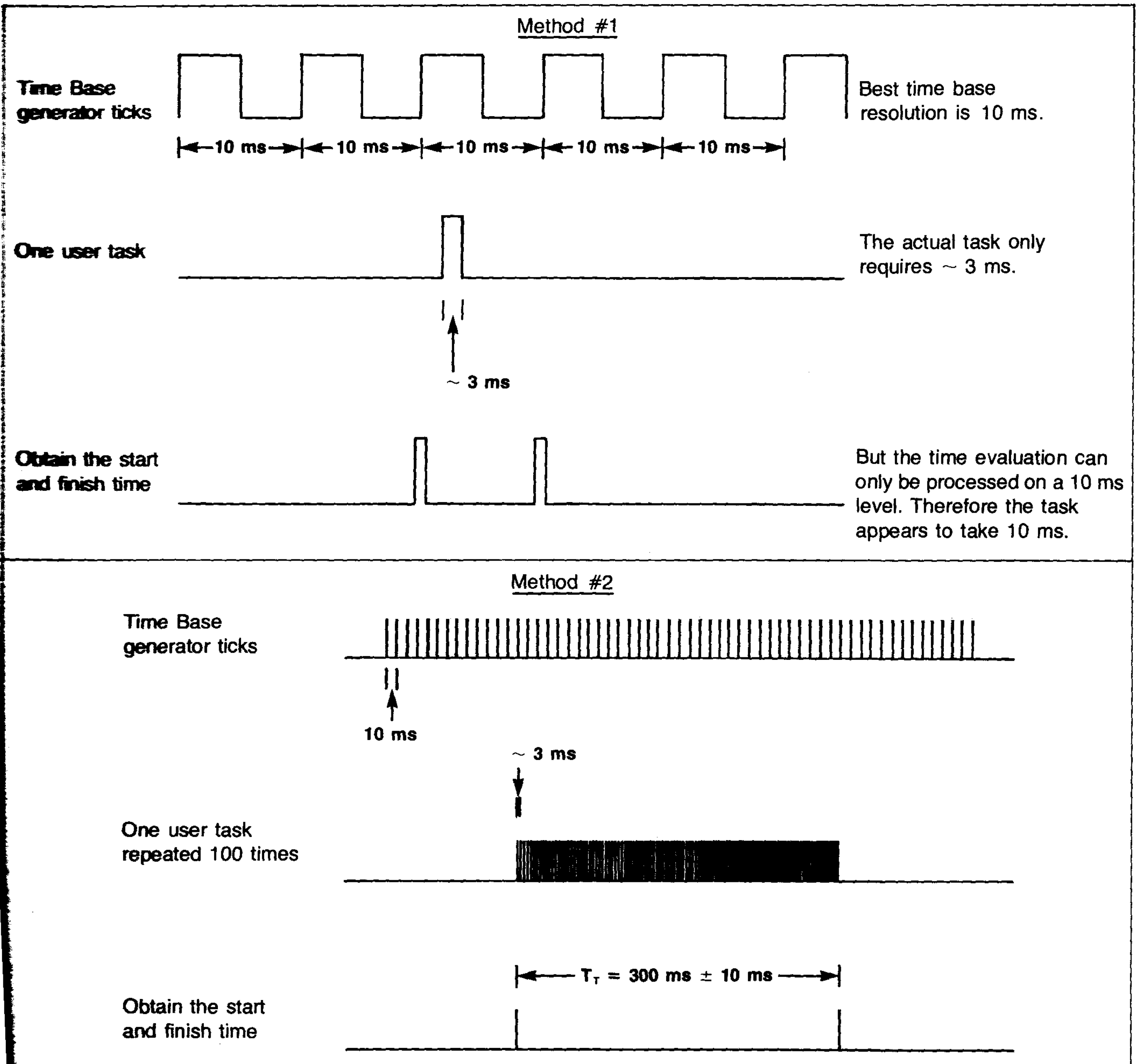
parameter A indicates where the measurement results are to be printed. (This is the LU of the computer output device.)

The parameter B is the HP-IB device LU.

The value C, in most cases, represents the number of measurements to be taken from the HP-IB device during the task. This parameter is user-definable.

Parameter D defines the number of times the task will be repeated to give good resolution. If this value is not supplied, it defaults to 100.

Parameter E is a flag which denotes whether system utilization during the task is to be measured. System utilization is recorded by scheduling MEMAA at a lower priority to count while the task in process is suspended (waiting for the HP-IB device). The number of counts per second which MEMAA is capable of performing is saved in file "HPIB". Therefore, the total time program MEMAA counts, during the task, can be computed to determine the CPU free time (TF, in figure 5-1) during the task.



$$T_T = 300\text{ ms} \pm 10\text{ ms}$$

$$T_u = \text{Single task} = \frac{300\text{ ms}}{100} \pm \frac{10\text{ ms}}{100} = 3\text{ ms} \pm .1\text{ ms}$$

$$= 3\text{ ms} \pm .1\text{ ms}$$

$$2.9\text{ ms} < T_u < 3.1\text{ ms}$$

Figure 5-7. Visual Description Time Tick Phenomena

Note that different total times will appear when parameter E is toggled, because TIDL (time spent by RTE in the idle loop) is incurred instead of TDISP (time spent by RTE dispatching other programs), as shown in Chapter 4.

For example, suppose the rate at which four ASCII characters can be sent to a particular HP-IB instrument needs to be measured. The FORTRAN statement which will do the job is:

```
      WRITE(IDLU,10)
   10 FORMAT("F1R1")
```

In this case, the above two statements should be added to program PERF between lines 63 and 70. When it is run, PERF will open the "HPIB" file in line 33 and obtain the system overhead data.

Next, program PERF initializes two memory locations, in system common (lines 39 and 40), in which program MEMAA saves its count values. Note that each MEMAA count represents the amount of time another program in the HP 1000 could be using for constructive work. This is a measurement of how much the system is utilized. Program MEMAA is scheduled, but it will not actually run until I/O begins with the HP-IB device.

Next, PERF obtains the current count values in system common and saves them in locations "IUT2" and "IUT1" (lines 55 and 56). It also obtains the current time and saves the value in "ISTRT" (line 58).

Line 63 contains the maximum loop value given in the run parameters. This value controls the resolution of the measurements as descibed earlier. User task statements begin in line 64.

Once the task is finished, program PERF again obtains the time and saves this value in "IENDT". The current MEMAA count is obtained from system common and the MEMAA counter is turned off.

The total number of MEMAA counts is obtained by subtracting "ISTR1" and "ISTR2" from "IFIN1" and "IFIN2". The subroutine CMPUT is used for this computation.

The value TT (figure 5-1) is equivalent to the "normalized" total test time. This value is obtained by taking the total time required for the test (including the processing time for system clock ticks) and normalizing out the system clock ticks. In other words, multiply the total time by the overhead factor (obtained from file "HPIB") and subtract this value from the total time.

If SEC represents the total time and OHED (from file "HPIB") is the overhead factor,

TNORM = SEC − (SEC*OHED)

See line 90 in figure 5-5.

The value TF (equation a in figure 5-1) is obtained by multiplying the number of counts MEMAA can perform per second (from file "HPIB") times the number of counts MEMAA was able to achieve during the task in process. (See line 91 in figure 5-5.) The value TC in figure 5-1 may then be computed.

The value "%UTL" (figure 5-1, equation c) is computed as the ratio of computer free time to the total normalized time for the task in process.

A file manger transfer file such as the one in figure 5-8 may be used to determine equation b in figure 5-1. The program PERF is continually rescheduled using incremental run parameters until enough values have been obtained to determine a line. The slope n and slope intercept TS may then be found.

System Overhead Procedure

Before the HP-IB performance measurements can begin, the operating system overhead must be determined. (Overhead usually ranges from 1 to 10%.)

NOTE

> When running the overhead programs, the system should be quiescent, i.e., no programs should be in the time list unless this is the condition which generally prevails. No other programs should be scheduled or actually running.
>
> The interrupt system is turned completely off for five minutes during the overhead procedure.

