HEWLETT PACKARD

HP-IB

Performance Evaluation of HP-IB Using RTE Operating Systems

Application Note 201-4

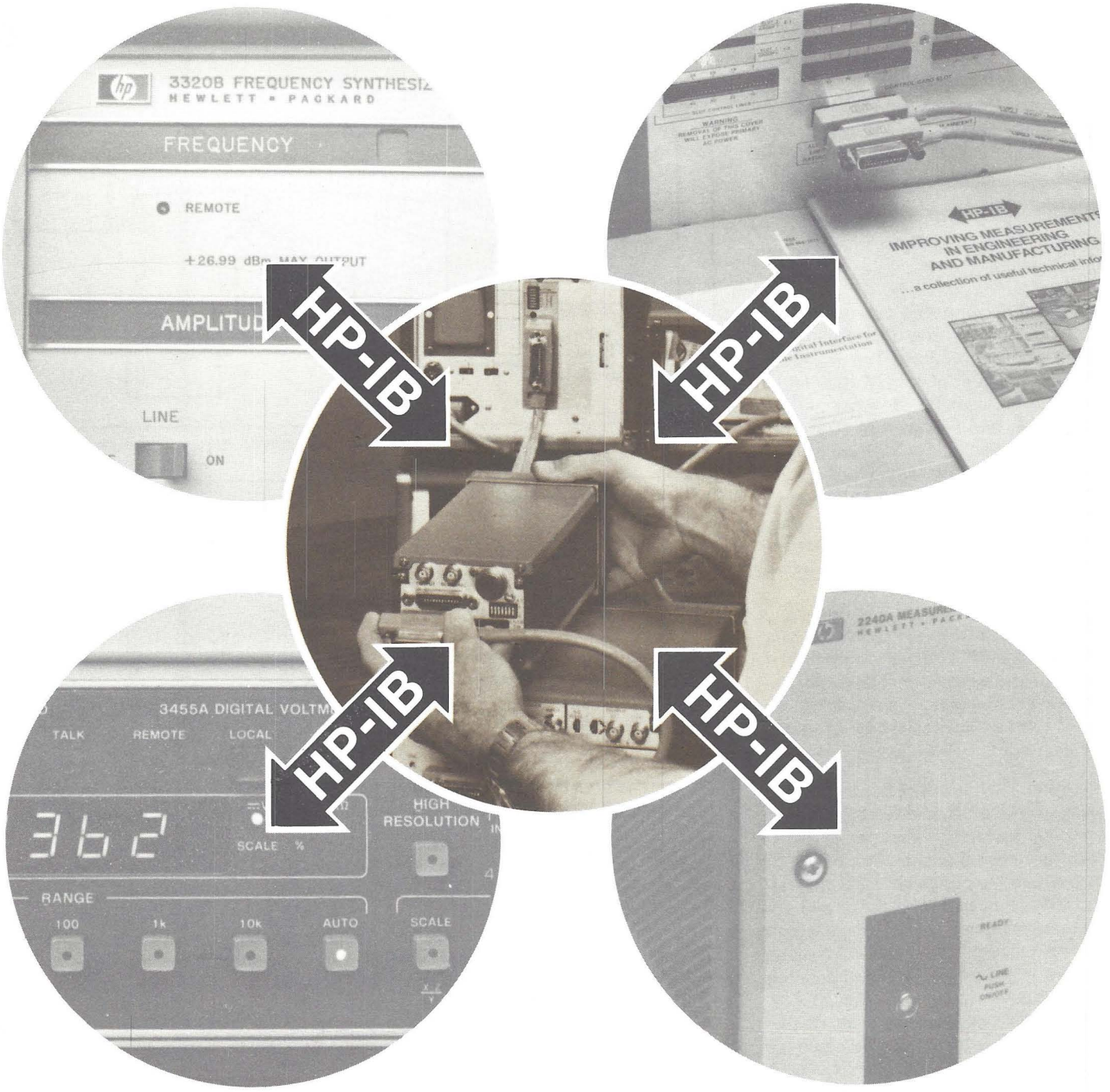


Table of Contents

This application note will help you to determine the capabilities and limitations of the Hewlett-Packard Interface Bus (HP-IB) when used with an HP 1000 Computer System. HP-IB is Hewlett-Packard's implementation of IEEE Standard 488-1978, "Digital Interface for Programmable Instrumentation".

All HP 1000 computers use the Hewlett-Packard family of Real-Time Executive (RTE) operating systems. The newest and smallest member of the HP 1000 family, the L-Series, uses RTE-L, while the HP 1000 M, E, and F-series use RTE-II or RTE-IV. Each version of RTE has its unique features, but the real-time philosophy is similar. This gives the important benefit of program transportability. Most programs written in a high level language are transportable from machine to machine without any changes.

The references in this note pertain to all RTE systems, HP-IB drivers, and HP-IB I/O cards unless specifically stated. To determine the operating system, driver, and HP-IB card for your particular system, please refer to table 1

The HP-IB interface card is just one of the many interface cards available for your computer system. Its vast capabilities, however, make it unique. The HP-IB I/O card allows the computer to control, talk, and listen to a variety of both fast and slow devices connected to it. HP-IB compatible devices include instruments such as voltmeters, high speed D/A's and counters, and computer peripherals such as graphic devices, discs, or even other computers. The length of messages can span from one byte to thousands of bytes per message (and higher, as new and more intelligent devices become available).

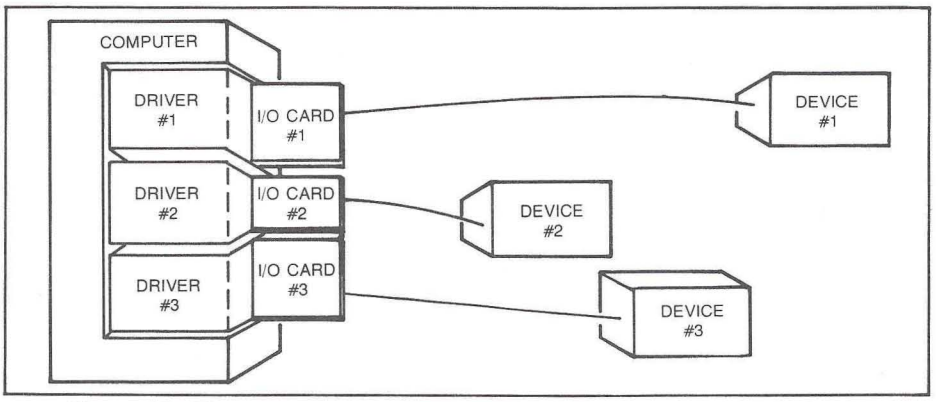


Figure 1A. The "old" way, with a separate driver, separate I/O card, and separate cable for each device.

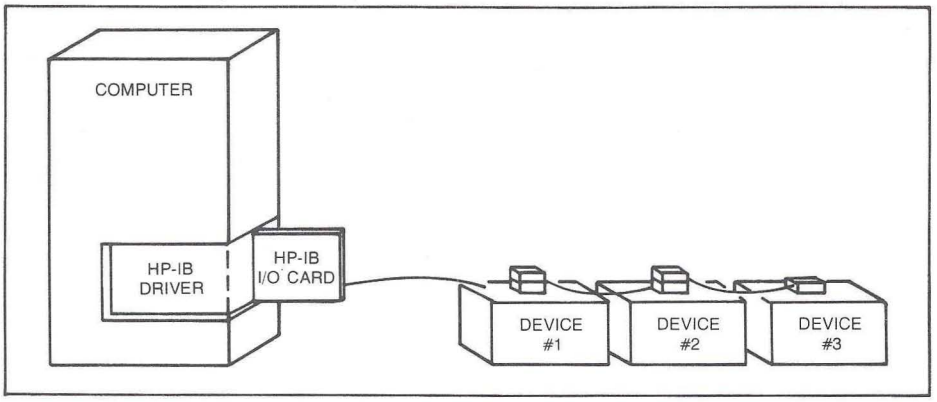


Figure 1B. The HP-IB way. One driver, and up to 14 devices on each bus.

Because it is designed for an extremely wide range of interfacing uses, HP-IB may not be the fastest method. However, it offers the most flexibility at a minimum cost. Up to 14 HP-IB devices can be interconnected with only one interface card. The same driver operates all devices on the bus. When compared to the previous methods of requiring a separate interface card and separate driver for each device, the cost advantage of the HP-IB is readily apparent.

