

HP 2250 Measurement
and Control Processor

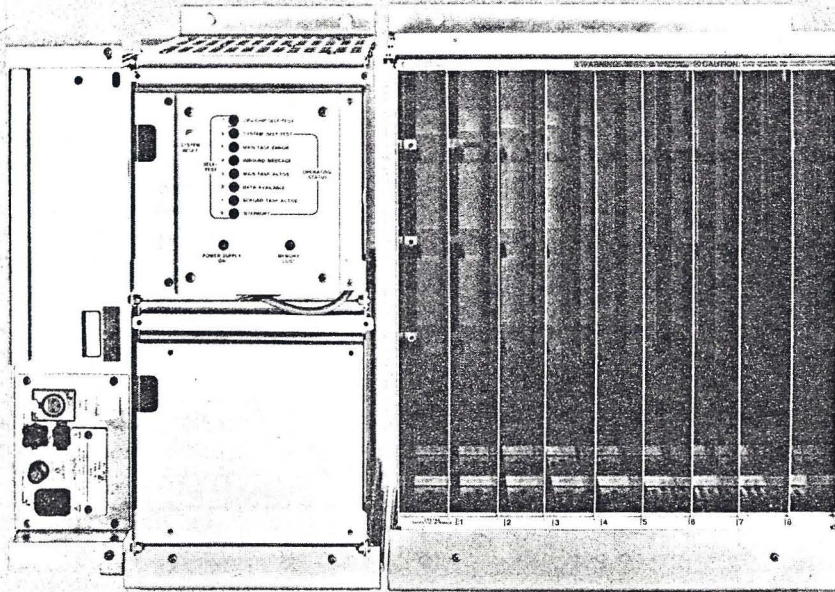


HP 1000
Computer



Programming Guide

Application Note 401-24



Introduction

The HP 2250A Measurement and Control Processor is an intelligent, high-performance analog/digital subsystem designed to handle a wide range of industrial and laboratory automation applications. The HP 2250A is programmed in HP-MCL/50, an easily-learned language which provides the user with the capabilities to customize memory allocations, write custom data-reduction routines, implement multiple decision-making control and supervisory tasks, and respond to real-world schedules and events. FORTRAN and ASSEMBLER subroutines can be written on an HP 1000 computer and down-loaded to the HP 2250A for execution, thus minimizing I/O overhead and dependence on the host computer.

The HP 2250A uses the Hewlett-Packard Interface Bus (HP-IB) to communicate with its host controller, which can be any of the HP 1000 Computers and the 9800- Series Desktop Computers. A built-in "Watchdog" timer can be used to monitor communications with the host and produce an interrupt if no bus communication occurs within a specified period of time. The standard HP-IB cables can be replaced with up to 1000 metres of fibre-optic cable by using the HP 37203A HP-IB Extender, if the length of the bus or an electrically noisy environment is a problem.

Figure 1 shows a typical configuration for an HP 2250A. The system is composed of a "controller section" and a "Measurement and Control Unit", or "MCU". The controller section coordinates the processing activities of the HP 2250A, and the MCU handles the interfacing to the real world. Each Measurement and Control Unit can contain up to eight function cards. Up to 8 MCUs can be connected to a single controller section, and so each HP2250A can support up to 64 function cards.

The analog subsystem of the HP2250A can supply 14-bit analog samples to the host computer at continuous rates up to 50,000 samples per second. Conversion and linearization algorithms for thermocouple measurements are provided in firmware. The HP2250A can monitor up to 240 analog channels per Measurement and Control Unit (MCU), and this can be extended to 1920 channels by using the maximum of 7 HP2251A Extenders. Table 1 describes the analog family of function cards.

The digital subsystem in the HP2250A can interface to all commonly found sensing and actuating devices, using signal conditioning modules to convert the external signals to internal logic levels. Typical applications include controlling actuators, stepper motors, and motor contactors, and measuring periods and frequencies of waveforms. Table 2 describes the family of digital function cards.

Table 1. ANALOG Function Cards

Function Card	No. Of Chan.	Range(s)	Gain	Samples Per Second		Purpose Of Card
				On-Channel	Scan	
Analog/Digital Converter (25501A)	16	$\pm 1.25V$ (min) $\pm 10.0V$ (max)	1 2 4 8	50,000	50,000	Convert analog input signal to digital representation. Resolution = 14 bits (.006%)
High-Level Solid-state Multiplexor (25502A)	32	$\pm 1.25V$ (min) $\pm 10.0V$ (max)	1	50,000	50,000	Extend the 25501A ADC by 32 input channels. Prerequisite — 25501A
Low-Level Solid-state Multiplexor (25503A)	32	$\pm 12.5 mV$ (min) $\pm 100V$ (max)	1, 10, 100	50,000	20,000	Extend 25501A ADC by 32 input channels and provide gain. Prerequisite — 25501A
Relay Multiplexor (25504A)	16	$\pm 12.5 mV$ (min) $\pm 100V$ (max)	0.1, 1, 10, 100	10,000	1,000	Extend the 25501A ADC by 16 input channels, provide gain and electrical isolation. Prerequisite — 25501A
Digital/Analog Converter (25510A)	4	$\pm 10.24V$ (bipolar) 0, +10.24V (unipolar) 0, 20.5 mA (at 20V)	1	N/A	30,000	Provide analog voltage and current output. Resolution = 5 mV

Table 2. DIGITAL Function Card Summary

Function Card	No. Of Channels	Purpose Of Card
Digital Input (25511A)	32	Provide digital inputs for monitoring AC or DC signals (0-120 VDC, 0-230 VAC) through required signal conditioning modules (SCMs). Inputs can be monitored individually or collectively and can generate interrupts upon transitions. Maximum Read frequency = 24 KHz.
Counter Input (25512A)	4	Provide independently-programmable counters which totalize, count up or down, and measure periods, time intervals, and frequencies. Interrupts can be generated upon overflow or completion of counting. Maximum measurable frequency = 500 KHz.
Digital Output (25513A)	32	Provides solid-state switching of AC and DC loads of up to 60V (peak) at 300 mA and zero-voltage switching up to 120 VAC at 800 mA through required signal conditioning modules. Outputs can be programmed individually or as 2 16-bit fields. Maximum output rate = 40 KHz.
Relay Output (25514A)	16	Provides switching of high-current loads (up to 2A at 30 VDC or 3A at 125 VAC). Signal Conditioning Modules suppress transients to protect the relays and prevent noise. The 16 channels can be programmed individually or collectively. Maximum operating speed = 30 Hz.
Pulse Generator (25515A)	4	Provides digital pulses of programmable frequency, width, and acceleration for controlling stepper motors. Limit switch inputs on this card can be programmed to abort pulse trains. Maximum pulse rate = 20 KHz.
Multifunction (25516A)	16 input, 16 output	Provides independently-programmable digital inputs and outputs, counting and interrupt capabilities. SCMs provide a wide range of interfacing for monitoring and controlling transducers, instruments, and switches.

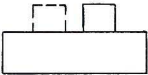
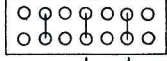

Addressing

There are two types of HP-IB addressing used with the HP 2250A : primary addressing and secondary addressing. Secondary addressing is discussed in the upcoming "PROGRAMMING" section.

The host computer uses primary addresses to distinguish between separate devices connected to the same HP-IB interface card. The primary address, also called the "device address", is set on the HP 2250A's HP-IB card, and is related through the tables in the host computer's operating system to a logical unit number (an LU). All programmatic access to the HP 2250A is conducted through this LU. The switches which determine the HP-IB device address for the HP 2250A are shown in Figure 2.

Switch Settings on the HP 2250 Hardware

There are several switches on the hardware of the HP2250A which need to be set before it can be used. These include the HP-IB address and select-code switches, the battery-backup/power restart selector, and the Backplane Interface (BIF) selector. In addition, if there are any analog output cards, then some application-dependent settings will be required. Consult the "HP2250A Installation and Start-up Manual" for specific instructions on setting these switches. The sample application program in the "PROGRAMMING" section uses the HP2250A with the switches in the positions shown in Figure 2.