Two programs (MEMAA and FIGUR) are used to determine the operating system overhead. See "Program Loading Procedures" later in this section to load the programs into a particular RTE system.

```
:SV,4,,IH
:CA,4,9
:CN,4G,TO
:LL,4G
:DP,ENTER MESSAGE WITH ANNOTATE MESSAGE
:PAUSE
:CA,1,10
:IF,1G,GE,100,2
:AN,1G,READINGS _
:IF,,EQ,,1
:AN,1G, READINGS  _
:RU,PERF,4G,18,1G,1000,0
:AN,
:CA,1,1G,+,10
:IF,1G,LE,100,-8
:AN,
:CN,4G,TO
:CN,4G,TO
::
```

Figure 5-8. *PERF, A Transfer File for Doing Multiple Measurements

The program MEMAA runs in a special mode to obtain the system overhead. Otherwise, it runs in the normal mode. MEMAA is responsible for actually doing the measurement, and scheduling FIGUR to calculate the results. FIGUR creates a file "HPIB" and outputs the data to the file. This file contains:

a. counts/sec (for MEMAA outside the operating system).

b. percent/100 (system overhead).

The file "HPIB" is used later during the actual HP-IB performance measurements.

NOTE

> Program FIGUR will attempt to create a file "HPIB". If this file already exists, FIGUR will abort and the measurement will be lost. Therefore, purge all old "HPIB" files before beginning the operating system overhead procedure.

A stopwatch (or a clock with a second hand) is needed for this procedure. One free block on LU 2 is also required.

Figure 5-9 contains the details for executing the programs used for measuring the system overhead.

1. `*ON,WHZAT` (to determine that nothing but WHZAT is operating in the system)

   or `*ST` (in RTE-M)

2. Make sure that both MEMAA and FIGUR have ID segments in the RTE system, and that one free block of File Manager file space is available on system disc LU 2.

3. `*ON,MEMAA,,1` (At this point, the extend register on the front panel will begin blinking)

4. Set bit 15 (and quickly release) to start the first measurement. The output on the switch register will be an incremental octal count. The interrupt light on the front panel will be on.

   After setting bit 15 to start the measurement, wait 5 minutes and set bit 15 again (quickly release). The extend register will again begin blinking. The interrupt light will now be off.

5. Reset the stopwatch and again, set bit 15 (quickly release) and wait 5 minutes. When the time period is finished, set bit 15 again. Program MEMAA will now schedule FIGUR and pass it the results.

Figure 5-9. Overhead Procedure

Program Loading Procedures

General

The system must be generated with at least four words of system common (foreground common in RTE-II, RTE-III, or RTE-IV).

RTE-M

MEMAA and PERF are relocated on-line (accessing system common) into a partition, as specified by the snapshot. Similiarly FIGUR is relocated, but it need not access common. MEMAA should have a priority of 32767. PERF and FIGUR should each have a priority of 99.

For example, a LOADR file might look like:

```
*LO,RTMLD
*RU,RTMLD,AN,SF,IL,1,3
```

The answer file might be,

```
TR,SNAP
OUTPUT ON MEMAA
MAP MODULES ON 1
REL %MEMAA
SEARCH %MSYLB
SEARCH %RLIB1
SEARCH %RLIB2
END
```

RTE-II

MEMAA must be loaded during generation as a memory-resident program accessing foreground common (priority = 32767).

FIGUR may be loaded on-line (background or foreground, no common needed).

PERF may be loaded on-line as background disc-resident, accessing foreground common (priority = 99). For example,

```
:RU,LOADR
/LOADR:   OP,RC
/LOADR:   RE,%PERF
   :
   :
/LOADR:   EN
```

RTE-III and RTE-IV

MEMAA may be loaded either during generation as a memory-resident program accessing system common, or it may be loaded on-line as a foreground disc-resident program accessing foreground common (the common access will default to those above). It should have a priority of 99.

FIGUR may be loaded on-line (background or foreground, no common needed).

PERF is loaded on-line as a background disc-resident program, accessing reverse common (see RTE-II), having a priority of 99.

Assigning HP-IB Logical Unit Numbers

Chapter 6

The logical unit (LU), — Equipment Table (EQT),— device address (DA) — mapping scheme is a generalized method within the RTE system for maintaining maximum flexibility between the computer and its external peripherals.[1] Although several RTE manuals discuss this concept, the assignment procedures sometimes appear cryptic to the new user.

In most situations an interactive user program performs input, output, and — in this series of application notes — it also interacts with an HP-IB instrument. In the Real Time Executive, numeric values called logical unit numbers are assigned to external peripherals to make referencing them an easier task. A line printer may be assigned LU "6" or a digital voltmeter LU "27," for example.

The number assigned to the peripheral is arbitrary, but the user must know it to reference the terminal, line printer, or HP-IB instrument. Therefore, for this series of application notes, the input terminal will be denoted "ILU," the list device "ILST," and the instrument "IDLU."

The purpose of this chapter is to broaden the reader's knowledge of logical units and to address some details of LU mapping. A series of user subroutines will show a method for identifying ILU, ILST, and IDLU for the user's program.

All of the programs and subroutines shown may be obtained in one package from the contributed library (part number 22683-13346).

Figure 6-1 shows a typical user program having references to two function subprograms "INPRM" and "GTDLU". The purpose of these routines is to obtain ILU, ILST, and IDLU for the user program. They can make standard procedures — such as obtaining LUs for the user's input terminal, the standard output peripheral (for listings,etc.), and the HP-IB instrument LU — appear much simpler to the novice.

The subroutine INPRM (figure 6-2) tries to find ILST, and IDLU in the user run statement,

```
:RU,TDLU,6,20
```

for example. If these parameters are not supplied, the user just types,

```
:RU,TDLU
```

then the subroutine GTDLU (figure 6-3) must be called.

The subprogram "GTDLU" assumes that the user is armed only with the HP-IB model number of the instrument, and that he knows whether multiple instruments having the same model number are connected to the HP 1000.

Subroutine INPRM

The function subprogram "INPRM" can be defined as any user-written subroutine which is capable of obtaining the LU for the user's input terminal (ILU, a numeric value), the LU number where computer output will appear (ILST, a numeric value), and the LU number of the HP-IB instrument (IDLU, a numeric value), and putting each of them in the user program area.

In an HP 1000 environment, the on-line user (at an input terminal) can supply these values in the RUN statement for his program. For example,

```
:RU,PROGA,ILST,IDLU
```

or

```
:RU,C3582,6,20
```

```
FTN4,L
      PROGRAM TDLU
      INTEGER       INPRM,GTDLU
      COMMON        ILU,ILST,IDLU
      COMMON        /IEROR/YES,NO
      IF(INPRM(ID).EQ.YES) GO TO 20
      WRITE(ILU,10)
   10 FORMAT(/"ENTER 'MODEL NUMBER,INSTRUMENT NUMBER' : _")
      READ(ILU,)RMOD,IN
      IF(GTDLU(RMOD,IN,IDLU).EQ.NO)STOP
   20 WRITE(ILU,30)IDLU
   30 FORMAT("IDLU= "I3)
      STOP
      END
```

Figure 6-1. User Program Using INPRM and GTDLU

---

[1]*Appendix E of the HP-IB Users Guide (part number 59310-90064) discusses the LU concept and how it is used.*

will allow the user program to obtain the value "6" for the list peripheral (ILST) and the value "20" for the HP-IB device (IDLU). Note that the user input terminal LU can be obtained automatically in RTE.[2] (It need not be specified in the RUN statement.) The user simply includes the function call to LOGLU in his FORTRAN program:

```
ILU=LOGLU(ID)
```

The variable ID is a dummy parameter required for a function call.