This performance note presents a model which can be used to determine the time to send or receive data messages from various instruments and devices operating with any of the HP 1000 computers using RTE and HP-IB. The model can also be used to calculate the amount of spare time the computer will have during a measurement. This spare time, or unused computer potential, can be used to operate other HP-IB test stations or perform other progam operations.

The speed of an HP-IB system will depend on the measurement and commmunication speeds of the instruments being used. For this reason, various popular instruments (with differing measurement and communication speeds) have been used as examples. The instruments used are the HP 3455A Digital Voltmeter (average speed and average message length), HP 3437A System Voltmeter (high speed and short message length), and HP 2240A Measurement and Control Processor (high speed and large buffer capable of long message lengths). The HP 3495A Scanner (a listener only) is also shown in conjunction with the 3455A.

Measurement speeds and timing requirements for instruments are usually available in their respective Operation and Service Manuals, or from the instrument supplier.

This performance note is divided into five sections:

1. An overview of how HP-IB is used
2. Timing and performance of HP-IB systems
3. Service request timing and usage
4. Time-out handling and usage
5. HP-IB performance enhancement possibilities

Table 1. Hardware and Software needed for HP-IB in HP 1000 Computers.

COMPUTER	HP 1000 L-Series	HP 1000 M-Series	HP 1000 E-Series	HP 1000 F-Series
OPERATING SYSTEM	RTE-L	RTE-II or RTE-IV	RTE-II or RTE-IV	RTE-II or RTE-IV
DRIVER	ID.37	DVR37	DVR37	DVR37
HP-IB I/O CARD	12009A	59310B	59310B	59310B

The driver's role in controlling the HP-IB

The HP-IB driver is the key link in any RTE based system between a running program and the HP-IB interface card (figure 2). It is a general purpose driver used in all HP-IB/RTE configurations to the bus. The driver can be called through FORTRAN, BASIC, PASCAL, and Assembly languages, making it a powerful tool that can be used in a variety of environments.

The driver assumes responsibility for all HP-IB interaction between the computer and the interface card for the bus. This includes control modes, device addressing, and data passage. In an automatic addressing mode (autoaddressing), which is the most common technique used, each device is addressed by a pre-chosen logical unit number. Standard READ/WRITE statements are then used. For example, if a digital voltmeter was chosen to be logical unit (LU) [1] number 29, a voltage reading in BASIC to be called "V1" would be set up as follows:

```
READ#29;V1
```

The driver assumes the responsibility of addressing the device, controlling the transmission of the data, and accepting the data into the computer as variable "V1".

Translating the data: The formatter

When data is brought into the computer, it is usually translated to a format internally usable by the computer (i.e., ASCII to binary). The same process is used in reverse on output; the data is translated from the internal representation to the format which the device can accept. The routine used to make this translation is called the formatter.

[1] A logical unit (LU)is the method that RTE uses to reference I/O devices. This allows the flexibility of writing a program without specifying the physical location of the device. At execution time, RTE dynamically matches the LU and the device.

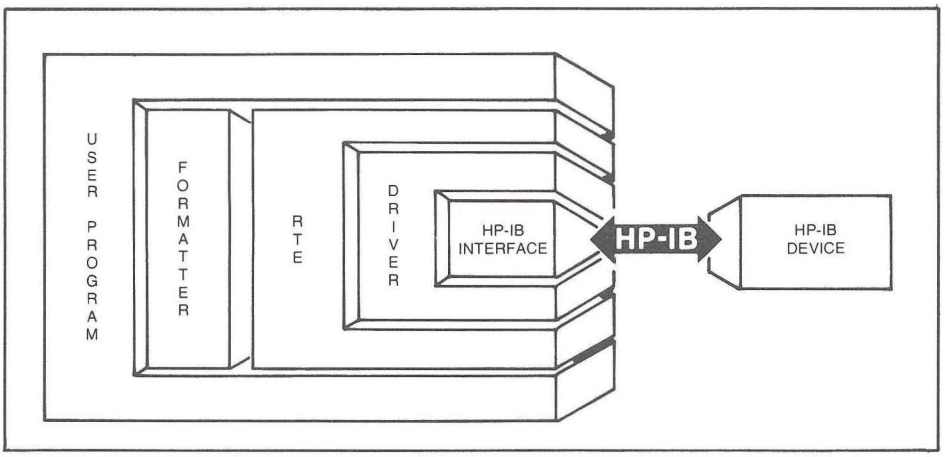


Figure 2. Structure of the various components used when working with HP-IB.

ASCII format is most commonly used for data transfer over an HP-IB bus. For example, the value "$3.141592653 \times 10^{12}$" (which is stored in two computer words as a floating point number) would be transferred via HP-IB as fourteen 8-bit bytes (each byte holds one character).

Timing studies for RTE have produced a formula that can be used in the HP-IB model for the formatter overhead. The equations for the HP 1000 L, M, E, and F-series computers are:

$$t(L) = 4.402 + .606n$$
$$t(M) = 1.928 + .288n$$
$$t(EF) = 0.697 + .109n$$

where n is the number of bytes and t is the time in milliseconds. For example, a 14 byte string on an HP 1000 L-series computer would take approximately 12.9 milliseconds to convert.

The formatter, drivers, languages, and user programs are all overseen by the RTE operating system. It controls the order of execution, and provides protection for unauthorized actions. For efficiency, certain routines (like the driver) are linked to and included in the operating system. Therefore, each time the driver is accessed, the request must go through RTE. Conceptually, this flow is shown in figures 3 and 4, which shows a model of HP-IB within an RTE environment. This model shows that the overhead of the driver is only a small portion of total system overhead when using HP-IB. Overhead expenses for other components of the system, such as the formatter and operating system, must be considered to give a proper view of overall HP-IB performance.

For both input and output, notice in figures 3 and 4 that communication with HP-IB can be performed with or without the formatter. Skipping the formatter step will save processing time. If computation is not to be performed on the data (i.e., the data is to be stored or displayed), you probably won't need formatting.

Data transfer techniques

Data brought into HP 1000 M, E, and F-Series computers via the HP-IB I/O card is handled in one of two ways. The first method is Direct Memory Access (DMA). With DMA, the information is brought into memory directly from the interface card. This process is controlled by the Dual Channel Port Controller (DCPC) which acts as a direct pipeline between the memory and I/O sections of the computer. DMA is the fastest method for transferring information, and is necessary for functions such as disc transfers. Once assigned, the DMA channel is dedicated to the I/O card until the data transfer is completed. There are 2 DMA channels available when the DCPC is installed. They can both be used for data transfer, but if you have a disc, remember that it requires one of the channels when swapping programs or transferring data. If you use both channels for an HP-IB data transfer, the disc and other devices which use DMA must wait until one of the data transfers completes.

The other method of data transfer for HP 1000 M, E, and F-Series computers is non-DMA, or "interrupt processing." With this method, as in the DMA case, the computer (via the driver) initiates the data transfer, and then goes off to perform other tasks. The difference is that the HP-IB card and driver work in conjunction to transfer data to or from the memory by triggering an interrupt when the operating system is needed. When the interrupt signal is noticed, the operating system returns, transfers the data via software, and goes off again to other tasks. The set-up time for interrupt processing is less than that for DMA, but the transfer time for each byte is greater since interrupt servicing is required for each byte (figure 5).

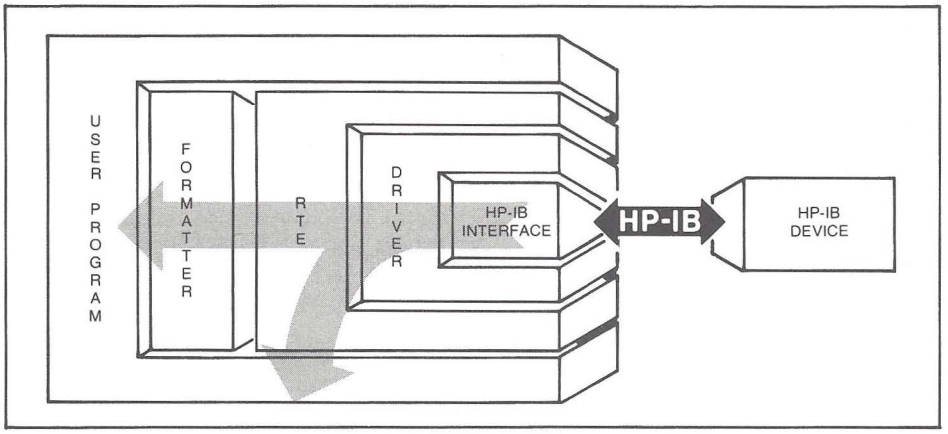


Figure 3. Input from the HP-IB.