HP 2250A Controller Card	Switches to set on the card	
<p>HP-IB Hewlett-Packard Interface Bus</p> <p>12009-60001</p> <p>Controller Slot 5</p>	<p>Switch U1</p> <div style="border: 1px solid black; padding: 2px; display: inline-block;">0 1 0 1 1 0 0 0</div> <p>1 2 3 4 5 6 7 8</p> <p>value = 130 octal</p> <p>Select Code for 2250A Backplane</p>	<p>Switch U16</p> <div style="border: 1px solid black; padding: 2px; display: inline-block;">0 0 0 0 0 0 0 1</div> <p>1 2 3 4 5 6 7 8</p> <p>value = 1 octal</p> <p>*HP-IB Device Address</p>
<p>MCI Measurement and Control Interface</p> <p>12071-60001</p> <p>Controller Slot 4</p>	<p>Switch U1</p> <div style="border: 1px solid black; padding: 2px; display: inline-block;">0 0 0 1 1 0 0 1</div> <p>1 2 3 4 5 6 7 8</p> <p>value = 31 octal</p> <p>Select code for 2250A Backplane</p>	
<p>CPU Central Processing Unit</p> <p>12001-60001</p> <p>Controller Slot 3</p>	<p>Switch U1</p> <div style="border: 1px solid black; padding: 2px; display: inline-block;">0 1 0 0 0 0 0 1</div> <p>1 2 3 4 5 6 7 8</p> <p>*Battery-Backup & Boot Address</p>	
<p>RRACK Firmware for MCL/50 plus User Memory</p> <p>12070-60001</p> <p>Controller Slot 2</p>	<p>BATT NORM</p>  <p>*Battery-Backup Switch</p>	 <p>3C 64K</p> <p>Jumpers</p>
<p>BIF Backplane Interface</p> <p>25544-60001</p> <p>MCU Slot 0</p>	 <p>BIF number (Rotary dial)</p>	

*Denotes a setting which is application-dependent
Key: 1 = Open, 0 = Closed

Figure 2. Sample Switch Settings for HP 2250A BIF and Controller Cards

Each function card is referenced in MCL commands by a unique "slot number". Slot numbers are related to the physical position of the function card in the Measurement and Control Unit (MCU), and to the setting of a rotary switch on the "Backplane Interface" card, as shown in Figure 3. Each MCU uses a "Backplane Interface (BIF)" to communicate with the controller section of the HP2250A. Each BIF card

has a unique "BIF number" which is used to determine the slot number of each card connected to it, and this BIF number is set by a rotary switch on the card. The BIF cards should be numbered sequentially from 0 (the BIF in the HP2250A) to up to 7 (depending on how many HP2251A Extenders are included in the system).

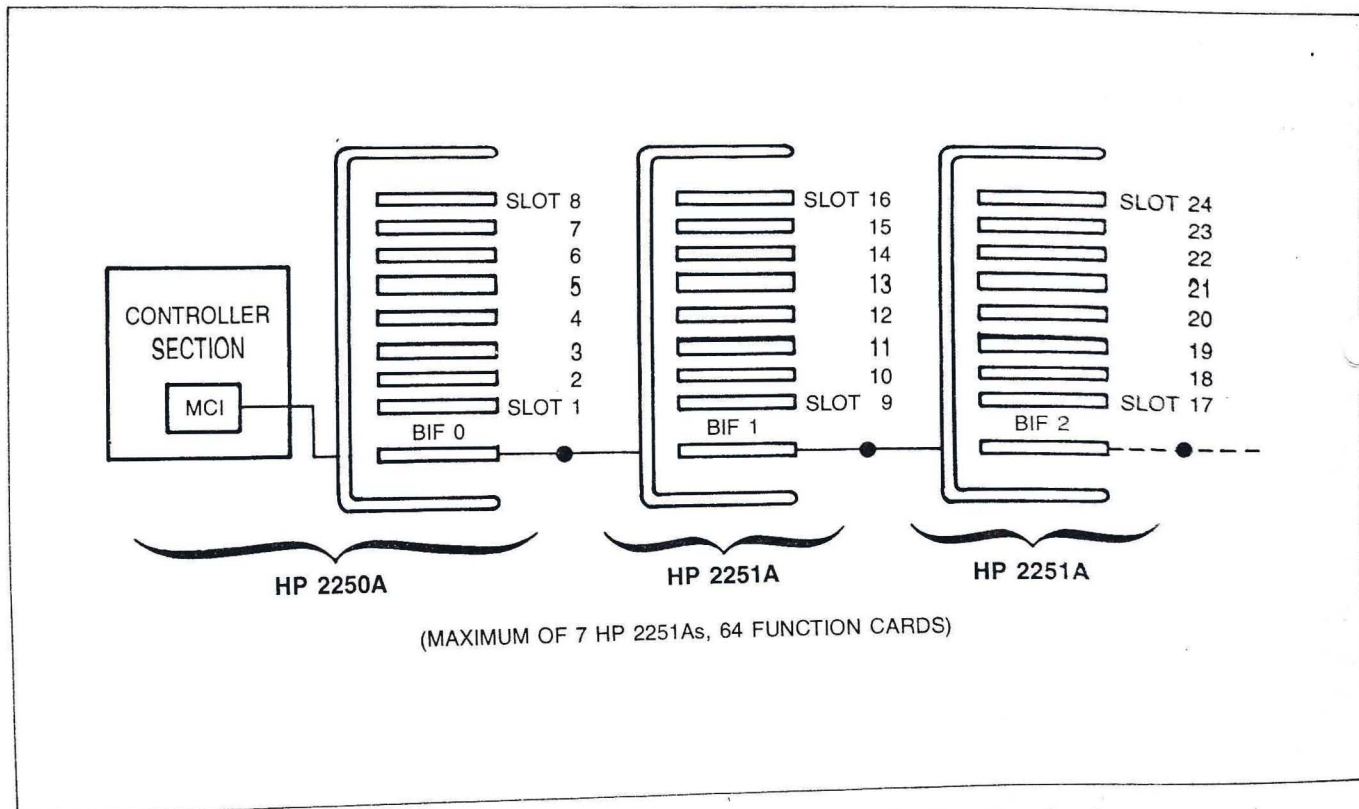


Figure 3. How the Slot Numbers are determined for the Function Cards

System Preparations

LU Assignments

All programmatic access to the HP2250A is conducted via a logical unit number (LU) which is assigned by the user to the HP2250A when it is first hooked up to the host computer system. The LU is related back to the unique device address which is set on the HP2250A's HP-IB interface card. (See Figure 2.) How this relation is accomplished depends upon which computer is being used.

RTE-IVB: A logical unit can be assigned to the HP2250A with the following FMGR command (FMGR is the name of the operator interface for all HP1000 Computers):

```
:SYLU,11,25,1
```

This will assign LU 11 to equipment table number (EQT) 25, subchannel 1. Each bus on the M-, E-, and F-Series computers has a unique equipment table number which is assigned to the bus when the operating system is generated. Each device on the bus has its own subchannel of the bus EQT, and this subchannel number corresponds exactly to the HP-IB device address, described above.

RTE-L: A logical unit number can be assigned to the HP2250A with the following FMGR commands:

```
:LA,11,25
:CN,11,AD,1
```

This will assign the HP2250A to device table 25 and logical unit 11, assuming that the device address of the HP2250A is set to 1.

HP-IB Configuration

User error-processing

The user has several choices to make when configuring an HP-IB device, such as an HP2250A. The first choice to make is whether to process HP-IB errors within the application program or to let the RTE operating system suspend the program whenever an HP-IB error occurs. Such errors include device time-outs and illegal bus control signals. RTE-L and RTE-IVB handle user error-processing in the same manner. If programmatic error-checking is desired, the system function IBERR should be used within the application program after every I/O operation with the HP2250A to determine whether or not the transaction completed successfully. An example of the use of IBERR in programmatic error-checking is shown in the sample program in the "PROGRAMMING" section.

DMA

The second decision involves the use of Direct Memory Access on the host computer side.

RTE-IVB: The HP1000 M-, E-, and F-Series offer the user a choice of whether or not to use DMA with HP-IB devices. DMA provides much higher transfer rates than "programmed I/O" (non-DMA), but it requires a longer set-up time as well. Generally, if the transfers exceed 50 characters in length, then the use of DMA will be advantageous. The HP1000 M-, E-, and F-Series computers offer two DMA channels, so the anticipated overhead for using DMA should include waiting for a DMA channel if more than two devices (disks, magnetic tape drives, other HP2250As, etc.) ever need DMA simultaneously.

RTE-L: DMA is always allocated for all I/O transfers with the HP1000 L-Series, so no user configuration is required for DMA usage.

EOI

The End-or-Identify control line (EOI) is used in HP-IB transactions to signal the end of a transmission. There is no standard sequence of signals and/or ASCII characters to denote the end of the message for all HP-IB devices. Some use control characters such as carriage return and linefeed, some merely stop after a certain number of characters goes by, and some depend upon the EOI signal for ending indications. On input, the HP2250A requires the host computer to assert the EOI line (set it true) with the last data byte that the host sends or with a control character (carriage return or line feed) after the last byte is sent. On output, the HP2250A will assert the EOI line with the last byte sent.

RTE-IVB: The EOI conditions mentioned above are programmable and are specified in the configuration word as shown in Figure 3 at the end of this section. The default settings of these conditions are appropriate for the HP 2250A.

RTE-L: The EOI conditions are not programmable on the L-Series. The EOI requirements for the HP2250A and the L-Series are compatible, so no user adjustment is necessary.

SRQ Priorities

One final question must be answered in order to complete the HP-IB configuration, and it involves how the host computer will respond to receiving an SRQ (service request) from a device on the bus. The HP1000 always initiates a serial poll when it notices that the SRQ signal line has been asserted by a device on the bus.