The function INPRM, shown in figure 6-2, is used in almost every program in this series of application notes. Inside INPRM it was assumed that ILST and IDLU were supplied in the RUN statement. If ILST is not specified, it defaults to ILU.[2]

On return to the main program, INPRM returns either "YES" or "NO." The value "NO" is returned when IDLU was found to be zero (i.e., the device LU was not supplied in the RUN statement).

```
FTN4,L
      INTEGER FUNCTION INPRM(ID),11-29-78 (GWG) RUN PRM FOR HP-IB
      INTEGER        ISTRNG(40),OSTRNG(10),STRT
      COMMON         ILU,ILST,IDLU
C
C 'INPRM' GETS:
C
C   A. THE INPUT LOGICAL UNIT (INTERACTIVE TERMINAL).
C   B. THE LIST LOGICAL UNIT FROM PARAMETER ONE (IT
C      SETS THE LIST LU EQUAL TO THE INPUT LU IF THE
C      LIST LU IS 0).
C   C. THE DEVICE LOGICAL UNIT(INPRM CHECKS TO SEE
C      IF IDLU IS NON-ZERO. IF NOT INPRM IS SET TO
C      '2HNO').
C
      INPRM=2HNO
      ILU=LOGLU(ID)
      CALL GETST(ISTRNG,-80,RTNCLN)
```

Documented in the RTE IV Programmer's Manual, 92067-90001.

```
C
      STRT=1
      DO 600 I=1,2
      IF(NAMR(OSTRNG,ISTRNG,RTNCLN,STRT))700,100
```

Documented in the Session Monitor Reference Manual, 09570-93022.

```
  100 ITYP=IAND(OSTRNG(4),3B)
      IF(I.EQ.1)GO TO 200
      IF(ITYP.NE.1) RETURN
      IDLU=OSTRNG
      GO TO 600
  200 ILST=OSTRNG
      IF(ITYP.EQ.0) ILST=ILU
  600 CONTINUE
  700 IF(IDLU.GT.0)INPRM=2HYE
      RETURN
      END
```

Figure 6-2. One Example INPRM

[2] See the RTE IV programming manual (92067-90001), function subprogram "LOGLU," for obtaining the user input terminal, ILU, automatically.

```
FTN4,L
C
C
      INTEGER FUNCTION GTDLU(MODN,INN,IDLU),02-15-79 (GWG) GET DEVICE LU
C GTDLU RETURNS IDLU
C              OR NO ON AN FMP ERROR
C              OR NO WHEN NO LU IS AVAILABLE
C
C HOW TO  USE IT (WITH INPRM) AT THE BEGINNING OF YOUR PROGRAM:
C
C FTN4,L
C               PROGRAM X(3),------------
C               INTEGER INPRM,GTDLU
C               COMMON ILU,ILST,IDLU
C               COMMON /IEROR/YES,NO
C               IF(INPRM(ID).EQ.YES) GO TO 20
C               WRITE(ILU,10)
C            20 FORMAT(/"ENTER 'MODEL NUMBER,INSTRUMENT NUMBER' : _")
C               READ(ILU,)RMOD,IN
C               IF(GTDLU(RMOD,IN,IDLU).EQ.NO)STOP
C                      :
C                      :
C               END
C               END$
C
      INTEGER          EQT,AD,NO,YES,IREG(2),IBFR(80),
     &                 EQTBL(64),EQN,IDMY(3),
     &                 AVAIL,AVEQT,ASNLU,GIBLU,GDATA,RCORD,GTADD
      REAL             MODN
      EQUIVALENCE      (IREG,REG,IA),(IREG(2),IB)
      COMMON           ILU,ILST,IDL
      COMMON           /IEROR/YES,NO
      COMMON           /IEQS/EQTBL,EQN
C IF THE DEVICE EQT AND ADDRESS IS NEEDED BY THE MAIN...
      COMMON           /INSTR/IDMY,EQT,AD
      GTDLU=NO
      IF(INN.EQ.0)INN=1
C GET A TABLE OF ALL HP-IB EQTS IN THE SYSTEM
      IF(AVEQT(EQTBL,EQN).EQ.YES)GO TO 200
      RETURN
C GET DATA FROM FILE IF THERE
  200 IF(GDATA(MODN,INN,EQT,AD).EQ.YES)GO TO 99
C DATA NOT IN FILE. TRACK DOWN THE INSTRUMENT ADDRESS
      IF(GTADD(EQT,AD).EQ.YES) GO TO 97
C DATA NOT AVAILABLE FROM THE BUS. ASK FOR IT.
    5 WRITE(ILU,10)MODN,INN
   10 FORMAT(/"                DATA WAS NOT FOUND FOR HP "F5.0,
     &        " NUMBER "I2".",
     &//,     "PLEASE ENTER (OR 'NO'):",
     &//,"    'MODEL, INSTRUMENT, EQT, DEVICE ADDRESS'"/,
     &/"           ")
      REG= EXEC(1,ILU+400B,IBFR,50)
      IF(IBFR.EQ.NO) GO TO 108
      IBFR(IB+1)=2H,
      L=IB+1
```

Figure 6-3. Determine Device LU If Possible

```
      CALL CNVRT(IBFR,L,MODN,INN,EQT,AD)
      DO 400 I=1,EQN-1
      IF(EQT.EQ.EQTBL(I))GO TO 98
  400 CONTINUE
      WRITE(ILU,170)
  170 FORMAT(/"GTDLU: YOUR EQT NUMBER IS INVALID."/)
      RETURN
C CONVERT THE INFO TO ASCII
   97 L=0
      CALL CNVRT(IBFR,L,MODN,INN,EQT,AD)
C RECORD THE INFORMATION IN THE 'HPIBD' FILE
   98 IF(RCORD(IBFR,L).EQ.NO)GO TO 108
C SEE IF THE LU ALREADY EXISTS
   99 IF(GIBLU(IDLU,EQT,AD).EQ.YES) GO TO 110
C GET AN LU ASSIGNED TO ZERO
      IF(AVAIL(IDLU).EQ.YES)GO TO 100
      RETURN
C ASSIGN IT TO THE SPECIFIED DEVICE
  100 IF(ASNLU(IDLU,EQT,AD).EQ.YES)GO TO 110
      RETURN
  108 WRITE(ILU,109)
  109 FORMAT("GTDLU: HP-IB DATA WAS NOT FOUND.",
     &       " NO FILE UPDATE WAS PERFORMED."/)
      RETURN
  110 GTDLU=YES
      RETURN
      END
```

Figure 6-3. Determine Device LU If Possible (Continued)

Subprogram GTDLU

One major problem for the novice is that ILU, ILST, and IDLU must be identified. At worst, ILST and IDLU must be assigned dynamically, on-line.[3]

HP-IB device addresses can also be a problem. The address switches must be physically located. Sometimes they are readily available on the rear panel of the instrument. However, some address switches reside inside the instrument and it must be dismantled for address adjustments. When no documentation is available, these instruments must be dismantled for correct address verification.

Reading direct binary values from address switches and mentally converting this number to octal or decimal, is a tedious task. Many instruments don't organize these switches in a readable binary format.

The function "GTDLU" can be defined as any user written function subprogram which can automatically determine IDLU for a "new HP-IB instrument"[4] from its model number (on the front panel) and its instrument number.[5]

The information needed to determine IDLU can be obtained from one of two places:

1. A disc file containing information about instrument model numbers, device numbers, their corresponding EQTS and device addresses.

2. By actually interrogating the HP-IB instrument on the bus for its listen address.

The function GTDLU makes references to several other functions during it's process of finding the instrument, IDLU, as shown in figure 6-4.

---

[3] *The term "assigned" means that a correspondence must be mapped from the HP 1000 LU through the EQT to an HP-IB device address.*

[4] *The term "new HP-IB instrument" means that only one HP-IB instrument may be connected and programatically added at a time (i.e., only one HP-IB device address can be identified by interrogation at a time).*

[5] *The instrument number need only be specified if more than one instrument with the same model number resides in the system.*

FUNCTION OR SUBROUTINE	DESCRIPTION
GTDLU(MODN,INN,IDLU)	The model number, "MODN" and instrument number "INN" are passed to GTDLU. The instrument number "IDLU" is returned to the caller.
GDATA(MNUM,NUM,EQT,AD)	Obtains Instrument data from a memory table created by function FLASS. The values MNUM, NUM, EQT, and AD are passed to GDATA. When MNUM and NUM are passed as zero parameters, but EQT and AD non-zero, then MNUM and NUM are returned to the caller. When EQT and AD are passed as zero parameters, but MNUM and NUM are non-zero, then EQT and AD are returned to the caller.
GTADD(EQT,IAD)	Interrogates the bus for the HP-IB instrument listen address. EQT and IAD are passed from GTADD back to its caller.
RCORD(IBFR,L)	Records new instrument data in file "HPIBD:GG:26". (The file "namr" may be changed by modifying block data common.) IBFR and L are passed to RCORD.
AVAIL(LUN)	Determines the first available LU in the system. AVAIL passes LUN back to its caller.
ASNLU(LU,EQT,AD)	Uses the operator routine MESSS to assign an HP-IB LU. LU, EQT and AD are passed to ASNLU.
AVEQT(EQTBL,TBLN)	Obtains a complete list of the HP-IB EQTs in the system and saves them in a memory table EQTBL. EQTBL and TBLN are passed from AVEQT back to its caller.
GIBLU(LUX,EQT,IAD)	Determines whether an HP-IB instrument has an assigned LU. EQT and IAD are passed to GIBLU. LUX is passed from GIBLU back to its caller.
FLASS(MNUM,NUM,EQT,AD,ISTRT)	Interrogates file "HPIBD:GG:26", creates a memory LU table, and supplies binary data to other functions. ISTRT is passed to FLASS and determines whether the data table should be redone or the next record should be read from the data table. MNUM, NUM, EQT, and AD are returned from FLASS to its caller.
CNVRT(IBFR,L,MODN,IN,EQT,AD)	Uses the FORTRAN formatter for ASCII to Binary conversion and vice-versa. IBFR and L are supplied to the subroutine and MODN, IN, EQT, and AD are returned to its caller. Alternately, MODN, IN, EQT and AD can be supplied then IBFR and L returned.
ERROR(NAM,IERR)	Prints FMP errors. NAM and IERR are supplied to the function.