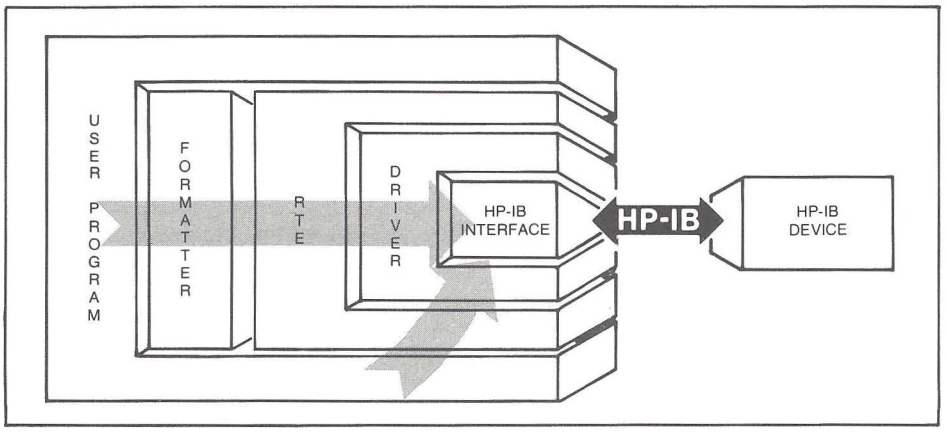


Figure 4. Output to the HP-IB.

Note: For both input and output, communication with HP-IB can be performed with or without the formatter. Skipping the formatter step will save processing time. If computation is not to be performed on the data (the data is to be stored or displayed) you don't need formatting.

The previous discussion pointed out that the fastest and least CPU-demanding technique for data transfer is DMA. Sufficient DMA channels would need to be available, however, to satisfy all I/O processes. This is the design philosophy for the HP 1000 L-Series. Every I/O card in the L-Series computer has a DMA channel, and uses it automatically. With this design, multiple devices such as discs, tape drives, terminals and instruments can all use DMA concurrently. Data transfer rates have been maximized, and CPU utilization has been minimized. The use of interrupt processing for HP-IB in an L-series computer has been eliminated.

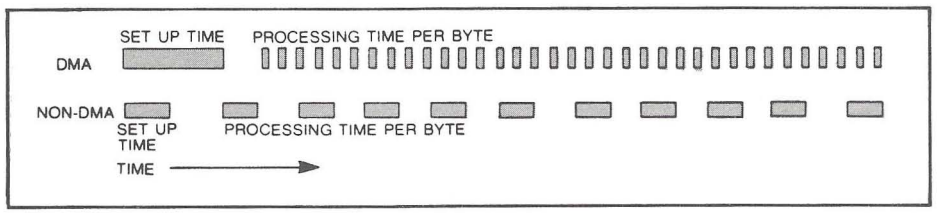


Figure 5. Although the set-up time is greater for DMA, the time to process each individual character or byte in a message is much less.

HP-IB Performance

The most important question to be asked about any HP-IB configuration is whether there is sufficient capability to perform the needed task. This really breaks down to whether the devices connected on the bus will run at the required rate, and how much capacity is left in the computer for other tasks. Factors influencing this decision include interface card speed, device speed, length of the data message, and how the data will be treated once inside the computer.

The results reported in this performance note were obtained on an HP 1000 Model 45 system with RTE-IVB using standard (595ns) memory, and on an HP 1000 Model 10 with RTE-L.

HP-IB driver speed

The first item needed to determine HP-IB performance is the speed of the interface card operating with its driver. Table 2 shows the maximum transfer speeds for both DMA and non-DMA for RTE-IV, and DMA for RTE-L. Figures 6 and 7 illustrate these results graphically. It can be seen that for each method there is a fixed set-up time, and then a fairly linear increment of time per byte of data in a message. The set-up time includes the time to pass through the operating system, driver housekeeping, and addressing a device as a talker or listener (the formatter is separate).

The total time to perform an input or output task is the sum of the time required for the data transfer plus the time required to format the data. This can be represented by the formula:

$$t(IO) = t(exec) + t(format)$$
$$= (\text{HP-IB setup time} + \text{transfer rate} \times \text{number of bytes}) + (\text{format setup time} + \text{conversion rate} \times \text{number of bytes})$$

Note that if the formatter is not required (i.e., an EXEC CALL is used), the terms for the formatter in the above equation revert to zero.

All coeffecients for the above equation are presented in this note, but a question arises about the data transfer rate. Most devices communicate at a rate slower than HP 1000 computers can. Therefore, should the data transfer rate of the computer be used, or should the data transfer rate of the device be used instead? This question has to be answered separately for DMA and non-DMA operation.

Using DMA processing, the 59310B can handle approximately 550,000 bytes per second, and the 129009A can handle over one million. When a device transfers data slower than these rates, the DMA transfer occurs at the slower device rate. To compute the DMA transfer time, multiply the number of bytes to be transferred by the time required for the device to transfer one byte (seconds/byte). Then, add 6 milliseconds for approximate set-up time.

Whenever a byte has been transferred using non-DMA processing in an HP 1000 M, E, or F-Series computer, the driver checks to see if the next byte is ready. If it is, the driver processes it directly. This is displayed by the fast rate equation (curve B). The speed of the device must be faster than 17,000 bytes per second (60 microseconds/byte) to enjoy the higher speed slope.

If the next byte is not ready, the computer goes off to another task. When the next byte is ready, an interrupt is generated, causing execution to return to the driver. It then takes up to 1.1 milliseconds (on input) to process the byte. This is displayed by the slow transfer rate equation (curve C). Note that the set-up time also appears to increase. This slow transfer rate is slower than the transfer rate of most devices. The result is 2 distinct curves, B and C, for non-DMA processing. Either the device is fast enough to ride the fast curve (B), or it will "miss the bus" and have to take the slower curve (C). As stated earlier, the L-Series uses DMA, so these non-DMA curves are not applicable.

The intersection of the fast non-DMA and DMA curves for RTE-IV is approximately 130 bytes for output and 56 bytes for input. As a general rule, total time is minimized when fast interrupt processing is used to the left of the intersection and DMA to the right. However, common sense dictates that a short burst of data too fast for non-DMA[2] should be handled by DMA. Likewise, a long, slow string should be handled via non-DMA. For example, the 3437A voltmeter can be programmed to delay up to 1 second between readings. At this rate, a burst of 10,000 readings (the maximum the 3437A can send after a single request) would take almost 3 hours, sending only 2 data bytes per reading. Tying up a shared DMA channel for such infrequent use would be an inefficient use of the resource.

[2]The maximum non-DMA transfer rate is approximately 17,000 bytes per second. This limitation is imposed by the driver structure.

Table 2. HP-IB data transfer equations.

Operating System	INPUT		OUTPUT	
	Setup Time (MS)	Time Per Byte in Message (MS)	Setup Time (MS)	Time Per Byte in Message (MS)
RTE-L DMA (Curve "A")	6.28	.001n	6.29	.001n
RTE-IV DMA (Curve "A")	5.95	.002n	6.14	.002n
RTE-IV Fast Interrupt (Curve "B")	2.71	.060n	2.95	.020n
RTE-IV Slow Interrupt (Curve "C")	5.73	1.10n	7.61	.560n

Note: The maximum data rate for the 59310B is 550,000 bytes/second.
The maximum data rate for the 12009A is 1,098,000 bytes/second.

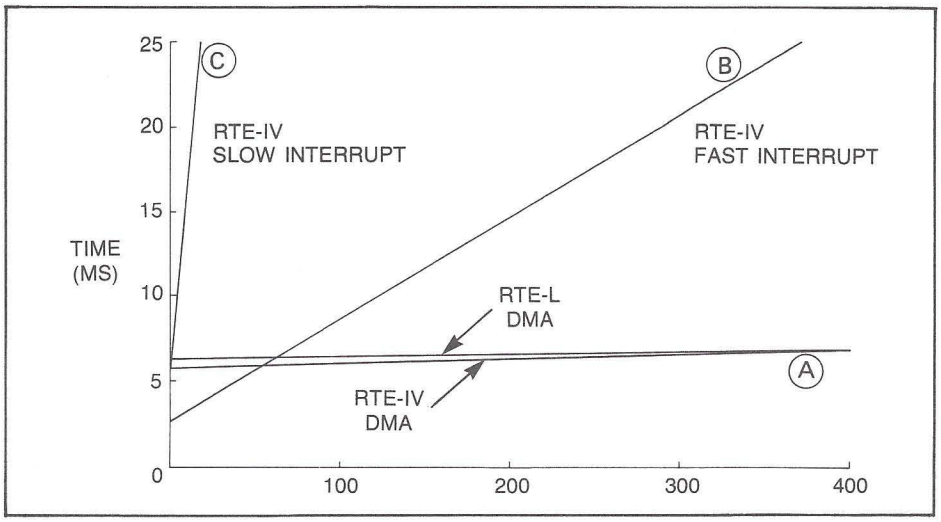


Figure 6. Maximum Number of Input Bytes.