HP 2250A/HP 1000

RTE-IVB: The user has the choice of letting an ongoing I/O transaction complete before having the computer initiate the serial poll, or aborting the I/O and starting the serial poll immediately. If the I/O requests are to be aborted in response to an SRQ, the user also has the choice of having the host try to restart the aborted I/O or to abandon it, once the SRQ from the other device has been serviced. Normally, the host will wait for the transaction to complete, because most HP-IB devices (including the HP2250) cannot programmatically recover from an abnormal termination such as this. The sample program in this note does not let SRQs from other devices abort I/O transactions between the host computer and the HP 2250A.

RTE-L: The L-Series always waits for any I/O transactions to complete before servicing an SRQ, so no user configuration is required.

Configuration word

All of the above decisions concerning the HP-IB configuration are made known to the host computer via a "configuration word" which is stored in the operating system table area of the host computer. The format of this word is shown in Figure 4.

RTE-IVB: There are two ways to specify this configuration word: programmatically, using the HP-IB system subroutine CNFG, or from FMGR, using a CN command. Examples of each are shown below. In the sample program included in the "PROGRAMMING" section of this note, the programmatic method is used at the beginning of the program.

Programmatically:

```
CALL CNFG (11, 1, 17400B)
```

where:

11 = LU of the HP 2250A
 1 = Configure the device
 17400B = Configuration word

From FMGR:

```
:CN, 11, 25B, 17400B
```

where:

CN = Control request
 11 = LU of the HP 2250A
 25B = Configure device
 17400B = Configuration word

RTE-L: Only the E-bit of the configuration word shown above is significant in RTE-L. The other bits are masked off and ignored, since their specifications are not user-modifiable. This means that the same programmatic call used in RTE-IVB can be used with RTE-L, but only the E-Bit will be affected. No FMGR command for device configuration is available for the L-Series. Thus, to set up the configuration word described above on the L-Series, use the following FORTRAN program statement:

```
CALL CNFG (11, 1, 400B)
```

where 11 = LU of the HP2250A
 1 = Configure the device
 400B = Configuration word

bit name	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		
value	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0		
octal	0			1			7			4			0			0		

S = 0	Don't abort I/O operations with this device to process an SRQ
R = 0	Do not attempt to restart aborted I/O transactions
D = 0	Do not use DMA
I = 1	Require an EOI from the device at end of transmission
J = 1	Expect EOI to occur with the last byte of the transfer
O = 1	Issue an EOI to the device at end of transmission
P = 1	Issue the EOI with the last byte of the transfer
E = 1	Don't abort programs because of HP-IB errors

Figure 4. Sample HP-IB configuration word for the HP 1000 to be used with the HP 2250A

For more information on configuring HP-IB devices for HP1000 systems, refer to Application Note 401-1 (5953-2800) for the M-, E-, and F-Series, and to Application Note 401-1L (5953-2830) for the L-Series.

Buffering

The next decision involves the buffering of the device. Buffering applies only to transfers from the host computer to an HP-IB device. Normally, the bus operates in the "unbuffered" mode. This means that the host application program will wait for each I/O transfer that it initiates to complete before it will continue its execution. With buffering, the host program does not wait for an I/O transfer to complete before executing its next statement. The data to be sent to the device is instead directed to a buffer in memory and then transferred to the device as fast as the device can accept it. This technique is used primarily to speed the execution of a host program which, by no fault of its own, must send requests to a slow device. The HP2250A, by the way, is by no means a slow device.

Buffering allows I/O requests to stack up in the buffer if the host program sends requests faster than the device can execute them. If the bus hangs up during an I/O transaction with the host, the remainder of the requests in the buffer will not be executed. In addition, buffering prevents the user program from implementing programmatic error checking.

Therefore, buffering with the HP2250A is not recommended.

To specify unbuffered operation, use the following FMGR commands:

RTE-IVB: The FMGR command:

```
:SYEQ, 25, UN
```

if the HP2250A is connected to the bus with equipment table number 25.

RTE-L: The FMGR command:

```
:BL, 25, UN
```

if the HP2250A is connected to the bus with device table number 25.

One final point to remember is that with the M-, E-, and F-Series, the entire bus is either buffered or unbuffered. With the L-series, buffering is done on a per-device basis.

Time-out

The "time-out" value associated with a particular HP-IB device is the amount of time that the computer will wait for a response from the device after the computer initiates an I/O operation with it. The value is specified in tens of milliseconds.

If the HP-IB device does not respond within the specified time, one of two things happens. If user error-processing is not enabled in the HP-IB device configuration, the RTE operating system will suspend the program which encounters the time-out and make the device unavailable until the operator corrects the problem. If user error-processing is enabled, RTE will not suspend the user's program, but when the program calls the IBERR subroutine, which is provided with the HP1000 HP-IB Library, it will return an error code of 1 to indicate that a time-out has occurred. The user program can then take appropriate action.

The HP2250A is capable of transferring large amounts of data quite quickly and so a relatively short time-out value is usually appropriate.

When downloading main tasks which take data, the main task error code should be read from the main result buffer along with any data it has accumulated after the task has compiled and executed. The time-out value must then include the time that the HP2250A needs to compile and execute the task. For resident tasks, the task error code should be read from the main result buffer directly after the task has been downloaded and compiled. In this case, the time-out value needs to account only for the compilation time. For most tasks, the compilation time is less than 2 seconds. Execution time is totally application — dependent.

For most applications with the HP2250A, a time-out value of 5 seconds is quite sufficient. To specify a time-out value for an HP2250A, use the following FMGR command:

RTE-IVB: The FMGR command:

```
:SYT0, 25, 500
```

will set the time-out value of the bus assigned to Equipment Table (EQT) 25 to 5 seconds.

RTE-L: The FMGR command:

```
:T0, 11, 500
```

will set the time-out value of LU 11 to 5 seconds.

Note that with RTE-IVB, the time-out value applies to the bus as a whole, while with RTE-L, the time-out value applies only to the individual device. This implies that with RTE-IVB systems, the time-out value should be chosen to suit the slowest device on the bus.

Also note that a time-out value of zero means that the host computer will wait forever for the device to respond. This is useful if the device could possibly require more than 327.67 seconds (the maximum time-out value) to respond to the host computer. However, the zero time-out value also eliminates the ability of a program to determine whether or not a device is properly connected and functioning on the bus. If the host computer is configured to wait for I/O transactions with this device to complete before servicing SRQs from other devices, and this device has a zero time-out value, then an I/O operation with this device which fails to complete will effectively hold off the servicing of SRQs indefinitely.

It is recommended that the host computer program use secondary addressing to determine through system status when to initiate a read operation from the HP2250's main result buffer, instead of letting the read operation "hang" on the HP2250A's LU with a zero time-out.

Programming

MCL/50

The HP2250A is programmed in MCL/50, a measurement and control language composed of over 100 commands and modifiers. MCL/50 provides extensive control of function cards, performs arithmetic and logical operations on data, and allows run-time errors to be handled programmatically.

Many of the MCL/50 commands which control function cards can operate on more than one channel at once. The format of the DO (digital output) command, for example, is shown below.

DO (slot, starting channel, no. channels to output to) data items

To send a digital "1" to channels 2, 3, 4, and 5 of the digital output card in slot number 3, the following MCL/50 command could be used :

```
DD (3, 2, 4) 1,1,1,1
```

Tasks

MCL/50 commands are grouped into "tasks", much as the statements of other languages are grouped into programs. There are two types of tasks : main and resident. There can be many resident tasks loaded in the HP2250A's memory concurrently, but only one main task can exist at one time. Main tasks are used mostly to perform short operations which need to be done only once, such as configuring the HP2250A memory or starting resident tasks. Main tasks need to be downloaded to the HP2250A and compiled for every execution. Resident tasks are used to perform operations which need to be executed repeatedly, such as taking periodic measurements. Resident tasks need to be downloaded and compiled only once.

Main tasks are removed from memory when they complete; resident tasks remain in memory until they are removed by an NTASKS or a RESET command, and so can be executed as often as desired without the need for re-loading them into the HP2250A. Resident tasks have a task number and a priority ranging from 1 to 32767. The default priority for a resident task is 99. A main task automatically has a priority of 0 (the highest possible) and has a task number of zero.