Figure 6-4. Subroutines Associated with GTDLU

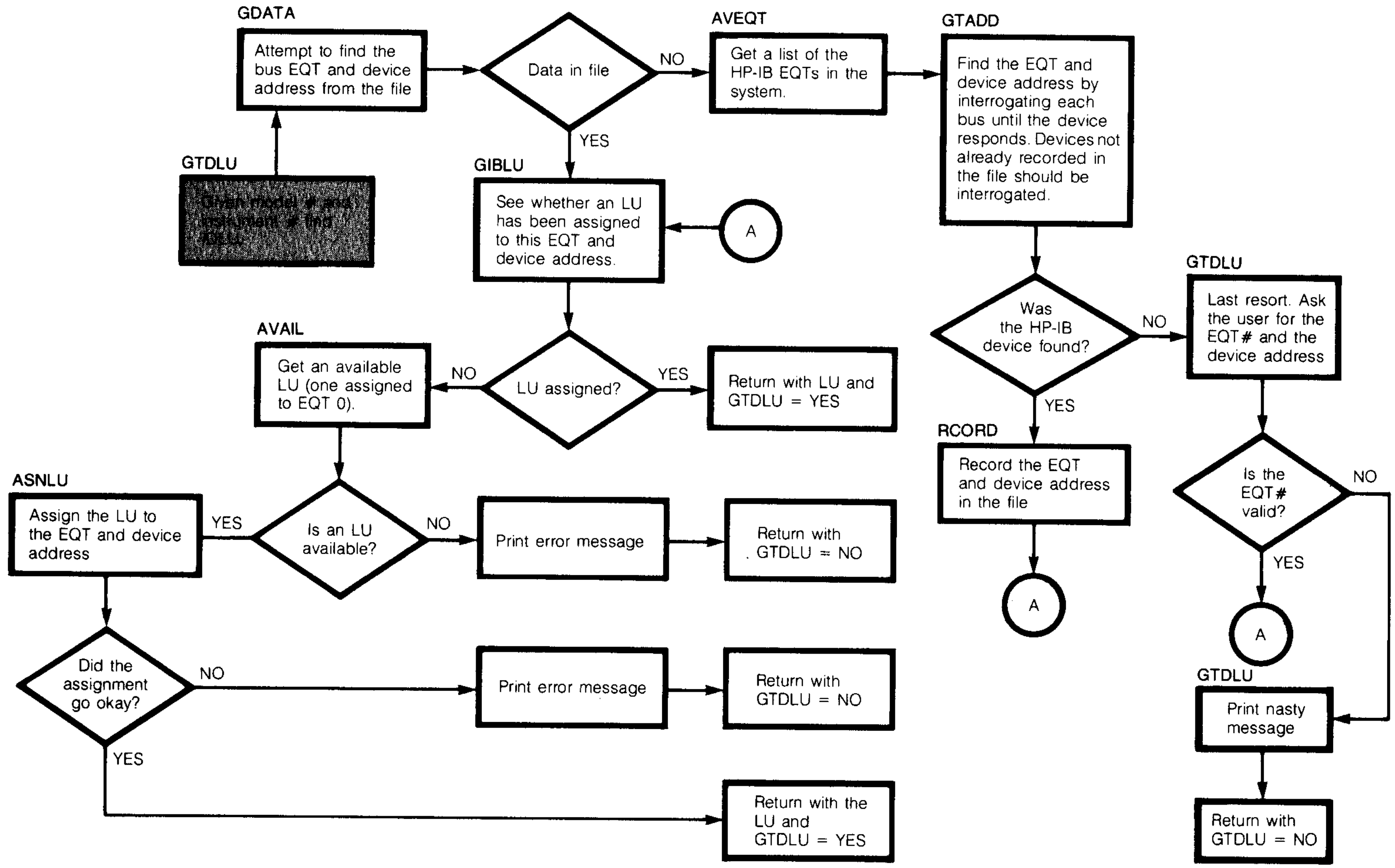


Figure 6-5. Flowchart Showing the Steps Required for HP-IB LU Determination Using GTDLU

Shown in figure 6-5 is a flowchart of GTDLU depicting the sequence of operations required to find an LU given an HP-IB model and instrument number.

Generally, the flowchart shows that the LU must be determined from a disc file, or by interrogation, or as a last resort, by asking the user.

The subprogram GDATA does the needed comparisons of file information with that supplied by GTDLU, but actual file access is handled by subprogram FLASS. When the HP-IB LU cannot be found from the file of current HP-IB devices, subprogram GTADD must be called to interrogate each HP-IB bus in the system. Subprogram GTADD can be a potentially sophisticated process requiring the functions described in the following sections. Some of these subroutines use the function IGET to interrogate the system area for device reference table, and EQT table information.[6]

```
      INTEGER FUNCTION GDATA(MNUM,NUM,EQT,AD),02-15-79 (GWG) FILE
C GDATA RETURNS YES WHEN DATA OBTAINED SUCCESSFULY FROM FILE HPIBD
C         RETURNS NO  WHEN NO DATA IS AVAILABLE
C         OUTPUTS AN ERROR MESSAGE WHEN AN FMP ERROR OCCURS
C         RETURNS EQTF AND ADF WHEN MNUM.NE.0 AND NUM.NE.0
C         RETURNS MFNUM AND NUMF WHEN MNUM.EQ.0 AND NUM.EQ.0
C
      INTEGER     GDATA,NUM,EQT,AD,YES,NO,FLASS,
     &            EQTF,ADF
      REAL        MNUM,MFNUM
      COMMON      ILU,ILST,IDLU
      COMMON      /IEROR/YES,NO
C
      GDATA=NO
      ISTRT=-1
C GET DATA FROM FILE OR TABLE
   10 IF(FLASS(MFNUM,NUMF,EQTF,ADF,ISTRT).EQ.NO)RETURN
C ATTEMPTING TO FIND EQT AND ADRESS INFO?
      IF(       ABS(MNUM-0).LT..01
     &    .AND. NUM.EQ.0            )GO TO 20
C NO, ATTEMPTING TO FIND MODEL NO. AND INSTRUMENT NO. INFO.
      IF(       ABS(MNUM-MFNUM).GT.(.01)
     &    .OR. NUM .NE.NUMF )GO TO 10
C RETURN EQT NO. AND ADDRESS INFO.
      EQT=EQTF
      AD=ADF
      GO TO 25
   20 IF(      EQT.NE.EQTF
     &    .OR. AD.NE.ADF   )GO TO 10
C RETURN MOD NO. AND INST. NO. INFO.
      MNUM=MFNUM
      NUM=NUMF
   25 GDATA=YES
      RETURN
      END
```