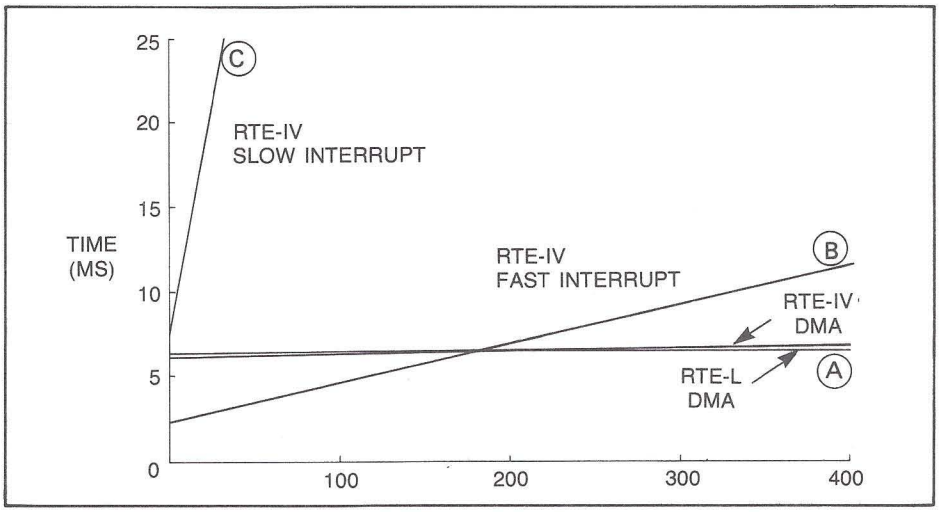


Figure 7. Maximum Number of Output Bytes.

An HP-IB model

The HP-IB transfer speed coefficients shown in table 2 can be used in conjunction with the system flow diagram (figures 3 and 4) to construct a conceptual model of HP-IB system performance.

The examples for this model will be the HP 3455A and HP 3437A Digital Voltmeters and the 2240A Measurement and Control Processor. The voltmeters represent many of the requirements of working with HP-IB, and are the most popular multimeters currently offered. The 2240A is one of the first HP products that uses Silicon-on-Sapphire (SOS) microprocessor technology, and lends itself to many intelligent measurement and control applications.

In most cases, there is a set pattern to the order in which HP-IB functions are performed. In the housekeeping section, the remote enable command is executed to get the device's attention for remote control. Also, system configuration of the device can be altered to provide for time-out processing, service requests, and DMA. RTE-IV defaults to non-DMA, for the resource management reasons previously explained. RTE-L automatically provides DMA.

When HP-IB system configuration has been completed, the HP-IB devices and computer go through the following steps:

- **Computer TALKS/device LISTENS**

  the computer sends the device codes to set up and program the device

- **Computer activates the device**

  the desired function is performed by the device

- **Computer LISTENS/device TALKS**

  the device sends a response message or a status message

Various combinations of the above steps can occur. Also, the computer may use the formatter to translate either the incoming or outgoing data. More complicated models could also be constructed. However, these steps are the basic building blocks, and will be used in our examples.

Performance Examples: HP 3455A Digital Voltmeter

Figure 8 shows the conceptual model for an HP-IB system with the HP 3455A Digital Voltmeter. The 3455A is a high accuracy digital voltmeter capable of transmitting up to 24 readings per second in the d.c. mode if high resolution is not required. In the high resolution setting, however, as few as 2 readings per second could be the maximum (remember, it depends on the device settling time and function performed). Sixteen bytes (including carriage return and line feed) are transmitted for each reading.

In the maximum-rate low resolution mode, according to the model, each reading should take a total of 62 milliseconds to complete for RTE-L, and 48 milliseconds for RTE-IV. During this period, the computer will be active for over 26 milliseconds in RTE-L, or 20 milliseconds in RTE-IV, to control the transfer and translate the data. The difference between

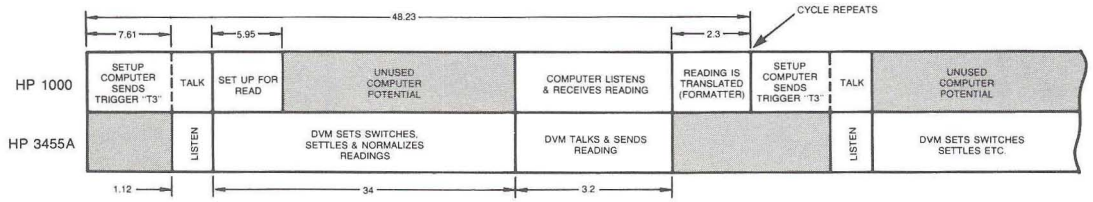
computers appears to be CPU processing speed for the formatter. The main point, however, is to observe that the computer is able to process other tasks during the voltmeter settling time period (shaded area) until the characters begin to transfer.

After running this test on the HP 1000, it was found that the readings actually took 50 milliseconds (or 20 readings per second) with system utilization of 16 milliseconds of CPU time for the RTE-IV system, and 60 milliseconds (16 readings per second) with 28 milliseconds of utilization for RTE-L.

System utilization is given in absolute time instead of a percentage, because it might be misleading to report utilization as a percentage. If the DVM settling varies, the total time to complete the measurement will also vary. However, the absolute time for computer involvement will remain constant, regardless of settling time. Therefore, the percentage utilization for a reading decreases as the settling time increases.

The results obtained for the 3455A DVM and all other devices tested were shorter than the model for total time required. This is understood when it is realized that the worst case settling times were used for estimation in the model. CPU utilization differed slightly in each example. This is mainly due to system overhead, the testing program, and DMA cycle stealing.

To characterize the use of non-DMA in RTE-IV, the same experiment was conducted with the 3455A. The model and non-DMA equations show that the data processing time can be expected to increase when not using DMA, and so will CPU involvement. Without DMA, measurement took 58 milliseconds to complete (17 readings per second) with 28 milliseconds of CPU utilization. This shows that DMA provides a benefit in speed and overhead even if the fast curve cannot be met. If DMA is available, it should be used.



STEP	HP 1000 COMPUTER	HP 3455A VOLTMETER	TOTAL TIME
COMPUTER SENDS MESSAGE TO PROGRAM AND TRIGGER DVM			
• SETUP	7.61		7.61
• SEND MESSAGE — METER LISTENS (2 BYTES)	1.12	1.12	8.73
DVM SETTLES		34	42.73
COMPUTER RECEIVES READING FROM DVM			
• SETUP	5.95		
• RECEIVE VALUE DVM TALKS (14 BYTES)	3.2	3.2	45.93
FORMAT VALUE	2.3		48.23
TOTALS	20.18		48.23

Note: All times in milliseconds

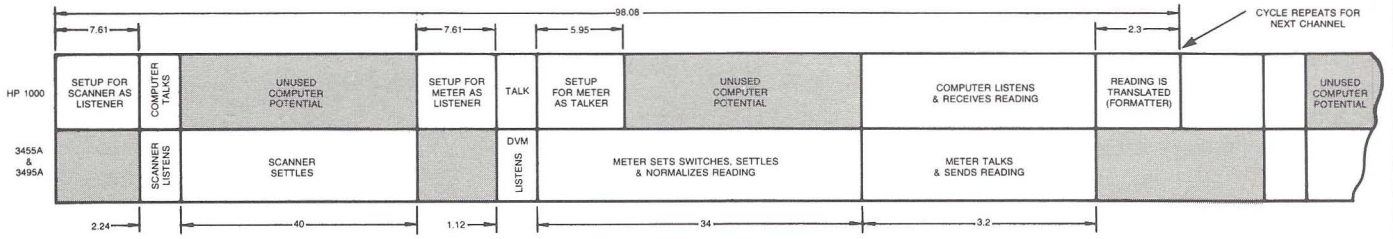
$$\%\ \text{UTILIZATION} = \frac{\text{CPU TIME}}{\text{TOTAL TIME}} = \frac{20.18}{48.23} = 42\%$$

Figure 8. Model for HP 3455A Digital Voltmeter.