There are no restrictions on the types of commands that can be used to compose a task, but there must be a "!" terminator after the last command in the task. For example, the task in Figure 5 would take a single reading from the first channel of the digital input card in slot 5, send a logical "1" to the first four channels of the digital output card in slot 3, and then terminate.

```
DI ( 5, 1 )
DO ( 3, 1, 4 ) 1,1,1,1
!
```

Figure 5. A sample task for the HP 2250A

All tasks are sent to the HP 2250A by the host computer as ASCII strings. The HP 2250A will not compile any of a task until it has received all of the commands plus the terminator, and it will not schedule the task for execution until all of the commands have been compiled. The following FORTRAN IVX statements would send the task in Figure 5 to the HP 2250A, assuming that it is assigned to LU 11. The blanks in the FORMAT statement are ignored by the HP2250A and are only for cosmetic purposes.

```
WRITE (11, 1)
1  FORMAT (" DI(5, 1)"
+ "          DO (3, 1, 4) 1,1,1,1"
+ "          !")
```


The value returned by the DI command above will be put into a place in the HP 2250A's memory called the "main result buffer". This buffer is the default destination for any data obtained by the HP 2250A in response to any input commands contained in a main task. Resident tasks which take data must specify destinations (variables or buffers) for this data with the "IN" command, and they cannot use the main result buffer to store data. The first word of the main result buffer always contains an integer error code for the main task (zero for no error), and the remainder of the buffer can be devoted to returned data.

Task Error Codes

There are two types of error codes for main and resident tasks : compiler errors and run-time errors. If the HP 2250A notices a syntax error, an undefined reference, an illegal parameter, or other problem during the compilation of a main or resident task, then a one-word error code is put into the main result buffer, and the would-be task is removed from memory. The compiler error code can also be obtained from the second word of the eight main task status words, which are available from secondary address 2. Main task status is used because all tasks are considered to be main tasks by the HP 2250A until they are compiled. Run-time errors indicate problems with the actual execution of MCL commands, such as division by zero, memory overflow, and function card malfunction. Secondary addressing is discussed later in this section. For a complete list of the possible error codes, see Appendix A of the HP 2250A User's Manual.

Resident tasks

The compiler error code for a resident task should be read from the main result buffer (through the LU of the HP 2250A without secondary addressing) directly after the resident task has been downloaded. If the code is zero, no compilation errors were found.

Main tasks

If a main task encounters a compiler error, then the error code will be available to the host computer immediately from the main result buffer. If the main task does not encounter a compiler error, then a zero is written into the first word of the main result buffer, and the task is scheduled for execution. Any data routed to the main result buffer by the main task will be stored after this first word.

Should the main task encounter a run-time error, then this error code will be stored into the first word of the main result buffer, and the task will halt. The host computer program should monitor the main task status through secondary

address 2 in order to determine when the task has completed. After the task finishes, the host program should read the main result buffer, which will contain a zero followed by any returned data, or just a run-time error code. Run-time errors cause any returned data in the main result buffer to be lost.

If the main task does not take long to execute, then the host program can read the main result buffer directly after the main task is downloaded to determine if the task was successful. If the main task will take longer to execute than the time-out value for the HP2250A will allow, then the host program should read the main task status through secondary address 2 after downloading the main task, and should keep reading the status until it indicates that the task has completed. The main result buffer can then be obtained.

Examples of using main and resident tasks are included in the &DEMO program at the end of this section.

Scheduling tasks

The HP2250A will not consider a task for execution until it has been scheduled. Main tasks are automatically scheduled as soon as they are received and compiled. Resident tasks can be scheduled by either the START or the GOSUB MCL/50 commands, or by a function card interrupt.

Only one task can execute on the HP2250A at any given time, and it will monopolize the HP2250A's attention until it either reaches a PAUSE command or the end of the task. The PAUSE command simply suspends execution of the current task and allows the next scheduled task to execute. If there are no other tasks waiting which have the same or higher priority as this task, then the PAUSE has no effect. When the task which PAUSEd is scheduled again, it will resume execution at the the command following the PAUSE. The PTIMER command performs the same operation as the PAUSE, except that a time interval is specified with the PTIMER such that the task will continue to pause until this time interval has elapsed. The PAUSE and PTIMER commands are most often used to implement a form of "time-sharing" between several tasks which must execute periodically but don't require large amounts of time.

The HP2250A keeps track of which task should execute by maintaining a "schedule list". Whenever a task is scheduled, its task number and priority are entered into this list. Whenever the currently-executing task ends or encounters a PAUSE command, the schedule list is consulted to determine which of the waiting tasks should next be executed. This decision is based upon task priorities, and in the case of having more than one task with the same priority, the equal-

HP 2250A/HP 1000

priority tasks are executed in a round-robin fashion. For a more in-depth discussion of the scheduling process, refer to the HP2250A User's Manual.

Returning data

The HP2250A returns data to the host computer in binary format. This is quite convenient for HP1000 computers, since they represent integer data internally in this same format. This saves the computer the time and trouble of converting data returned from the HP2250A.

To find out what the result of the DI operation was for the task in Figure 5, the following FORTRAN IVX statements could be used:

```
INTEGER ERROR, DATUM
READ ( 11 ) ERROR, DATUM
```

The above READ statement in FORTRAN IVX defaults to binary format, because no FORMAT statement is specified. This is appropriate for the INTEGER data to be returned by the HP2250A. The main task error code for the task in Figure 5 will be read into the variable ERROR, and the digital value will go into DATUM.

How to stop tasks

There are several different ways to halt executing tasks, and they differ in flexibility, severity, and in the conditions which require their use.

If tasks are to be stopped as a normal routine, and not as an error condition, then the MCL/50 STOP command, included within a task, is appropriate. The MCL/50 RESET command can also be used, but it returns the HP2250A to a power-on state, which may be overkill in most cases.

If "Task 1" is, for example, executing an infinite loop and must be stopped by the operator, then sending another task, "Task 2", to stop it may not be effective. This is due to the scheduling rules of the HP2250A, which require Task 1 (which is the one we want to stop) to either complete normally (which is what we don't want to wait for) or to PAUSE before Task 2 (which contains the STOP command) can execute. Remember, only one task can execute at any given time on the HP2250A. If Task 1 doesn't complete normally or PAUSE, then it cannot be stopped by another task. To stop Task 1, the host computer can send a "Device Clear" to the HP2250A. This is an HP-IB signal which will immediately halt an executing task and also remove any tasks from the schedule list.

The HP-IB Device Clear does not remove any tasks from memory or affect any variables or buffers, and so it is ideal for a "soft reset" during initial experimentation with the HP2250A. The FORTRAN program &HALTR shown below is a utility which performs a Device Clear.

An alternate (but inelegant) approach to stopping a task is to push the little white button marked "RESET" on the HP2250A's CPU card. This always works.

To implement a system with multiple tasks on the HP2250A, it is recommended that the PAUSE or PTIMER commands be used liberally in all tasks which must cooperate with one another.

```
FTN4X,L
PROGRAM HALTR ( 3, 88 )
C
C Program to stop all tasks on the HP2250 by executing a DEVICE CLEAR.
C It also removes pending SRQs from the HP2250.
C
C INTEGER HP2250, LOG
C DATA LOG/1/, HP2250/11/
C
C CALL CLEAR ( HP2250, 1 )
C
C WRITE ( LOG, 1 )
1 FORMAT ( 2/,"HALTR...",T15,"All quiet on the western front.", 2/ )
C
END
```


Configuring the HP 2250A Memory

The DIMENSION command allows the user to specify the number of one-word integer variables and the number and size of integer buffers that will be defined for the system. Every variable and every buffer in memory is accessible by every task in the HP2250A. This feature really simplifies the construction of small, cooperating tasks to handle measurement and control problems.

Because all of memory can be accessed by every task, the user will need to establish some safeguards to prevent tasks from interfering with one another's operations, especially if the programming is to be done by more than one person. One solution to the cooperation problem is shown in Figure 6. It is a "Memory Table", and it contains spaces for variable numbers, buffer numbers and sizes, task numbers, SRQ numbers, and comments for a simple measurement and control system.

Variables are much easier to keep track of if they are grouped into "families" that are functionally related. There is another benefit to grouping variables by function: variables can be accessed through secondaries only in groups in sequential order. For example, two I/O requests would be necessary to write to variables 2, 3, 5, and 6, but only one request is necessary to write to variables 2, 3, 4, and 5.

The HP2250A stores data from most MCL/50 operations in a one-word INTEGER format. The "DI" command, for example, stores an integer "1" or "0" into memory for every channel scanned. However, several MCL/50 operations use the same REAL format as the single-precision real numbers on HP 1000 systems. The "AIR" command (real-format analog input) returns two words of data for every channel scanned. These two words can be read directly into a single real variable on an HP 1000. Downloaded subroutines can also use real variables as well as integers. This is demonstrated in the &DEMO program at the end of this section.

Routing data

There are two MCL/50 commands which route data to and from the function cards. The "IN" command changes the destination of input data from the main result buffer (default) to a buffer or to specified variable(s). If there is more than one data value to be returned, and variables are the desired destination, the data is stored in sequential variables, starting with the specified number.

```
IN ( V2 )
DI ( 5, 1, 3 ) !
```

would input three binary values (1 or 0) from the first three channels of a digital input card in slot number 5 into variables V2, V3, and V4. Likewise, the "OUT" command is used to send values out to the function cards. The following two statements would send the values of variables 2, 3, and 4 to the second, third, and fourth channels of a digital output card in slot 7.

```
OUT ( V2 )
DO ( 7, 2, 3 ) !
```

A pointer is associated with every buffer, and it determines which of the buffer elements will be accessed. It normally points to the last word read into the buffer by the IN command or to the last word written out of the buffer by the OUT command. It is recommended to execute an IN or OUT command just before every related set of function card I/O operations to make sure that the source or destination really is what it's supposed to be.