Figure 6-6. Function GDATA Obtains File Data From FLASS Table

---

[6] *This function subprogram is documented in the DOS/RTE Relocatable Library Reference Manual (part number 24998-90001).*

```
      INTEGER FUNCTION GTADD(EQT,IAD),02-20-79 (GWG) GET THAT ADD!
      INTEGER        EQT,AD,IPBUF(2),GDATA,GIBLU,
     &               IEQT5,GTADD,NO,YES,EQTBL(64),EQN,
     &               RCORD,AVAIL,ASNLU,IBFR(80)
      REAL           MNUM
      COMMON         ILU,ILST,IDLU
      COMMON         /IEROR/YES,NO
      COMMON         /INSTR/MNUM,NUM
      COMMON         /IEQS/EQTBL,EQN
      GTADD=NO
      IFLG=0
      IEQT5=0
      DO 100 I=1,EQN-1
C GET THE BUS LU INFO FROM THE 'HPIBD' FILE
      IF(GDATA(59310.,I,IEQT,AD).EQ.YES)GO TO 20
C NO BUS DATA SO PUT IT THERE
      L=0
      CALL CNVRT(IBFR,L,59310.,I,EQTBL(I),0)
      IF(RCORD(IBFR,L).EQ.NO)RETURN
      IEQT=EQTBL(I)
C MAKE SURE AN LU IS ASSIGNED TO THE BUS
   20 IF(GIBLU(LU,IEQT,0).EQ.YES)GO TO 30
C NO LU ASSIGNED TO THE BUS GET AN LU
      IF(AVAIL(LU).EQ.NO)RETURN
C ASSIGN A BUS LU
      IF(ASNLU(LU,EQTBL(I),0).EQ.NO)RETURN
   30 IPBUF(1)=77B256+137B
      DO 100 J=41B,76B
      IDAD=J-40B
C CHECK DATA FILE FOR KNOWN LU
      DMOD=0.
      NMB=0
      IF(GDATA(DMOD,NMB,EQTBL(I),IDAD).EQ.NO)GO TO 40
      GO TO 100
C LU UNKNOWN GO OUT AND FIND IT
   40 IPBUF(2)=J256+40B
C SEND THE LISTEN ADDRESS OF THE DEVICE
      CALL EXEC(2,012100B+LU,0,0,IPBUF,-3)
C CHECK TO SEE IF THE DEVICE ACCEPTED THE DATA
      CALL EXEC(3,600B+LU)
      CALL EXEC(13,LU,IEQT5)
      IF(IAND(IEQT5,40B).NE.0) GO TO 100
      IF(IFLG.EQ.1)GO TO 50
      IFLG=1
      GTADD=YES
      IAD=IDAD
      EQT=EQTBL(I)
      GO TO 100
   50 WRITE(ILU,60)IDAD,EQTBL(I)
   60 FORMAT(/"GTADD: UNKNOWN HP-IB INSTRUMENT WITH ADDRESS "K2"B,",
     &        " IS PRESENT ON BUS EQT "I2"."/)
  100 CONTINUE
      RETURN
      END
```

Figure 6-7. Interrogate the Bus for the Device Address

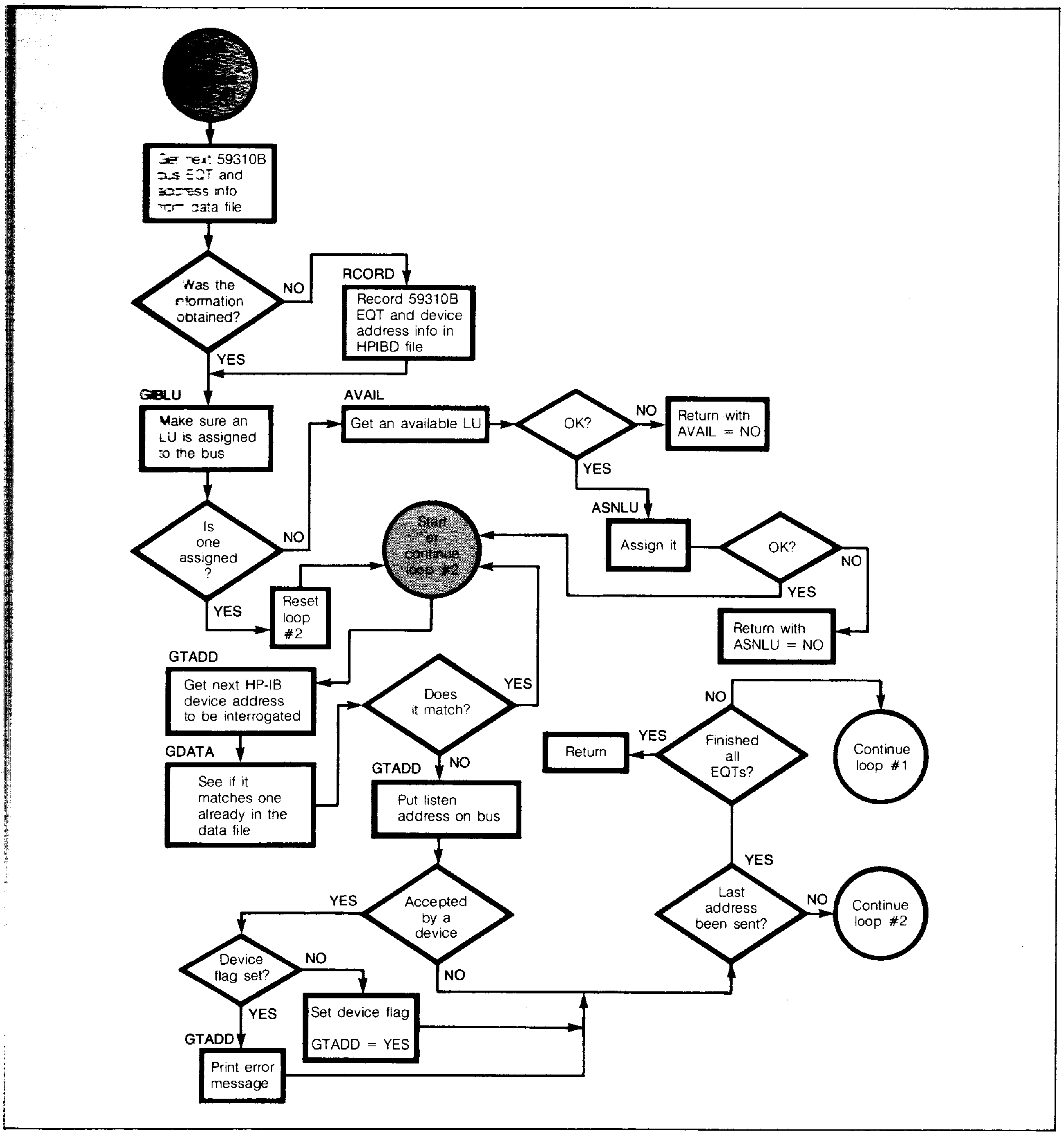


Figure 6-8. Flowchart of GTADD

Function GTADD

The purpose of this function is to interrogate the HP-IB for the device address. Several operations may be required before the bus EQTS in the system can be interrogated.

As shown in figure 6-8, the LU of each bus (with device address 0) must be used for interrogation. Because I/O requests of a generalized nature must be initiated, each bus must have a logical unit number assigned with a device address of zero. This requires one or more entries in the HPIBD:GG:26 data file. Note that the file namr may be changed by modifying the BLOCK DATA COMMON statement shown in figure 6-18. The file is created or interrogated for 59310B EQT/device address entries. Once the proper entries have been made and the zero address LU's have been assigned, device interrogation can begin.

Note that function GTADD interrogates each bus for device addresses which are not found in the HPIBD data file. The limitation of GTADD is that the HP-IB instrument whose LU is to be found has an internal address which is distinct from all others in the HPIBD file and distinct from those on the bus. The latter can be assured by disconnecting the other HP-IB devices during the device address interrogation phase.

A device is interrogated by sending untalk, unlisten, and a device listen address. The function then checks the 59310B I/O card status (lines 40 and 41) to determine whether the device accepted the data (the NDAC line was set LO). The success of this method hinges on the fact that a device will handshake data only after receiving its talk or listen address.[7] This is the normal mode of protocol for most HP-IB instruments.

All listen addresses are sent on each bus, even though a valid listener is found. If more than one address is accepted, a message stating the predicament is printed on ILU. The function GTADD does not return "NO" in this situation however. The first valid address found is selected and used in this situation.

```
      INTEGER FUNCTION RCORD(IBFR,L),02-20-79 (GWG) ADD FILE
C RCORD RETURNS YES WHEN NEW INFO RECORDED SUCCESSFULLY IN FILE
C        RETURNS NO  WHEN NO UPDATE IS MADE
C        ERROR MESSAGES ARE PRINTED AS FMP ERRORS
      INTEGER      IBFX(80),RCORD,YES,NO,IDCB(144),
     &             OPEN,CLOSE,READF,WRITF,CREAT,HPIBD(3),
     &             FLASS,NAM(3),ERROR
      COMMON       ILU,ILST,IDLU
      COMMON       /IEROR/YES,NO
      COMMON       /IBFIL/HPIBD,ISEC,ICR,ISZ,IDCB
C RCORD
      DATA         NAM/2HRC,2HOR,2HD /
      RCORD=NO
C     OPEN THE 'HPIBD' FILE
    4 IF(OPEN(IDCB,IERR,HPIBD,2,ISEC,ICR).LT.0) GO TO 500
   10 IF(READF(IDCB,IERR,IBFX,80,LEN).LT.0) GO TO 1000
      IF(LEN.GE.0) GO TO 10
      IF(WRITF(IDCB,IERR,IBFR,L).LT.0) GO TO 1000
   30 CALL CLOSE(IDCB,IERR)
      ISTRT=0
      IF(FLASS(A,MM,NN,LL,ISTRT).EQ.NO)RETURN
      RCORD=YES
      RETURN
  500 IF(CREAT(IDCB,IERR,HPIBD,ISZ,3,ISEC,ICR).LT.0)GO TO 1000
      CALL CLOSE(IDCB,IERR)
      GO TO 4
 1000 IDUM=ERROR(NAM,IERR)
      GO TO 30
      END
```

Figure 6-9. Record New Information in HP-IB Data File

[7] It should be noted that some devices, i.e., bus extenders, etc., do not follow this method of protocol.