HP 3455A Digital Voltmeter and HP3495A Scanner

The next example of HP-IB performance is with a 3455A Digital Voltmeter and the HP 3495A Scanner. This combination is very popular, since it allows one meter to be used to measure multiple test points. Figure 9 shows the timing model when reading through 10 channels on the DC volts range. First a scanner channel must be closed. Then the meter must be triggered and read. It is estimated by the model that it requires 121 milliseconds per channel read with 48 milliseconds of computer utilization time for RTE-L, and 90.47 milliseconds per channel with 30 milliseconds utilization for RTE-IV. The actual time was 120 milliseconds per channel (9.5 reading/sec) with 46 milliseconds computer utilization for RTE-L, and 95 milliseconds with 30 milliseconds utilization for RTE-IV.

The previous examples showed the operation of HP-IB with typical instruments that perform tasks individually. You can perform the same analysis for other HP-IB devices by consulting the particular device manual to find the device-dependent times, such as settling time, gate time, integration time, or other delays. Also, Application Note Series 401, HP-IB Programming Examples, presents performance information for many HP-IB instruments used with an HP 1000 computer.



STEP	HP 1000 COMPUTER	3455A & 3495A VOLTMETER & SCANNER	TOTAL TIME
COMPUTER SENDS MESSAGE TO CLOSE SCANNER CHANNEL			
• SETUP	7.61		7.61
• SEND MESSAGE — SCANNER LISTENS (4 BYTES)	2.24	2.24	9.85
SCANNER SWITCH SETTLES		40	49.85
COMPUTER SENDS MESSAGE TO PROGRAM AND TRIGGER DVM			
• SETUP	7.61		57.46
• SEND MESSAGE — METER LISTENS (2 BYTES)	1.12	1.12	58.58
DVM SETTLES		34	92.58
COMPUTER RECEIVES READING FROM DVM			
• SETUP	5.95		
• RECEIVE VALUE — DVM TALKS (14 BYTES)	3.2	3.2	95.78
FORMAT VALUE	2.3		98.08
TOTALS	30.03		98.08

Note: All times in milliseconds

$$\%\,\text{UTILIZATION} = \frac{\text{CPU TIME}}{\text{TOTAL TIME}} = \frac{30.03}{98.08} = 31\%$$

Figure 9. Model for HP 3455A Digital Voltmeter and HP 3495A Scanner with Option 2 installed.

HP 3437A Digital Voltmeter

The HP 3437A Digital Voltmeter is a good example of an instrument requiring DMA processing. The 3437A can take over 5000 readings per second, and up to 9999 readings per burst. Each reading transmits 2 bytes when "packed format" is selected. This rate is faster than non-DMA processing can handle. Although the 3437A can take a reading in 175 microseconds, the driver would revert to the slow non-DMA curve because the delay between bytes is greater than 60 microseconds. Therefore on curve C, it would take 1.1 milliseconds to process the interrupt for the next reading. During this time, 6 readings would be lost, due to the synchronous operation of the 3437A. DMA must be used.

A test was performed with the 3437A, taking various numbers of readings (from 100 up to 5000). The total time values were then plotted on a graph (figure 10), and linear regression performed to extrapolate the setup time and computer involvement. The slope of the lines in figure 10 shows a reading rate of 5535 readings per second (with 2.9% CPU utilization) for RTE-L, and 4758 readings per second (with 3.8% CPU utilization) for RTE-IV. The y-intercept in figure 10 represents the setup time to trigger the 3437A plus the setup time for the data transfer.

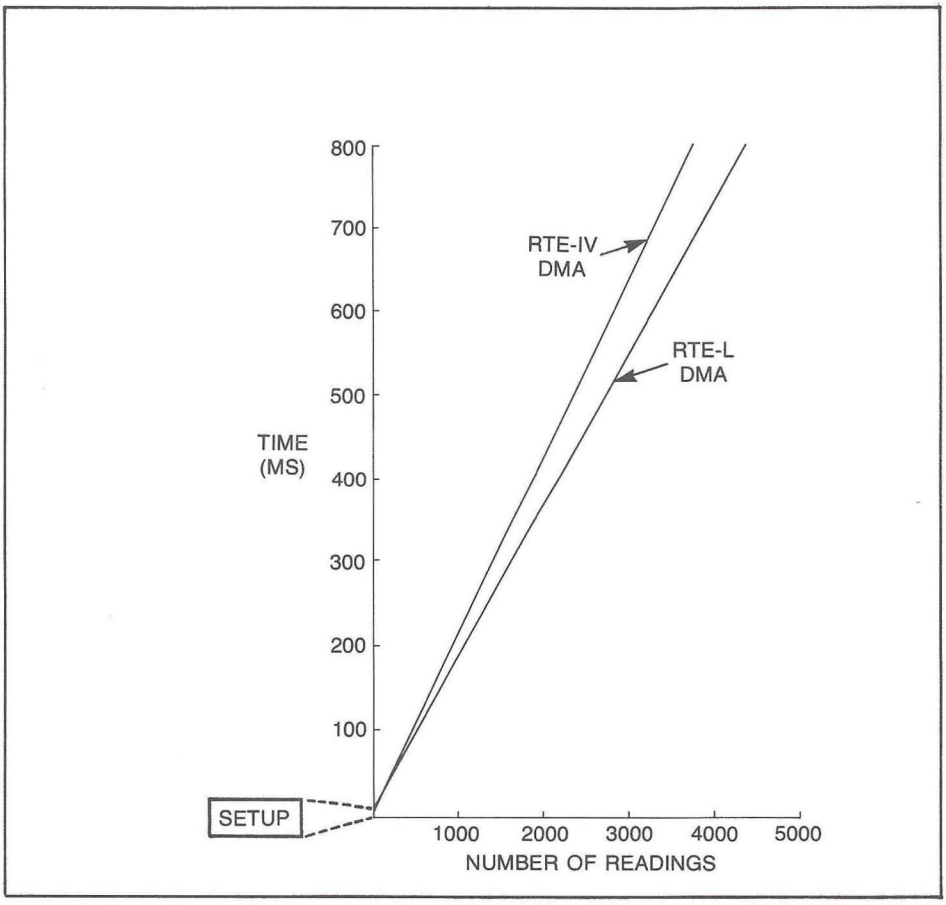


Figure 10. Reading rates with 3437A voltmeter. The intersection at the "Y" axis represents the setup time to make the 3437A the talker.

HP 2240A Measurement and Control Processor

One interesting aspect of HP-IB is its performance with more intelligent devices. As devices on the bus become more intelligent, RTE involvement tends to decrease. To demonstrate this, look at the curves shown in figures 6 and 7. HP-IB performance was described as a line with y-intercept representing the set-up time. Typically, via DMA, the y-intercept is approximately 6 milliseconds with a positive slope in microseconds per byte. When using DMA most of the overhead is incurred before the actual transfer begins. Therefore, it is to the user's advantage to transfer as many bytes as possible in one burst. An optimal solution is to program a device one time and have it perform a large number of tasks independent of the computer. One such intelligent device is the 2240A Measurement and Control processor. The 2240A can be programmed to perform analog point scanning, digital point scanning, establish control loops, respond to interrupts, and finally return requested results to the computer.

A test was performed with the 2240A, having it capture and return 8, 16, and 24 analog samples via DMA. The total time values were then plotted and extrapolated to observe computer involvement. The setup value extrapolated from the test (figure 11) was approximately 11.5 milliseconds for RTE-IV and 12.50 milliseconds for RTE-L.

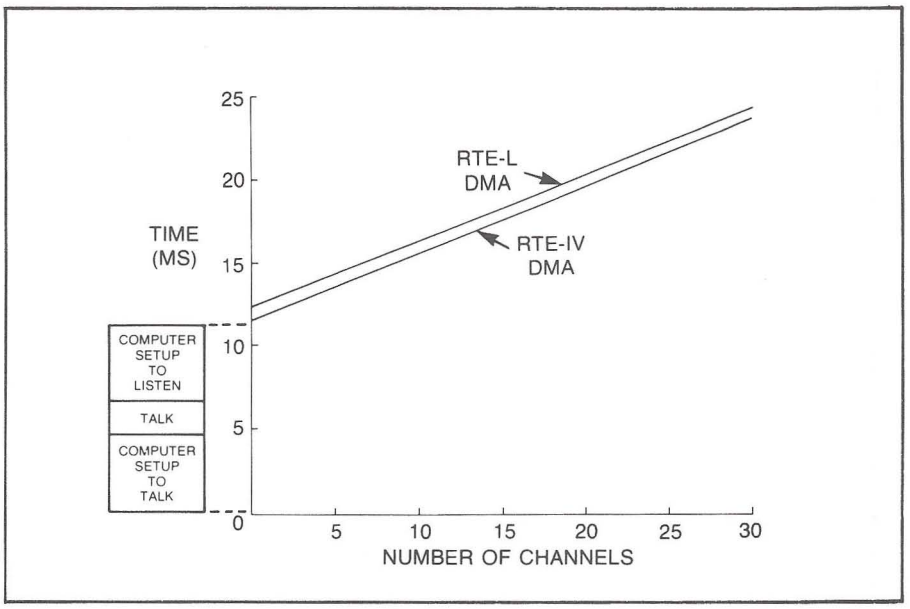


Figure 11. Analog operation with the HP 2240A. The intersection at the "Y" axis represents the total setup times.