Buffers are normally filled and accessed serially as a unit. Elements of buffers can be accessed randomly as well as serially by moving the buffer pointer around with the REWIND (which resets the pointer to zero) and SKIP (which moves the pointer backward or forward a number of items) commands. A "buffer index" is used to specify the position of the desired element, relative to the buffer pointer as a reference. The important thing to remember is to always know where the buffer pointer is before attempting to use indexing. For example, B1(3) does not necessarily refer to the third element of buffer 1. It refers to the third element after the buffer pointer, which may or may not be at zero.

Table 3 shows a comparison between using buffers and using sequential variables for the storage of related data.

HP 2250A/HP 1000

Memory Table		Date:	
Title: <u>DEMO system for AN 401-24</u>		<u>2/17/81</u>	
Task No.	Purpose	Disk File namr	
<u>φ(Main)</u>	<u>Initialize the 2250</u>	<u>SETUP</u>	
<u>2</u>	<u>Take thermocouple readings</u>	<u>TEMPS</u>	
Variable No.	Purpose	Used In Task No.	
<u>1</u>	<u>Loop Counter</u>	<u>2</u>	
<u>2</u>	<u>Scratch for analog averaging</u>	<u>2</u>	
<u>3</u>	<u>Value of thermocouple reading in Fahrenheit (takes 2 words)</u>	<u>2</u>	
<u>4</u>		<u>2</u>	
<u>13</u>	<u>Contain the run-time error code</u>	<u>2</u>	
Buffer No.	Size (words)	Purpose	Used In Task No.
<u>1</u>	<u>5</u>	<u>Contain the four thermocouple readings to be averaged</u>	<u>2</u>
<u>2</u>	<u>20(2 words per reading)</u>	<u>contain the averaged readings for release to the host computer</u>	<u>2</u>
<u>3</u>	<u>2</u>	<u>Scratch for "REF" command</u>	
SRQ Numbers	Purpose		
<u>1</u>	<u>Tell host that released data is ready</u>		
<u>2</u>	<u>Tell host that a run-time error has occurred</u>		

Figure 6. A Memory Table for the HP 2250A for the DEMO System described in AN 401-24

Table 3. Differences Between using Buffers and using Sequential Variables for storing Related Data

Buffers	Sequential Variables
Centralized storage which provides easy access for the data as a unit.	Centralized storage which provides easy access to individual elements.
Built-in pointers keep track of storing history data.	Task must supply the variable number explicitly.
Ten words of system overhead are needed for every buffer.	No extra system overhead is required.
Data can be protected from erasure by the RELEASE command.	Data is always vulnerable to being overwritten.
Tasks can be indefinitely suspended if they try to access a released buffer.	Variables can always be overwritten by any task.

Secondary Addressing

Secondary addressing is used to transfer data to and from variables and buffers (even while a task is running), to determine task and system status, and to download user-written subroutines. Secondary addresses are device-dependent extensions to the primary address, and are specified by appending the number of the desired secondary to the LU of the HP 2250A.

Thirteen secondary addresses have been defined for the HP 2250A, and each has a special function. These are shown in Table 4. For example, system status is read from the HP 2250A by reading eight integer words from secondary 1. The following FORTRAN IVX statements would obtain the eight status words (assuming that the HP 2250A is assigned to LU 11):

```
INTEGER STATUS(8)
READ ( 11:1 ) STATUS
```

When using secondary addressing, it is important to read exactly the number of words that the HP 2250A expects you to read. The system status is always returned as an eight-word block; reading only seven words will not complete the I/O transfer. SRQ assertions by a task using the "SRQ" command will be held off until the host computer either finishes the I/O transfer or until it initiates another transfer with the HP 2250A. This delay in sending SRQ messages is discussed more fully under "SRQ interrupts".

Table 4. Secondary Address Functions

Secondary Address	Purpose
1	Read system status (8 words)
2	Read main task status (8 words)
3	Read resident task status (8 words)
4	Read interrupt status (16 words)
5	Write data into buffer
6	Read data from buffer
7	Write data into variable(s)
8	Read data from variable(s)
9	Download subroutines
11	Read released buffer from Port A
12	Read released buffer from Port B
13	Read released buffer from Port C
14	Read released buffer from Port D

For an exact description of the layout of system, task, and interrupt status, see the HP 2250A's User's Manual, Appendix A.

Ports

The RELEASE command makes buffers available to the host computer at a "port". A port is a very special data path to the host computer. When a buffer is released to a port, any MCL task that attempts to alter the data in this buffer by executing a "DIMENSION", "IN", "OUT", or "CBUF" command to this buffer is suspended until a host program reads the buffer through the port. This protects important data from accidental erasure. It is important to note that no other task can execute while a task is suspended in this condition. Thus, trying to alter the data in a RELEASEd buffer in this manner effectively brings the HP 2250A to a grinding halt, and things remain stopped until the host reads the RELEASEd buffer from the port.

The host computer obtains a released buffer by first reading system status from secondary one to find out how many words are waiting at the port. The host then reads that number of words through one of four secondary addresses. This frees up the buffer so that it can again be filled up. There are four ports on the HP 2250A, labeled A, B, C, and D, and they are read by the host computer through secondaries 11, 12, 13, and 14 respectively. Any buffer (except the main result buffer) can be released to any port and more than one buffer can be released to a single port at any given time.

Function Card Interrupts

The Digital Input, Multifunction, Pulse Output, and Counter cards can be configured to generate an interrupt to the HP2250A. By using a combination of the INTERRUPT, SENSE, and SOVERRIDE MCL/50 commands, interrupts can be produced:

1. when a digital point that is connected to a Digital Input or Multifunction card goes high, goes low, or upon either transition,
2. when a Multifunction card completes a counting operation,
3. when the Counter card completes a count of external events or detects an overflow or underflow,
4. and when the Pulse Output card completes a pulse train or detects a limit condition.

These are referred to as "function card interrupts". The user has a choice of either sending an SRQ message when the HP 2250 receives a function card interrupt (this is the default condition), or scheduling a task to handle the interrupt without the host computer's intervention. The ITASK command is used to link a function card interrupt to a resident task such that when the interrupt occurs, the task is scheduled. A task which has been scheduled by an interrupt is subject to the same rules of priority scheduling as any other task, so that it will run when the currently-executing task completes or PAUSEs and its turn arrives.

Function card interrupts are used to attract the attention of the HP 2250A to external events while another task is being executed, whereas "programmed interrupts" (executing the "SRQ" MCL/50 command) are used within a task to attract the attention of the host computer to the HP 2250A.

SRQ Interrupts

As intelligent as the HP2250A is, it is still a controlled device. As such, all communications with it must be initiated by the controller (the host computer), with one exception. When the HP 2250A must inform the controller that it needs attention, it can generate a "service request (SRQ)". By executing the MCL/50 "SRQ(n)" command, the HP 2250A sets the HP-IB SRQ control line true, which causes an interrupt in the host. The "n" in the SRQ command is a number from 1 to 127 which the user can specify in order to give the host more information about the reason for the SRQ interrupt. This "n" will be placed in the HP 2250A's interrupt status, which the host can read through secondary address 4. An example of this process is shown in the &DEMO program at the end of this section.

The host computer then conducts a "serial poll" of all of the devices on the bus to determine which of the devices on the bus needs attention. This is done automatically without the need for user programming. Once the culprit has been identified, a user-written service program can be scheduled on the host to handle the situation. The service program should first read the interrupt status of the HP 2250A through secondary address 4 to find out the cause of the SRQ interrupt. To find out more of the intricacies of SRQ processing, refer to the HP-IB User's Manual. This SRQ-scheduling method of communication between the HP 2250A and the host is a powerful tool for establishing and maintaining cooperation between the two boxes.

But there is a complication with the use of SRQs. The HP 2250A uses direct memory access (DMA) for all input and output operations with its host computer, irrespective of whether or not the host computer also uses DMA, and it achieves a blazing-fast (i.e. up to 500 KBytes/second) transfer rate as a result. However, the SRQ message will not be sent by the HP 2250A while an I/O transfer with the host computer is in progress. This is a hardware restriction with the HP-IB interface for the HP 2250A. Instead, SRQ will be asserted (another way of saying "send an SRQ message") when the I/O transfer completes or when another transfer is started.

The specific conditions that cause a delay in asserting SRQ are listed below. Each of these conditions is an example of a I/O transfer in progress.