Function RCORD

This function is passed an ASCII buffer and a length. The purpose of the function is to record the instrument model information in the HPIBD:GG:26 data file. Note that if the data file is non-existent, the subprogram attempts to create one given the file "NAMR" in block data common (see "IBFIL" in line 10).

Each time RCORD is called, it opens (or creates) the data file and adds one record to the end of the current list. If this is accomplished successfully, subprogram FLASS is called to reinitialize the memory file containing current instrument data.

Function AVAIL

This subroutine obtains the value of the first logical unit assigned to EQT 0 in the system. By using subroutine IGET to determine the maximum number of LUS in the system, and the starting point of the device reference table, the available LU can be found. If none are available, a message is printed on ILU and AVAIL returns "NO" to the caller.

Function ASNLU

This subroutine assigns an LU given the three required parameters, LU, EQT and AD (the device address). The subroutine CNUMD is used to convert the binary values to ASCII. The system message processor then proceses these parameters for the new LU assignment. If an error occurs in the "MESSS" processor, the error is printed, another message is printed by ASNLU, and the function returns "NO" to the caller.

```
      INTEGER FUNCTION AVAIL(LUN),02-20-79 (GWG) GET AVAILABLE LU
      INTEGER   EQT,LUMAX,IDRT,LUN,NO,YES,AVAIL
      COMMON    ILU,ILST,IDLU
      COMMON    /IEROR/YES,NO
      AVAIL=NO
      LUMAX=IGET(1653B)
      IDRT=IGET(1652B)
      LUN=0
      DO 10 I=0,LUMAX-1
      EQT=IAND(IGET(IDRT+I),77B)
      IF(EQT.NE.0) GO TO 10
      LUN=I+1
      AVAIL=YES
      RETURN
   10 CONTINUE
      WRITE(ILU,20)
   20 FORMAT(/"AVAIL: NO AVAILABLE LUS IN THIS SYSTEM."/)
      RETURN
      END
```

Figure 6-10. Finds the First Available LU in the System

```
      INTEGER FUNCTION ASNLU(LU,EQT,AD),02-20-79 (GWG) ASSIGN AN LU
      INTEGER LU,EQT,AD,ICM(11),YES,NO
      COMMON ILU,ILST,IDLU
      COMMON /IEROR/YES,NO
      ASNLU=YES
      ICM=2HLU
      CALL CNUMD(LU,ICM(2))
      ICM(2)=2H,
      CALL CNUMD(EQT,ICM(5))
      ICM(5)=2H,
      CALL CNUMD(AD,ICM(8))
      ICM(8)=2H,
      IANS=MESSS(ICM,20)
      IF(IANS.GE.0)GO TO 100
      CALL EXEC(2,ILU,ICM,IANS)
      WRITE(ILU,10)
   10 FORMAT(/"ASNLU: LU ASSIGNMENT DID NOT FUNCTION PROPERLY."/)
      ASNLU=NO
  100 RETURN
      END
```

Documented in the RTE IV Programmer's Manual (92067-90001).

Figure 6-11. Assigns an LU to an EQT and Address

```
      INTEGER FUNCTION AVEQT(EQTBL,TBLN),02-20-79 (GWG) TABLE
      INTEGER   EQTBL(1),EQN,MXEQS,IEQT,NO,YES,AVEQT,EQT,CONTA,
     &          INITA,INITE,CONTE,TBLN
      COMMON    ILU,ILST,IDLU
      COMMON    /IEROR/YES,NO
      AVEQT=NO
      MXEQS=IGET(1651B)
      IEQT=IGET(1650B)
      TBLN=1
C GET HP-IB DRIVER INITIATOR/CONTINUATOR ADDRESSES
      CALL CEQT(INITA,CONTA)
      DO 10 EQN=1,MXEQS
      EQTBL(EQN)=0
      EQT=(EQN-1)15+IEQT
      ISTAT=IGET(EQT+4)
      INITE=IGET(EQT+1)
      CONTE=IGET(EQT+2)
C MAKE SURE THIS IS NOT A SPOOL EQT
      IF(         IAND(ISTAT,37400B)/256.NE.37B
     &   .OR.     INITE.NE.INITA
     &   .OR.     CONTE.NE.CONTA                ) GO TO 10
      EQTBL(TBLN)=EQN
      TBLN=TBLN+1
   10 CONTINUE
      IF(EQTBL.EQ.0) GO TO 15
      AVEQT=YES
      RETURN
   15 WRITE(ILU,20)
   20 FORMAT(/"AVEQT: NO HP-IB EQTS IN THIS SYSTEM."/)
      RETURN
      END
```

Figure 6-12. Builds a Memory Table of HP-IB EQTs in the System

Function AVEQT

This function uses the subprogram "IGET" to track through the system EQT area. During the process, the HP-IB EQT values are saved in table EQTBL.

If no HP-IB EQTS are found AVEQT prints an error message on ILU and returns "NO" to the caller.

A special note should be made concerning the reference to CEQT from subroutine AVEQT. In some RTE environments, in particular when the Spool Monitor program is being used, a spool EQT can, at times, appear very similar to an HP-IB EQT. One way to verify that the EQT being referenced is indeed an HP-IB EQT, is to obtain the entry point address of the HP-IB driver. Locations 2 and 3 in any EQT contain the entry point addresses of the I/O driver for that EQT. By verifying the actual driver entry points against those in the EQT the HP-IB EQT's in the system can be identified. The limitation of this method is that Table Area II (RTE IV) must be accessed to determine the system entry point addresses for the HP-IB driver. Therefore the user program must be declared Type 2 or Type 3 (i.e., it cannot be declared Type 4, a large background program). See "Program AUTLU" later in this section for more details.

```
ASMB,L
      NAM HDVR,7 09-27-78 (GWG) CHECK FOR SPOOL EQT

 THIS SUBROUTINE RETURNS THE HP-IB DRIVER ADDRESSS FOR THE
 INITIATOR AND THE CONTINUATOR. THE INTENDED PURPOSE IS TO
 CHECK THESE ADDRESSES WITH THOSE CURRENTLY BEING LOOKED AT
 IN THE CURRENTLY ADDRESSED EQT. A SPOOL DRIVER EQT (WHICH
 LOOKS LIKE AN HP-IB) CAN THEN BE DETECTED.

      CALL CEQT(INIT,CONT)

      ENT CEQT
      EXT I.37,C.37,.ENTR
ARGI  NOP
ARGC  NOP
CEQT  NOP
      JSB .ENTR    SET UP FOR RETURN PARMS.
      DEF ARGI
      LDA INIT     GET THE HP-IB INITIATOR ADDRESS.
      STA ARGI,I
      LDA CONT     GET HP-IB CONTINUATOR ADDRESS.
      STA ARGC,I
      JMP CEQT,I   RETURN THEM TO THE CALLER.
INIT  DEF I.37+0
CONT  DEF C.37+0
      END
```

Figure 6-13. Check for Valid HP-IB EQT

Function GIBLU

Given the EQT number and device address, this function determines whether an LU is currently mapped to these values. GIBLU returns "YES" to the caller only when an appropriate LU has been found.