Summary/one final example

In every example shown, the time to send or receive a message has been estimated and measured for RTE-IV and RTE-L. In each case, the only external inputs which the model requires are the number of bytes in the message, the speed which the device sends or receives data, and the time for the device to perform its functions, i.e., settling, switching, gating, computing, etc. This information is available in the device manual, or from your instrument supplier. This final example shows how to combine many of the techniques found in this brief for a typical application.

An engineer has a machine with four waste water tanks. The level in each tank must be monitored and logged once each second. To accomplish this task, pressure transducers were installed in the bottom of each chamber. The The 3495A/3455A scanner and digital voltmeter connected to an HP 1000 E-Series computer will be used to monitor the transducers. The 3455A is an interesting choice, because it can be programmed with mathematic coefficients that normalize the readings and return the results in units of level instead of voltage.

In order to calculate the measurement speed and expected computer efficiency, the user will need to know:

1. The number of bytes to program each device and the number of bytes they return.
2. The speed at which the devices can send or receive information.
3. The time required for the devices to complete their function

For this specific example, the user must answer the following questions.

Scanner

- **How do I find the length of the program message?**

  Refer to the scanner manual. Four bytes are used to program the scanner (i.e.,"CO1E")

- **How fast will the scanner receive the message?**

  This information could not be found in the manual but the following assumption could be made. DMA does not begin in an RTE-IV system until the fourth byte is sent. It is inefficient to tie up the resources for only 1 byte, so non-DMA processing will be used. Most devices are not fast enough to take advantage of the high speed curve (B), so it is assumed that the slower rate will apply. Table 1 shows that the setup time will be 7.61 milliseconds and the transfer time will be 4x.56 = 2.24 milliseconds.

- **How long does it take for the scanner to settle?**

  Again, referring to the manual we found that the relay takes less than 10 milliseconds to settle.

- **Does the scanner talk back to the computer?**

  No, it doesn't, so no consideration for a talker is needed.

Voltmeter

- **How long will it take to trigger the DVM?**

  The instruction manual shows that we are allowed to trigger the DVM with a "T3" command. By consulting with a 3455A representative, we found that the message transfer rate for the DVM is 5,000 bytes per second. Since only two bytes are used, a non-DMA transfer will occur. The fast non-DMA transfer rate requires 17,000 bytes per second. So again, we will use the slow non-DMA output equation from Table 1. This gives a setup time of 7.61 milliseconds. The transfer time will be 2 x .56 = 1.12 milliseconds.

- **How long will the DVM require to perform its function?**

  Again, we learned that from the time that the DVM receives the trigger until it is ready to talk (in the low resolution d.c. mode) typically takes 34 milliseconds.

- **How long will it take for the DVM to talk to the computer?**

  The manual states that the data message is 14 bytes plus carriage return/linefeed. Since we discovered that the transfer rate is 5,000 bytes per second, Table 1 gives us the setup time of 6.25 milliseconds and a message transfer time of 16 bytes x 1/5000 seconds/byte = 3.2 milliseconds.

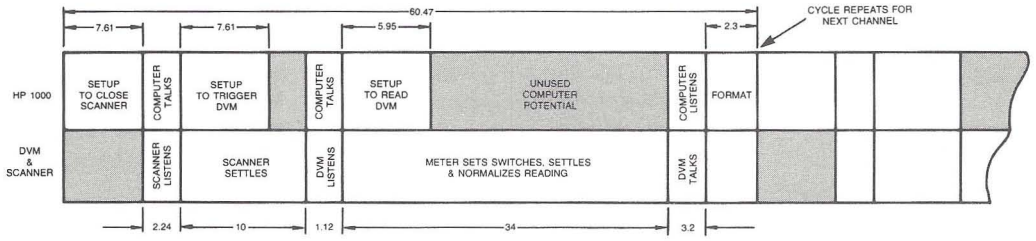
- **Should the data be formatted?**

  Since limit checking will be performed, it should be formatted.

- **How long will it take to format the data?**

  Page 3 gives the formatting equation as t = [0.69 + .11(14)] =2.23 milliseconds.

Figure 12 shows how this information was put together to estimate 1 channel. The total time and utilization is obtained by multiplying by 4. The model estimated that it would take 241.68 milliseconds with 120.12 milliseconds (50%) of computer utilization. It actually took 255 milliseconds with 120 milliseconds of utilization.



STEP	HP 1000	3455A & 3495A	TOTAL TIME
COMPUTER SENDS MESSAGE TO CLOSE SCANNER CHANNEL			
• SETUP	7.61		7.61
• SEND MESSAGE — SCANNER LISTENS (4 BYTES)	2.24	2.24	9.85
SCANNER SWITCHES SETTLE		10	19.85
COMPUTER SENDS MESSAGE TO PROGRAM AND TRIGGER DVM			
• SETUP	7.61		
• SEND MESSAGE — METER LISTENS (2 BYTES)	1.12		20.97
DVM SETTLES		34	54.97
COMPUTER RECEIVES READING FROM DVM			
• SETUP	5.95		
• RECEIVE VALUE — DVM TALKS (14 BYTES)	3.2	3.2	58.17
FORMAT VALUE	2.3		60.47
TOTALS	30.03		60.47

Note: All times in milliseconds

$$\% \text{ UTILIZATION} = \frac{\text{CPU TIME}}{\text{TOTAL TIME}} = \frac{30.03}{60.47} = 50\%$$

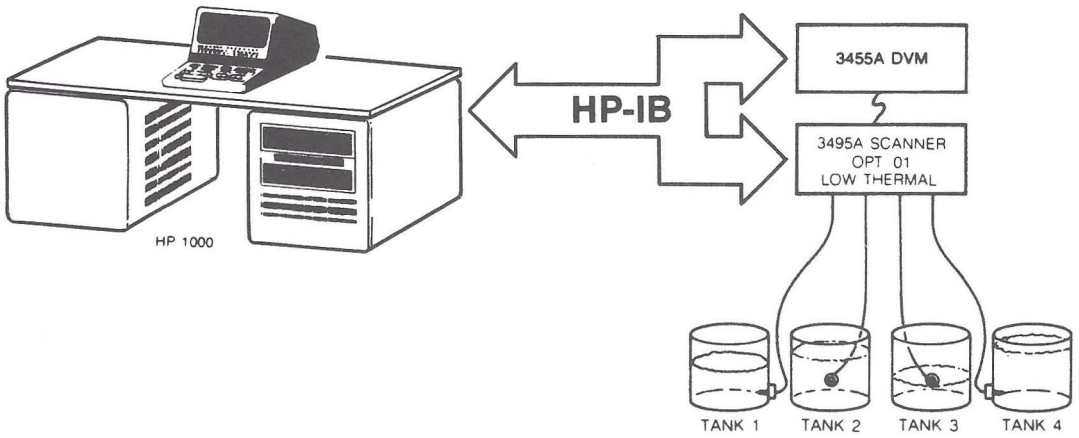


Figure 12. Model for HP 3455A Digital Voltmeter and HP 3495A Scanner with Option 1 installed for fluid level example.

Service request processing

In a programmable HP-IB system, the controller determines who should talk and who should listen. There are times, however, when an instrument needs to let the controller know that it must be attended to by the controller. The instrument can do this by asserting the HP-IB service request line, SRQ. The time for the controller to respond to SRQ assertions is an important performance factor in many applications.

The usual cases for SRQ being asserted are when an instrument error has occurred, or the instrument has been told to indicate when data is available. This latter case can be used to improve system efficiency for slow measurement rates or long communication delays. An example of this SRQ usage would be for a counter waiting for an external event to occur to initiate a measurement, and the timing of the event is unknown. When the event occurs, and the measurement is taken, the counter can then assert SRQ to indicate that a value is available. This process is much cleaner and uses less of the computer's resources than if the computer continuously interrogates the counter for completion, or if an infinite time-out is specified. The usage of an infinite time-out also presents the drawback of preventing time-out handling for any other device on the same bus (EQT) in RTE-IV.