1. Before the user has read results after a main task completes (if the task returns data as well as a main task error code)
2. While waiting for an EOI signal on an input to the HP 2250A
3. While waiting for a "!" terminator
4. In the condition that the user doesn't transfer all of the data required when using secondaries
5. During any long data transfer with a slow-moving host computer

The morals of this story are:

1. ALWAYS check system status (via secondary 1) after an I/O transaction with the HP 2250A completes, if there is any doubt of the success of the transfer
2. ALWAYS read the main task error code and all data in the main task buffer after a main task completes on the HP 2250A.

3. ALWAYS transfer the number of data items that the HP 2250A expects you to transfer.

This will ensure that desired SRQ interrupts on the host will occur in the shortest possible time after the SRQ command has been executed on the HP 2250A. This is also an incentive to build safeguards into your programs which can protect against other problems as well as speeding the SRQ assertion. Checking status is always a good idea.

Downloaded Subroutines

For those cases when reducing the involvement and the I/O overhead of the host computer is desired, or when the integer arithmetic capabilities of the HP 2250A are not sufficient, FORTRAN subroutines can be written on the host computer, processed by the LINKR program, and downloaded into the HP2250A via secondary 9. LINKR is an HP-supplied program in the "HP2250A Automation Library" which converts relocatable object code (as produced by the FORTRAN compiler, for example) to binary code which the HP2250A can execute. These subroutines are then "called" within an MCL/50 task just as they would be on the host computer:

```
CALL DGREE ( V1, V2 )
```

The sample program &DEMO in this section shows how to use a downloaded subroutine to convert thermocouple readings from the integer units of tenths of a degree Celsius (as they are returned by the ETEMP command) to single-precision real numbers in units of degrees Fahrenheit.

Sample Program

The sample program shown in this section is intended to demonstrate the following processes:

1. How to send MCL/50 tasks from disk files storage on the HP 1000 to the HP 2250A
2. Main and resident tasks
3. SRQ-scheduling on the host computer
4. Secondary addressing to read status and MCL/50 variables
5. User error processing, both on the host and on the HP 2250A
6. Thermocouple measurement technique
7. Ports
8. Bus triggers

9. Downloaded subroutines
10. Using real numbers with the HP 2250A

This demonstration requires the following items:

1. HP 2250A with:
 - Analog/Digital Converter (25501A), MCU slot 1
 - Low-Level MUX Card (25503A), MCU slot 2
 - Thermocouple Reference Connector (TRC) (25594A), connected to channels 1-16 of the Low-Level MUX.
2. HP 1000 M-,E-,F-, or L-Series with:
 - HP-IB interface
 - "FTN4X" FORTRAN compiler to compile &DEMO and &DGREE
 - "DEMO" FORTRAN application program (shown in Figures 7 and 8).
 - "&DGREE" FORTRAN application subroutine (shown in Figure 9).
 - "SETUP", "TEMPS" MCL/50 tasks in disk files (shown in Figures 10, 11 and 12).
 - "LINKR" program (HP-supplied through the "Automation Library", part no. 25581A)
 - "*DGREE" command file for the LINKR program to convert the %DGREE relocatable code (as produced by running the FTN4X compiler on the &DGREE subroutine) into downloadable file .DGREE for the HP 2250A. (Shown in Figure 13).
3. E-type thermocouple, connected to channel 1 of the TRC

To get started, first make sure that all of the hardware connections and switch settings are correct. Refer to Figure 2 if you need to.

Then, produce the necessary absolute files for the host computer. From FMGR, execute the following commands:

```
:RU,FTN4X,&DEMO,1,-
:RU,LOADR,,%DEMO
:RU,FTN4X,&DGREE,1,-
:RU,LINKR,*DGREE,.DGREE
```

This will compile and load the DEMO program and also convert the &DGREE subroutine source into downloadable form in the file ".DGREE". Make sure that the SETUP and TEMPS MCL/50 source files are on the system, and then run the DEMO program.

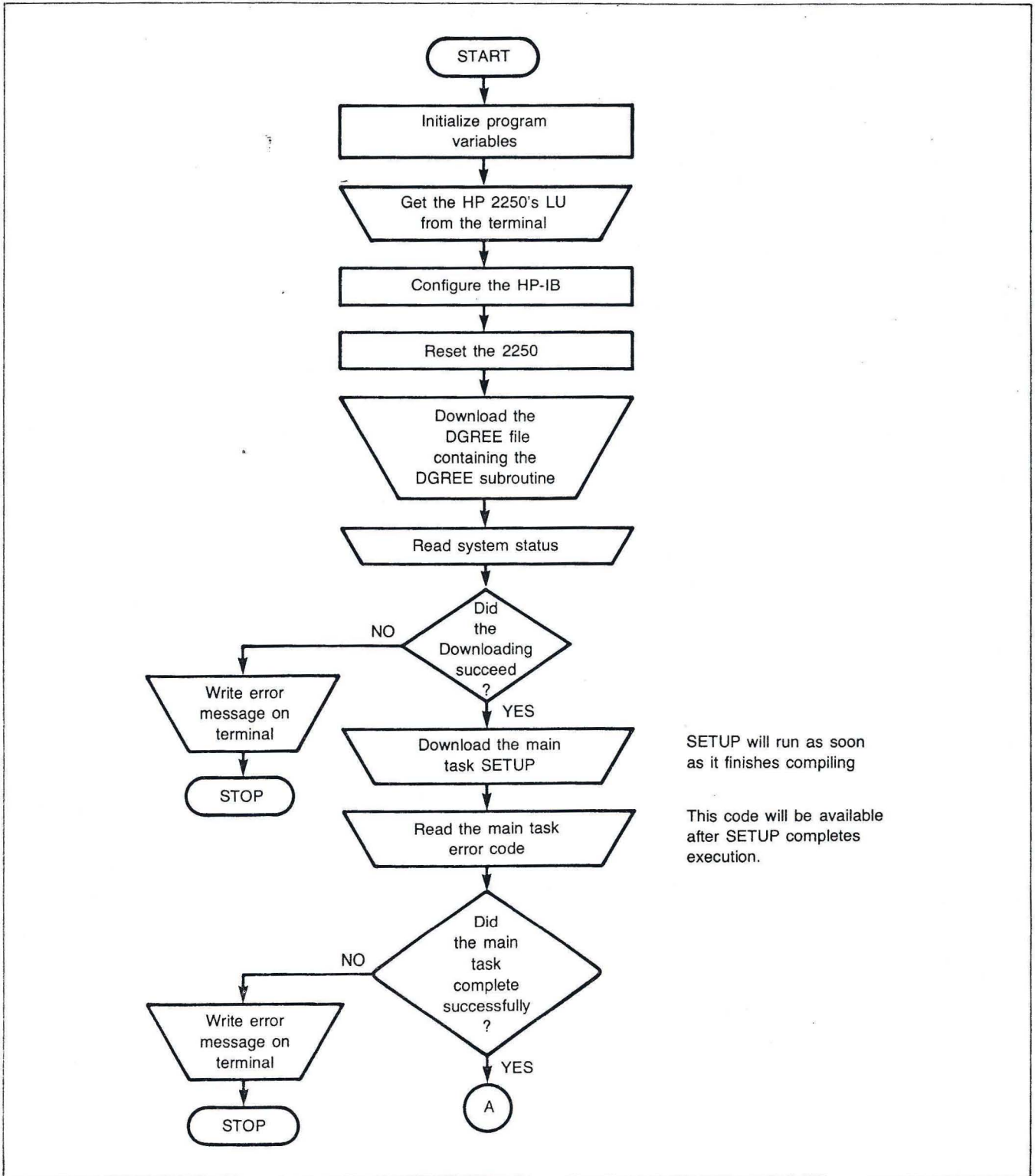
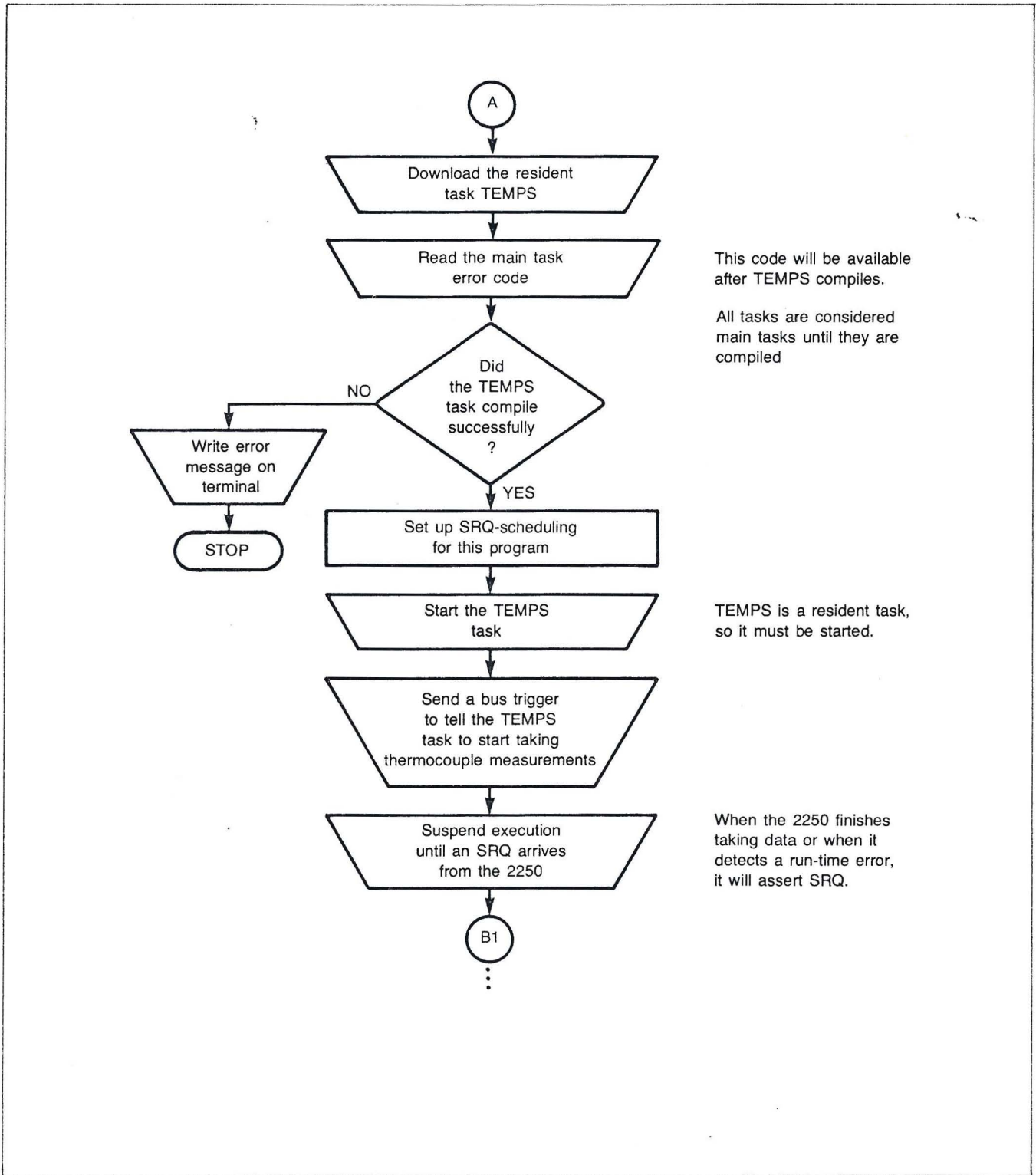