Function FLASS

Initially, this subroutine opens the HPIBD:GG:UU data file and reads the data into a table in memory. Each consecutive call obtains the next record from the table. The memory table was created to decrease access time.

```
      INTEGER FUNCTION GIBLU(LUX,EQT,IAD),02-20-79 (GWG) ASSIGNED
C TRACK DOWN AN ASSIGNED LU IN THE SYSTEM GIVEN EQT AND AD
C GIBLU RETURNS NO WHEN NO LU IS FOUND
      INTEGER         NO,YES,GIBLU,GTEQT,EQT,IAD
      COMMON          ILU,ILST,IDLU
      COMMON          /IEROR/YES,NO
      GIBLU=NO
      IDRT=IGET(1652B)
      LUMAX=IGET(1653B)
      DO 10 I=0,LUMAX-1
      IVAL=IGET(IDRT+I)
      GTEQT=IAND(IVAL,77B)
      IF(GTEQT.NE.EQT) GO TO 10
      ISUB=IAND(IVAL,74000B)/2048
      IF(IVAL.LT.0) ISUB=ISUB+20B
      IF(ISUB.NE.IAD) GO TO 10
      GO TO 20
   10 CONTINUE
      RETURN
   20 LUX=I+1
      GIBLU=YES
      RETURN
      END
```

Figure 6-14. Find an LU, Given an EQT and a Device Address

```
      INTEGER FUNCTION FLASS(MNUM,NUM,EQT,AD,ISTRT),02-15-79 (GWG)
C FLASS RETURNS YES WHEN DATA OBTAINED SUCCESSFULY FROM FILE HPIBD
C FLASS RETURNS NO  WHEN NO DATA IS AVAILABLE OR FMP ERROR
C FLASS OUTPUTS AN ERROR MESSAGE WHEN AN FMP ERROR OCCURS
C ISTRT=-1 RESTART AT FIRST RECORD
C ISTRT= 0 INITIALIZE ARRAY (OPEN FILE,ETC.)
C ISTRT= 1-->N RECORD NUMBER
      INTEGER     FLASS,NUM,EQT,AD,YES,NO,FARRY,ERROR,
     &            OPEN,CLOSE,READF,IDCB(144),HPIBD(3),
     &            IBFR(80),IARRY(6,1),NAM(3)
      REAL        MNUM,RARRY(1)
      EQUIVALENCE (RARRY,IARRY(1,1))
      COMMON      ILU,ILST,IDLU
      COMMON      /IEROR/YES,NO
      COMMON      /IBFIL/HPIBD,ISEC,ICR,ISZ,IDCB
C MODIFY BLOCK DATA ARRAY FOR LARGER 'HPIBD' FILES
      COMMON      /FARRY/LNTH,RARRY
C FLASS
      DATA        NAM/2HFL,2HAS,2HS /
C
```

Figure 6-15. Read HP-IB Data File and Make Memory Table

```
      FLASS=NO
      IF(ABS(RARRY(1)-0.).LT..01)ISTRT=0
      IF(ISTRT.NE.-1)GO TO 10
C RESTART AT FIRST RECORD
      ISTRT=1
      J=1
      MNUM=RARRY(1)
      NUM=IARRY(3,1)
      EQT=IARRY(4,1)
      AD=IARRY(5,1)
      FLASS=YES
      RETURN
   10 IF(ISTRT.NE.0) GO TO 50
      ISTRT=1
      J=1
C OPEN THE 'HPIBD' FILE
      IF(OPEN(IDCB,IERR,HPIBD,1,ISEC,ICR).LT.0) GO TO 500
   20 IF(READF(IDCB,IERR,IBFR,80,LEN).LT.0) GO TO 1000
      IF(LEN.LT.0) GO TO 30
      L=LEN
      CALL CNVRT(IBFR,L,RARRY(J),IARRY(3,ISTRT),IARRY(4,ISTRT),
     &IARRY(5,ISTRT))
      ISTRT=ISTRT+1
      J=J+3
      IF(J.LE.LNTH)GO TO 20
      WRITE(ILU,110)
  110 FORMAT("FLASS: INDEXED BEYOND ARRAY LENGTH. SEE SOURCE.")
      STOP
   30 RARRY(J)=-1
      ISTRT=1
      J=1
      MNUM=RARRY(1)
      NUM=IARRY(3,1)
      EQT=IARRY(4,1)
      ADF=IARRY(5,1)
      CALL CLOSE(IDCB,IERR)
      FLASS=YES
      RETURN
   50 ISTRT=ISTRT+1
      J=J+3
      IF(RARRY(J).LT.0)GO TO 60
      MNUM=RARRY(J)
      NUM=IARRY(3,ISTRT)
      EQT=IARRY(4,ISTRT)
      AD=IARRY(5,ISTRT)
      FLASS=YES
      RETURN
   60 ISTRT=-1
      RETURN
  500 IF(IERR.EQ.-6) RETURN
 1000 IDUM=ERROR(NAM,IERR)
      RETURN
      END
```

Figure 16-15. Read HP-IB Data File and Make Memory Table (Continued)

Subroutine CNVRT

This routine was created to funnel all calls to the FORTRAN formatter into one subroutine. The formatter may be eliminated by replacing "CNVRT" with a user version of software.

Block Data Subprogram

These locations were created for global access by other subroutines and the user's main program. The area contains the file "namr", HPIBD:GG:26 which may be modified by the user.

```
      SUBROUTINE CNVRT(IBFR,L,MODN,IN,EQT,AD),02-20-79 (GWG) FORMATTER
C IF L=0 MOVE MODN:IN:EQT:AD INTO IBFR AND RETURN NON-ZERO LENGTH
C IF L.NE.0 MOVE IBFR INTO MODN:IN:EQT:AD AND RETURN
      INTEGER          IBFR(1),IN,EQT,AD,L,IDMY(5)
      REAL             MODN
      COMMON ILU,ILST,IDLU
      IF(L.NE.0) GO TO 50
      CALL CODE
      WRITE(IBFR,10)MODN,IN,EQT,AD
   10 FORMAT(F5.0",","I3",","I2",","I2",")
C
      L=13
      RETURN
   50 CALL CODE
      READ(IBFR,*)MODN,IN,EQT,AD
   60 FORMAT(F5.0,I3,I2,K2)
      RETURN
      END
```

Figure 6-16. Do all Binary to ASCII in One Place

```
      BLOCK DATA,02-16-79 (GWG) GLOBAL PARAMETERS
      INTEGER          YES,NO,EQTBL,EQN,NUM,FARRY,ARRAY,EQT,AD
      REAL             MNUM
      COMMON           /IEROR/YES,NO
      COMMON           /IEQS/EQTBL(64),EQN
      COMMON           /INSTR/MNUM,NUM,EQT,AD
      COMMON           /IBFIL/NAME(3),ISC,ICR,ISZ,IDCB(144)
      COMMON           /FARRY/LEN,ARRAY(300)
      DATA NO/2HNO/,YES/2HYE/,LEN/300/
      DATA NAME/2HHP,2HIB,2HD /,ISC/2HGG/,ICR/26/,ISZ/10/
      END
```

Figure 6-17. Global Parameters

Function ERROR

This function simply prints File Management errors on the user's terminal.

Program AUTLU

Sometimes it is not convenient to use the function GTDLU inside the user's program (one such situation arises when the user program must be of Type 4, or not enough memory is available to included GTDLU in the main program). Most programs in this series of application notes contain a reference only to INPRM.

When File Manager is available, a separate program, AUTLU, may be used in conjunction with a transfer file and the user's program to obtain IDLU automatically. Figure 6-19 shows one such transfer file described in AN 401-18.

AUTLU may be defined as a program which accepts the instrument model number and instrument number and converts this to IDLU. The parameters ILU, ILST, and IDLU are passed back to File Manager in FMGR globals 1P, 2P, and 3P respectively.[8]

The program is shown in figure 6-20.[9] One reference is made to function MODIN (figure 6-21) which is similiar to INPRM. This function reads the model number (a real variable) and instrument number from the run parameters however. The function GTDLU is then called which obtains IDLU.

```
      INTEGER FUNCTION ERROR(NAM,IERR),02-20-79 (GWG) WRITE FMP ERROR
      INTEGER NAM(3),ERROR,YES
      COMMON ILU,ILST,IDLU
      COMMON /IEROR/YES,NO
      ERROR=YES
      WRITE(ILU,10)NAM,IERR
   10 FORMAT(/3A2": FMP ERROR "I3"."/)
      RETURN
      END
```

Figure 6-18. Print FMP Errors

```
0001   :SV,1,9,IH
0002   :RU,AUTLU,,59306
0003   :RU,A306,2P,3P
0004   :SV,9G,,IH
0005   ::
```

This transfer file determines the 59306A device LU and schedules the test program "A306."