When SRQ is asserted, the controller must determine who needs service, and why. The requestor is determined by a process referred to as a serial poll. During a serial poll, the controller sequentially interrogates each configured instrument, asking for a status byte. The status byte contains one bit indicating whether SRQ is pending. The other bits are used to indicate device-specific status information. For example, a typical instrument could return a status byte in which bit 7 is the SRQ indicator, bit 6 indicates data is available to be read, and bits 1 through 5 indicate problems within its various modules.

When the instrument requiring service is isolated, the HP 1000 Computer responds by scheduling a program for execution that was previously selected. The mechanics of the SRQ response, serial poll, and scheduling are conducted by the HP-IB software, and are transparent to the user program.

The time required to respond to an SRQ varies, depending on the number of devices to poll, the response rate of the devices involved, and the state of the bus when the SRQ occurs (if an HP-IB I/O task is currently being performed). This makes the prediction of an absolute SRQ response time differ from application to application.

A good prediction of SRQ response can be obtained by measuring the time from the SRQ assertion until the serial poll has been enabled (SPE). This represents the interaction and response elements dependent on the computer. The times from SRQ assertion to SPE measured in a quiescent system are:

HP 1000 L-series	1.10 milliseconds
HP 1000 M-series	1.65 milliseconds
HP 1000 E-series	0.77 milliseconds
HP 1000 F-series	0.77 milliseconds

When a long data transfer is in progress, the processing of an SRQ will be held off until the data transfer completes. Also, if a device hangs up, processing of the SRQ will be held off until the time-out occurs. Note that if an infinite time-out value is assigned to a device on the bus, and the device hangs up, SRQ processing of the entire bus could be foiled. Therefore, we care to assign realistic time-out values if normal SRQ processing is desired.

There are times when response time for an SRQ can be critical, and disruption of another I/O task is permissable. Aborting a plotter when a nuclear process is out of limits is an extreme example. An advanced feature of the HP 1000 M, E, and F-Series computers provides for this capability. The task in progress can be aborted, and the SRQ response time will be the same as the time in a quiescent system. Application note 401-1, HP1000/HP-IB Programming Procedures, discusses the use and programmming of this feature (the "S" bit). Remember, most I/O tasks cannot be recovered when aborted, so avoid indiscriminate use of this capability.

An application program using SRQ should contain three elements. These are:

1. Setup and arming of the SRQ
2. The SRQ occurrence
3. SRQ disarming and cleanup.

Figure 13 shows a pair of programs which demonstrate service request scheduling. One program, called SETUP in this example, is used to setup a voltmeter and prepare the system to receive and process an SRQ. The statement in line 6 prepares the system to schedule a program called EVENT when the SRQ is asserted. Then SETUP goes on to other tasks. In this example, a message will be repeated until data is available. Notice that the data linkage between the two programs will be system common.

When the SRQ is received, program EVENT is scheduled. It has a higher priority than SETUP (40 vs. 99), it will execute before SETUP is allowed to continue. EVENT will retrieve the status of the device that caused EVENT to be scheduled with a call to the routine RMPAR. The various bits of the status byte could be checked at this point if the meaning of the SRQ is unclear. In this case, the status will be printed (line 9). Then, the data value will be read. Since VALUE is in system common, the data is also available to program SETUP.

When SRQ processing is completed, program SETUP will resume. SRQ scheduling should then be disarmed to prevent further scheduling of the EVENT routine, and eliminate unnecessary polling of the device (line 29).

```
0001  FTN4,L
0002        PROGRAM SETUP (3,99)
0003        INTEGER IPRG(5),IP(5)
0004        COMMON VALUE
0005        DATA IPRG/5,2HEV,2HEN,2HT /
0006        VALUE=0.0
0007  C     SET UP PROGRAM EVENT AS THE ALARM PROGRAM
0008        CALL SRQ(26,16,IPRG)
0009  C     PROGRAM METER TO MAKE MEASUREMENT AND ASSERT SRQ WHEN DONE
0010        WRITE(26,101)
0011  101   FORMAT("F5R7T3H1D1")
0012  C     NOW DO SOMETHING UNTIL SRQ OCCURS
0013  C
0014  C     THIS PROGRAM WILL STAY IN THE FOLLOWING LOOP UNTIL A VALUE
0015  C     IS RETURNED THROUGH SYSTEM COMMON.
0016  C
0017  44    IF(VALUE.NE.0.0) GO TO 55
0018        WRITE(1,102)
0019  102   FORMAT("WAITING FOR SRQ")
0020        GO TO 44
0021  C     WHEN SRQ OCCURS AND THE VALUE APPEARS, THE PROGRAM WILL
0022  C     RESUME HERE
0023  55    WRITE(1,103) VALUE
0024  103   FORMAT("VALUE IS " F12.6)
0025        WRITE(1,104)
0026  104   FORMAT("ALL DONE")
0027  C
0028  C     RESET SRQ PROCESSING
0029        CALL SRQ(26,17,0)
0030        STOP
0031        END


0001  FTN4,L
0002        PROGRAM EVENT (3,40)
0003  C     THIS PROGRAM IS SCHEDULED WHEN AN SRQ OCCURS
0004        INTEGER IP(5)
0005        COMMON VALUE
0006  C     GET STATUS PARAMETER
0007        CALL RMPAR(IP)
0008  C     DISPLAY STATUS
0009        WRITE(22,101) IP(1)
0010  101   FORMAT("EVENT STATUS = ", O6)
0011  C     READ VALUE FROM VOLTMETER AND STORE IN COMMON
0012        READ(63,*) VALUE
0013        STOP
0014        END
```

Figure 13. Sample SRQ Program

Time-out processing

A time-out error is defined as the failure of an I/O device to complete a task within a pre-specified period of time. Time-out errors are not always bad, though. Sometimes, they can be used to indicate end of data, or that a device is busy.

Time-out errors are handled in one of two ways. The default method is for RTE processing of the time-out error. The program requesting the I/O task is placed in the I/O wait state, and the LU of the device timing-out will be set down. The operator must intervene to restore operation by correcting the error situation, and entering the "UP" operator command.

The second method for handling time-out errors is for the user program to handle the error itself. RTE allows this if the user program requests it. When the user program processes its own errors, the device LU will not be set down, and the program will be allowed to continue execution. Note that this means the user program must obtain the the interface status after each I/O task to insure proper completion.

User processing of time-out errors consists of four steps.

1. Establish the desired time-out value. This is the time, in tens of milliseconds, to wait before deciding that the the device has timed-out. Most instruments have a settling or gating time which must be fulfilled before the instrument can respond. Make sure that a proper amount ot time is provide to allow for this.

   The time-out command is similar in structure for both RTE-IV and RTE-L, but the parameters are different. In RTE-IV, the time-out value is set for the EQT of the HP-IB bus, and encompasses all devices on the bus. In RTE-L, the time-out value can be set on an individual LU-for-LU basis. The FMGR command to set the time-out value in RTE-IV is the :SYTO command. This command in RTE-L was shortened to :TO. For example, the command:

   ```
   :SYTO,7,100
   ```

   will set all devices on EQT 7 (RTE-IV) to one second. The FMGR command:

   ```
   :TO,26,50
   ```

   will set the time-out value for LU 26 (RTE-L) to 500 milliseconds.

   Care should be taken if time-out processing is not needed and the time-out value is set to zero. When the time-out value is set to zero, the device will never time-out (infinite time-out value). A device with an infinite time-out value can be the cause of the bus hanging up if an SRQ occurs. Also, SRQ processing can be prevented if a device is hung up with an infinite time-out value. RTE waits for the I/O task to complete, and since the completion cannot occur, the SRQ servicing will be held off.

2. Tell the operating system that you want to handle your own time-out errors. From your user program, use the HP-IB library subroutine CNFG to set the "E" bit. The CNFG subroutine is also the same command needed to request DMA operation in RTE-IV. Therefore, you are able to configure everything needed in one command. For example, the statement:

   ```
   CALL CNFG(26,1,37400B)
   ```

   will turn on DMA for LU 26, and tell the operating system that you want to handle your own errors. For more specific information on HP-IB configuration, refer to the HP-IB Users Guide (HP part number 59310-90064).

3. After each I/O task, check to see if it was completed properly. The HP-IB function IBERR should be used. If the value returned by the function is zero, the task completed normally. If the value returned was 1, a time-out error has occurred.