Figure 7. Flowchart for DEMO Program (Sheet 1 of 3)



This code will be available after TEMPS compiles.

All tasks are considered main tasks until they are compiled

TEMPS is a resident task, so it must be started.

When the 2250 finishes taking data or when it detects a run-time error, it will assert SRQ.

Figure 7. Flowchart for DEMO Program (Sheet 2 of 3)

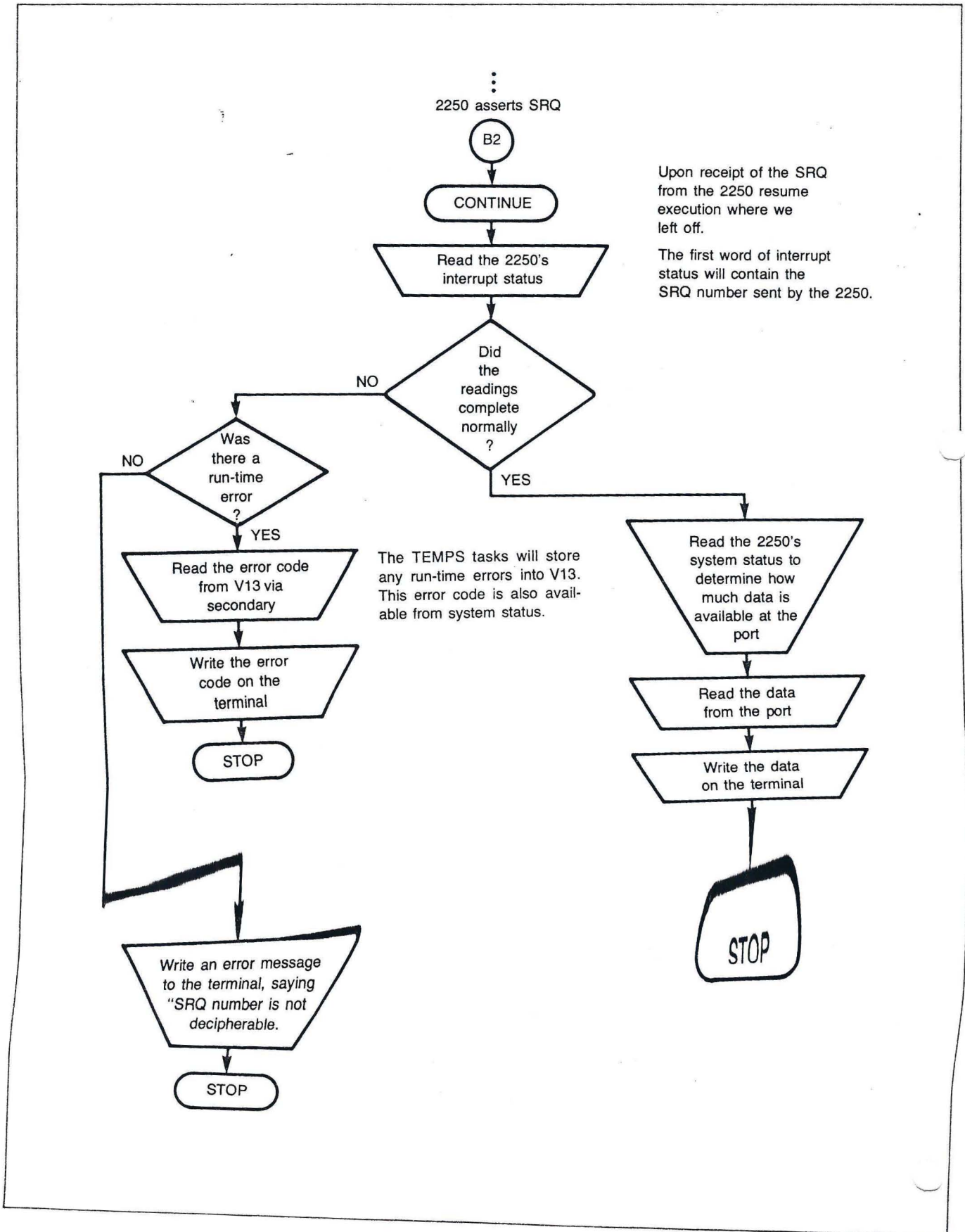


Figure 7. Flowchart for DEMO Program (Sheet 3 of 3)


```

C Set up the HP2250 by sending three disk files to it. The first contains
C a FORTRAN subroutine which has been processed by the LINKR program.
C Subroutines must always be downloaded before tasks. The second contains
C a main task with initialization commands, and the third contains the
C resident task which takes the thermocouple readings.
C
C DD 250 WHICH = -1, 3
C
C DD 201 K = 1, 3
201 FILNAM ( K ) = NAMES ( K, WHICH )
C
C Let the folks at home in on which file is being downloaded.
C
C WRITE ( LOG, 202 ) FILNAM
202 FORMAT ( 2/,"Now downloading the file ",3A2,".",2/)
C
C Open the disk file.
C
C CALL OPEN ( DCB, ERROR, FILNAM, OPTION, CODE, DISK )
C IF ( WHICH .EQ. 1.AND. ERROR .NE. 7 ) THEN
C WRITE ( LOG, 203 ) FILNAM, ERROR
C STOP 0001
C ELSE IF ( WHICH .NE. 1 .AND. ERROR .LT. 0 ) THEN
C WRITE ( LOG, 203 ) FILNAM, ERROR
C STOP 0001
C END IF
203 FORMAT ( "The disk file ",3A2," cannot be opened.",/,
+ "The ERROR code = ",I6,T70," I quit.",5/)
C
C Read the disk file and send it to the 2250.
C
210 CALL READF ( DCB, ERROR, LINE, BEFORE, LENGTH )
C IF ( LENGTH .EQ. -1 ) GO TO 220
C
C IF ( ERROR .NE. 0 ) THEN
C WRITE ( LOG, 212 ) FILNAM, ERROR
C STOP
C END IF
212 FORMAT ( "The disk file ",3A2," encounters read error ",I6,
+/,t70,"I quit.",5/)
C
C Send down the task or subroutine records.
C Trim the trailing blanks from tasks to speed up the downloading.
C
C IF ( WHICH .EQ. 1 ) THEN
C WRITE ( HP2250 : 9 ) ( LINE(K),K=1, LENGTH )
C ELSE
C CALL BLANK ( LINE, LENGTH, AFTER )
C IF ( AFTER .EQ. 0 ) GO TO 210
C WRITE ( LOG, 204 ) ( LINE(K),K=1,AFTER )
C WRITE ( HP2250, 204 ) ( LINE(K),K=1, AFTER )
C END IF
204 FORMAT ( 80A2 )
C
C CALL BSERR ( HP2250, LOG, SRQPRG )
C GO TO 210