Figure 6-19. 59306A Transfer File

---

[8] *File Manager globals are discussed in the Batch Spool Monitor Manual (92060-90013).*

[9] *Programs and subroutines shown in this chapter are available from the Contributed Library (22683-13346).*

```
FTN4,L
      PROGRAM AUTLU(3),02-15-79 (GWG) TEST SUBROUTINES
      INTEGER        NO,YES,GTDLU,EQT,AD,IPRM(5)
      REAL           MOD
      COMMON         ILU,ILST,IDLU
      COMMON         /IEROR/YES,NO
      COMMON         /INSTR/MOD,IN,EQT,AD
      CALL DTACH
      IF(MODIN(ILU,ILST,MOD,IN).EQ.NO) STOP
      IF(IN.EQ.0)IN=1
      IF(GTDLU(MOD,IN,IDLU).EQ.NO) STOP
  100 IPRM=ILU
      IPRM(2)=ILST
      IPRM(3)=IDLU
      CALL PRTN(IPRM)
      END
```

Figure 6-20. Program AUTLU, Automatic LU Assignment

```
FTN4,L
      INTEGER FUNCTION MODIN(ILU,ILST,RMOD,IN),11-29-78 (GWG) MODEL
      INTEGER        ISTRNG(40),OSTRNG(10),STRT
C
C 'MODIN' GETS:
C
C    A. THE INPUT LOGICAL UNIT (INTERACTIVE TERMINAL).
C    B. THE LIST LOGICAL UNIT FROM PARAMETER ONE (IT
C       SETS THE LIST LU EQUAL TO THE INPUT LU IF THE
C       LIST LU IS 0).
C    C. THE MODEL NUMBER OF THE HP-IB DEVICE FROM PARAMETER TWO
C    D. THE INSTRUMENT NUMBER OF THE HP-IB DEVICE FORM PARAMETER THREE
C
C
      MODIN=2HNO
      ILU=LOGLU(ID)
      CALL GETST(ISTRNG,-80,RTNCLN)
C     CALL EXEC(2,ILU,ISTRNG,-RTNCLN)
      CALL CODE
      READ(ISTRNG,*)ILST,RMOD,IN
      MODIN=2HYE
      RETURN
      END
```

Figure 6-21. Function MODIN, Obtain Model and Instrument Number

Automatic LU Assignment When Using the Session Monitor

HP-IB logical unit numbers may also be obtained automatically in a session monitor environment. Although the task is more involved, it can be accomplished using a few programs and a File Manager transfer file. The programs needed are listed and described in figure 6-22.

Scheduling a user program with automatic parameters requires six operations.

1. Transfer file *AUTOP is executed by a user at a CRT terminal.
2. Program GTSLU determines the first available (and assignable) session LU and returns it in File Manager global 1P.
3. Program AUTLU, discussed previously, finds a system LU corresponding to the model number of the HP-IB instrument and returns it in global 3P. This program was discussed in the previous section.
4. Transfer file *AUTOP executes the File Manager command :SL,1P,3P to set up the session LU correspondence for the user. When the process completes, a new session LU correspondence exists between the session LU and an HP-IB device.
5. Having determined ILST and IDLU, *AUTOP schedules the user program and it runs to completion.
6. *AUTOP then unassigns the session LU and the system LU and returns control to the user.

Transfer File *AUTOP

The procedure file does only simple error checking and executes the session switch command 'SL'. More sophisticated error checking is performed by GTSLU and AUTLU. Errors are output from within the programs. An example of *AUTOP is shown in figure 6-23.

Program GTSLU

Program GTSLU is defined as a user program which obtains the first available session LU from the user's session switch table. If no session LUs are available, or the user's capability is not sufficient, or no switch spaces are available, an error is returned. Figure 6-24 shows one example of how GTSLU may be written.

```
:TR,*AUTOP,ILST,Program Name,Instrument Model #,Instrument #
```

This is a File Manager transfer file which combines the capabilities of two user programs, GTSLU and AUTLU. The object of the transfer file is to allow a user to schedule a program which requires ILST and IDLU for it's run parameters. ILST must be supplied to AUTOP, but instead of supplying IDLU, the user supplies the model number and instrument number of the HP-IB device. Parameter IDLU is obtained automatically.

```
:RU,GTSLU
```

This program determines the first unassigned session LU if one is available. If none are available or all switch LUs are currently allocated, an error is printed and 0 is returned in File Manager 1P. Otherwise the first available (assignable) session LU is returned in global 1P.

```
:RU,AUTLU,ILST,Instrument Model #, Instrument #
```

This program obtains a system LU pointing to the instrument whose model number and instrument number is supplied in the run statement for AUTLU. The system LU is returned in File Manager global 3P. If no system LU is available an error is printed and 0 is returned in global 3P.

Figure 6-22. User Programs Needed for a Session Environment

```
:SV,0,,IH
:IF,9P,GE,50,2
:DP,SORRY, YOU HAVEN'T A HIGH ENOUGH CAPABILITY LEVEL.
:
:RU,GTSLU
:IF,1P,NE,0,1
:
:CA,-4:P,1P
:RU,AUTLU,,2G,1
:IF,3P,NE,0,1
:
:SL,-4P,3P
:RU,1G,-4P
:SL,-4P,-
:SYLU,3P,0
: 06-21-79 (TRK) (GWG)
: THIS TRANSFER FILE WILL RUN HP-IB
: PROGRAMS USING HP-IB DEVICE LUS. SIMPLY ENTER:
::TR,AUTOP::UU,PROGRAM NAME,ILST,HP-IB MODEL NUMBER,INSTRUMENT NUMBER
:
```

Determine the first available session LU.

Obtain the first available system LU.

Create a correspondence between the session LU and the system LU, and schedule the user program. Then clean up and exit.

Figure 6-23. Transfer File AUTOP

```
FTN4,L
      PROGRAM GTSLU(3),06-14-79 (TRK) (GWG) GET F/A SLU
      INTEGER IBUF(155),ILEN,IERR,SESLU,SYSLU,ILU,J,I
      INTEGER USEAR(256),FRSTLU,IPRAM(5),CAP,SPARE,IP(5)
      DATA LEN/155/
      ILU=LOGLU(J)
      SPARE=0
      FRSTLU=0
      CALL GTSCB(IBUF,LEN,IERR)
C SESSION NOT ACTIVE IF IERR=-1
      IF(IERR.NE.-1) GOTO 20
      WRITE(ILU,903)
  903 FORMAT(" SESSION IS INACTIVE.",
     &" NO SESSION LU CAN BE FOUND."/)
      GO TO 999
C BUFFER TOO SMALL IF IERR1
   20 IF(IERR.GT.-1) GOTO 30
      WRITE(ILU,904)
  904 FORMAT(" BUFFER 'IBUF' DIMENSIONS TOO SMALL.",
     &" THIS PROGRAM MUST BE MODIFIED."/)
      GO TO 999
   30 DO 5,I=1,256
      USEAR(I)=0
    5 CONTINUE
      SSTLEN=IABS(IBUF(13))
      DO 10 I=1,SSTLEN
```

Read the user's session control block.

Check for spares in the user's session control block.

Figure 6-24. Obtaining the First Available Session LU

```
C MASK OFF SYSTEM LU
      SYSLU=IAND(IBUF(13+I),177400B)
      SYSLU=SYSLU/400B
C MASK OFF SESSION LU
      SESLU=IAND(IBUF(13+I),377B)
      USEAR(SESLU+1)=1
C SPARE=1 MEANS A SPARE EXISTS
      IF(SYSLU.EQ.-1) SPARE=1
   10 CONTINUE
      IF(SPARE.EQ.1) GO TO 60
      WRITE(ILU,907)
  907 FORMAT(" NO SPARE SESSION LUS ARE",
     &" AVAILABLE. OPERATOR INTERACTION IS REQUIRED."/)
      GO TO 999
   60 DO 40 I=1,256
```

Index through table looking for available session LUs.

```
      IF(USEAR(I).NE.0) GO TO 40
      FRSTLU=I
      GOTO 100
   40 CONTINUE
  100 IF(FRSTLU) 70,70,200
   70 WRITE(ILU,909)
  909 FORMAT(" ALL SESSION LUS ARE ASSIGNED.",
     &" OPERATOR INTERACTION IS REQUIRED."/)
      STOP 4
  200 IPRAM=FRSTLU
  999 CALL PRTN(IPRAM)
      END
```

Figure 6-24. Obtaining the First Available Session LU (Continued)

HEWLETT PACKARD

Printed in U.S.A. 5/80 5953-2800