   The IBERR function can be placed in an IF statement. Doing this permits your program to test the result of the I/O task, and to jump to a time-out processing routine if needed. For example, the statement:

   ```
   IF(IBERR(26).EQ.1)GOTO81
   ```

   will cause the program to jump to statement label 81 if a time-out has occurred for the device on LU 26.

4. Reset the user error bit to turn off user handling of errors when you are done. Otherwise, RTE will not process time-out errors for the device in future programs. The right to handle your own errors also give you the responsibility to restore things for others when you are through. The CNFG subroutine should be used to reset the user error bit.

The program in figure 14 illustrates how a time-out can be trapped and processed by a user program. If the LU entered times-out, the message shown in line 81 will be displayed. If desired, this is where a more sophisticated time-out routine would be included.

```
FTN4,L
      PROGRAM TMOUT
C     THIS IS AN EXAMPLE PROGRAM TO DEMONSTRATE THE USE OF
C     INSTRUCTIONS WHICH CAN CONTROL THE THREE PHASES OF TIME-OUT
C     PROCESSING FOR HP-IB.  THE THREE PHASES ARE:
C
C        (1) SETTING THE TIME-OUT VALUE .
C        (2) SETTING THE DEVICE CONFIGURATION WORD TO ALLOW
C            USER PROCESSING OF THE TIME-OUT.
C        (3) CHECKING FOR TIME-OUTS AND PROCESSING THE ERROR.
C
C     *************************(1)*********************************
C
      INTEGER IP(5),IBUF(5),OS
C     GET USER TERMINAL NUMBER AND FIND OUT WHICH OPERATING SYSTEM
C     IS BEING USED. IT IS STORED IN PARAMETER IP(1).
      CALL RMPAR(IP)
      CALL OPSY(OS)
C     NOTE THAT OPSY IS A USER WRITTEN ROUTINE FROM THE APPENDIX OF
C     THIS NOTE.  IF OS=-9, THE SYSTEM IS RTE-IV.  IF OS=-31, THE
C     SYSTEM IS RTE-L.
      ILST=IP
      IF(ILST.EQ.0) ILST=1
C     ASK QUESTIONS TO SET TIME-OUT VALUE. THE FIRST IS FOR
C     RTE-IV, AND THE SECOND PART IS FOR RTE-L.
      IF(OS.EQ.-31) GO TO 31
      IF(OS.NE.-9) GO TO 98
      WRITE(ILST,101)
  101 FORMAT("ENTER EQT OF HP-IB")
      READ(ILST,*) IBUF(3)
      WRITE(ILST,102)
  102 FORMAT("ENTER TIME-OUT VALUE (IN TENS OF MS.)")
      READ(ILST,*)IBUF(5)
C     SET NEW TIME-OUT VALUE FOR HP-IB EQT
      IBUF(1)=2HTO
      IBUF(2)=2H,
      IBUF(4)=2H,
      INUM=10
      CALL MESSS(IBUF,INUM)
      WRITE(ILST,103)
  103 FORMAT("ENTER HP-IB DEVICE LU")
      READ(ILST,*) IDLU
      GO TO 44
C
C     RTE-L SECTION
31    WRITE(ILST,103)
      READ(ILST,*) IDLU
      WRITE(ILST,102)
      READ(ILST,*) ITIM
      CALL EXEC(3,2200B+IDLU,ITIM)
C     ************************(2)*********************************
C
C     SET THE DEVICE CONFIGURATION WORD
C
44    CALL CNFG(IDLU,1,17400B)
C
```

Figure 14. Time-Out Example

```
C     ************************(3)*******************************
C     THIS SECTION SHOWS HOW A USER PROGRAM CAN INTERPRET AND
C     PROCESS A TIME-OUT.  THE LIBRARY ROUTINE IBERR IS USED TO
C     DETERMINE THE OUTCOME OF AN HP-IB I/O TASK.  IN THIS EXAMPLE,
C     A READ REQUEST WILL BE DONE TO A DEVICE THAT IS SHUT OFF.
C     REMEMBER TO DISCONNECT THE DEVICE TO OBSERVE THE TIME-OUT!
C
      READ(IDLU,*) A
      IA=IBERR(IDLU)
C     CHECK ERROR CODES
      IF(IA.EQ.0) GO TO 80
      IF(IA.EQ.1) GO TO 81
C     IF NEITHER CODES OCCURRED, IT IS SOME OTHER TYPE OF ERROR.
      WRITE(ILST,105)
  105 FORMAT("A NON-TIME-OUT ERROR HAS OCCURED")
      GO TO 99
C     IF IA WAS 0, EVERYTHING WAS ALL RIGHT.
   80 WRITE(ILST,106)
  106 FORMAT("NORMAL COMPLETION")
      GO TO 99
C     IF IA WAS 1, A TIME-OUT OCCURRED.
   81 WRITE(ILST,107)
  107 FORMAT("A TIME-OUT OCCURRED")
      GO TO 99
C     THIS IS THE ERROR MESSAGE FOR AN UNKNOWN OPERATING SYSTEM
   98 WRITE(1,108)
  108 FORMAT("OPERATING SYSTEM UNKNOWN")
   99 STOP
      END
```

Figure 14. Time-Out Example (continued)

In the previous sections of this Brief, the performance of the HP-IB running under RTE was presented. At times, the system was CPU-bound (the device was waiting for the CPU) and at times it was I/O-bound (the CPU waiting for the I/O device). RTE as a resource allocator tries to balance the mix, but bottlenecks can occur. The bottlenecks can be located, however, and there are techniques to reduce them.

The ACCEL/1000 RTE Profile Monitor (RPM) is a software package which samples and analyzes the activity of any program executing under the RTE-IV operating system. The RPM program will indicate where the program spends most of its time. Once this is known, steps can be taken to reduce these critical paths. Real-time I/O bound programs can only be helped by faster I/O devices, but CPU-bound situations can be improved.

Direct I/O calls

Auto addressing is simple, but the price paid for its simplicity is increased execution overhead time. If high speed performance is needed, write all HP-IB functions as direct I/O calls to the bus. Direct I/O gives an approximately 25% increase in performance for typical HP-IB functions. For example, the scanner/voltmeter example for the water tank took only 95 milliseconds of utilization with direct I/O versus 120 with autoaddressing.

Formatter

The RTE formatter is a general purpose formatting and translation routine. It must include the necessary error checking and coding to make it flexible and general purpose. The price paid for the flexibility is an increase in execution time and memory space for the routine. If a device is to be used repeatedly, a device specific formatting routine could be written to take up less memory and execute faster. A device specific formatter for the 3455A can reduce the time for translation from 3 milliseconds to under a millisecond.

Microcode

Further speed and overhead improvements can be made in an RTE-IV system by rewriting bottleneck routines in microcode. User microcode can give up to 10 times improvement in execution time. A simple formatter would be a good candidate.

Possibilities also exist for other improvements (a less flexible, more device specific driver), but consider the tradeoffs. Changes and modifications to software can be very expensive in terms of labor, documentation, and test time. If the improvement is for production (used repeatedly), there can be a substantial return against cost from increased performance. However, if the application is "one-shot" in nature, the general purpose HP-IB driver and formatter should be used.

Appendix

There are times when it is useful for a program to determine which HP 1000 operating system it is using. This way, a program can be written to run on various systems, with the program adapting itself to the system without the operator intervening. The program shown below is an assembly language subroutine which will return the identification number of the various RTE operating systems. It is called from FORTRAN as shown in the listing.

```
ASMB,L,R,F
*      THIS IS A SUBROUTINE TO DETERMINE WHICH OPERATING
*      SYSTEM IS BEING USED.  THE FORTRAN CALLING SEQUENCE IS:
*
*      CALL OPSY(I)
*
*      THE VALUE RETURNED IN I IS:
*      +------------------+-------+
*      I OPERATING SYSTEM I VALUE I
*      +------------------+-------+
*      I RTE-MI           I  -7   I
*      I RTE-MII          I  -15  I
*      I RTE-MIII         I  -5   I
*      I RTE-II           I  -3   I
*      I RTE-III          I  -1   I
*      I RTE-IV           I  -9   I
*      I RTE-L10          I  -31  I
*      I RTE-L20          I  -29  I
*      +------------------+-------+
*
       NAM    OPSY
       EXT    .ENTR,$OPSY
       ENT    OPSY
VAL    BSS    1
OPSY   NOP
       JSB    .ENTR
       DEF    VAL
*      GET VALUE OF OPERATING SYSTEM
       LDA    $OPSY
       STA    VAL,I
       JMP    OPSY,I
       END
```

HEWLETT PACKARD

Printed in U.S.A. 10/80 (22) 5953-4274