```

Figure 8. Sample Program DEMO (Sheet 2 of 6)

HP 2250A/HP 1000

```
C
220 CALL CLOSE ( DCB, ERROR )
C
C If the file was a task, read the task error code next. Downloading
C subroutines doesn't return an error code in the main result buffer, but
C any errors with the downloading are logged in system status word 2.
C
  IF ( WHICH .NE. 1 ) THEN
C
      READ ( HP2250, IOSTAT=ERROR ) MTERR
C
      IF ( MTERR .NE. 0 ) THEN
          WRITE ( LOG, 221 ) MTERR, FILNAM
          STOP
      ELSE
          WRITE ( LOG, 222 ) FILNAM
      END IF
C
  ELSE
      READ ( HP2250:1 ) SSTAT
      IF ( SSTAT (2) .NE. 0 ) THEN
          WRITE ( LOG, 223 ) SSTAT (2), FILNAM
          STOP
      ELSE
          WRITE ( LOG, 224 ) FILNAM
      END IF
C
  END IF
C
221 FORMAT ( 2/,"Compiler ERROR ",I6," has occurred for the task",
+" in ",3A2,".",T60,"I quit.",5/)
222 FORMAT ( 3/,"The task in disk file ",3A2," compiled without"
+" error.",3/)
223 FORMAT ( 2/,"System ERROR ",I6," has occurred for the ",
+"subroutine in ",3A2,".",T60,"I quit.",5/)
224 FORMAT ( 3/,"The subroutine in disk file ",3A2," was downloaded",
+" successfully.",3/)
C
250 CONTINUE
C *****
C Set up this program to be scheduled by the HP-IB driver upon receipt of
C an SRQ message from the HP 2250. Use IBERR again to make sure it works.
C
300 CALL SRQ ( HP2250, 16, SRQPRG )
C *****
C Start the resident task, which should be number 2.
C
  WRITE ( HP2250, 301 )
301 FORMAT ( " START ( 2 ) ! " )
  READ ( HP2250 ) ERROR
  IF ( ERROR .NE. 0 ) THEN
      WRITE ( LOG, 302 ) ERROR
      STOP
  END IF
302 FORMAT ( "Cannot start the resident task. Error = ",I6,5X,
+"I quit." )
```

Figure 8. Sample Program DEMO (Sheet 3 of 6)


```

C
C      CALL BSERR ( HP2250, LOG, SRQPRG )
C *****
C Issue a bus trigger to signal the task running on the HP2250 to start
C taking temperature readings.
C
C      CALL TRIGR ( HP2250 )
C *****
C Go to sleep here. This program will be re-scheduled when the HP-IB driver
C receives an SRQ from the HP2250, and execution will begin at the next
C statement after the EXEC ( 6, 0, 1 ).
C
C      WRITE ( LOG, 303 )
303 FORMAT ( 3/,"The temperature measurements are now being taken.",
+/, "Wait three seconds, then press RETURN.",3/)
C
C      CALL EXEC ( 6, 0, 1 )
C *****
C An SRQ has been received ! Oh, boy ! The first thing to do is read
C the interrupt status of the HP2250 to see what's happening. The status
C is obtained by reading 16 words from secondary address 4.
C
400 READ ( HP2250 : 4 ) ( ISTAT(K), K = 1, 16 )
CALL BSERR ( HP2250, LOG, SRQPRG )
C
k Check the first interrupt status word. If it = SRQ1, then the temperature
C readings completed normally and data should be available at port A on
C the HP2250.
C
C      IF ( ISTAT(1) .EQ. SRQ1 ) THEN
C
C      Read the system status to find out how many words are available from
C Port A. System status is obtained by reading eight words from
C secondary 1. The fifth word read contains the # of words at Port A.
C
C      READ ( HP2250 : 1 ) SSTAT
C      CALL BSERR ( HP2250, LOG, SRQPRG )
C
C      Read the data from Port A ( accessed by reading from secondary 11 ).
C
C      IF ( SSTAT(5) .LT. 1 ) THEN
C          WRITE ( LOG, 401 )
C          STOP
C      END IF
401 FORMAT ( "No data at Port A. Something's wrong. I quit.",5/)
C
C      READ ( HP2250 : 11, IOSTAT=ERROR ) (DATA(K),K = 1,SSTAT(5))
C      CALL BSERR ( HP2250, LOG, SRQPRG )
C
C      Display the results on the log terminal and quit for good. Since the
C 2250 returned the data in "two-word per temperature reading" real
C format, only write out 1/2 the number of "integer words" which is
C indicated in the fifth word of system status.
C

```

Figure 8. Sample Program DEMO (Sheet 4 of 6)

HP 2250A/HP 1000

```

        WORDS = SSTAT(5) / 2
        WRITE ( LOG, 402 ) ( DATA(K), K=1, WORDS )
        WRITE ( LOG, 403 )
        STOP
402  FORMAT ( 2/,T23,"... Thermocouple Temperatures ...",/,
        +T29,"( degrees Fahrenheit )",2/, 5( 5X,5(F6.2,8X),5X,/ ) )
403  FORMAT ( 2/,T34,"End of Demo.",5/ )
C
C   The interrupt status word wasn't = SRQ1, so something went wrong.
C   If the status word = SRQ2, the task caught the error and completed under
C   its own power. Read the error code via secondary 8 from the variable
C   V13 on the HP2250.
C
        ELSE IF ( ISTAT(1) .EQ. SRQ2 ) THEN
C
        WRITE ( HP2250 : 8 ) ( SECOND(K),K=1,2)
        READ ( HP2250 : 8 ) MTERR
        CALL BSERR ( HP2250, LOG, SRQPRG )
C
        WRITE ( LOG, 404 ) CODE, FILNAM
404  FORMAT ( "The HP2250 has encountered run-time error ",I6,/,
        +"The task in file ",3A2," has abandoned ship.",T55," I quit.",5/ )
C
C   If we got here, the interrupt status word is not recognizable.
C   Display it and quit.
C
        ELSE
C
        WRITE ( LOG, 405 ) ISTAT(1)
405  FORMAT ( "Interrupt status =", I6,/, "The HP2250 sent an "
        +"unrecognizable SRQ message.",T55," I quit.",5/ )
C
        END IF
C *****
C That's all, folks.
C
        END
C *****
C This subroutine performs the HP-IB user error-processing.
C
        SUBROUTINE BSERR ( HP2250, LOG, SRQPRG )
C
        INTEGER HP2250, LOG, SRQPRG(4), TIMEOUT, NOBODY
        DATA TIMEOUT/1/, NOBODY/4/
C
        ERROR = IBERR ( HP2250 )
        IF ( ERROR .NE. 0 ) THEN
            IF ( ERROR .EQ. TIMEOUT ) THEN
                WRITE ( LOG, 101 ) HP2250
            ELSE IF ( ERROR .EQ. NOBODY ) THEN
                WRITE ( LOG, 102 ) (SRQPRG(K), K=2,4)
            ELSE
                WRITE ( LOG, 103 ) ERROR, HP2250
            END IF
        END IF
        STOP
    END IF

```

Figure 8. Sample Program DEMO (Sheet 5 of 6)


```

C
101 FORMAT ( "A time-out error has occurred. The HP2250 is not "
+"answering.", /,"Check to see that the HP2250 "
+"is indeed turned on and connected to LU ", I6,". I quit.",5/ )
102 FORMAT ( "The SRQ-handling program ",3A2," cannot be scheduled.",
+T70,"I quit.",5/ )
103 FORMAT ( "HP-IB error ",I6,2X," has occurred for LU ",I6,
+T70," I quit.",5/ )
C
RETURN
END
C *****
C Subroutine to remove trailing blanks from the MCL lines stored in the
C disk files. Downloading speed is much improved with this technique.
C
SUBROUTINE BLANK ( LINE, BEFORE, AFTER )
C
INTEGER LINE(80), LENGTH, BEFORE, AFTER
AFTER = BEFORE
C
100 IF ( LINE (AFTER) .NE. 2H ) RETURN
C
IF ( AFTER .EQ. 1 ) THEN
AFTER = 0
RETURN
END IF
C
AFTER = AFTER - 1
GO TO 100
END

```

Figure 8. Sample Program DEMO (Sheet 6 of 6)

```
FTN4X,L
C
C   SUBROUTINE   DGREE ( CELS, FAHR )
C
C   Subroutine to convert thermocouple readings in .1 Degrees C to Fahrenheit.
C
C   INTEGER CELS
C   REAL FAHR
C
C   FAHR = ( FLOAT(CELS)/10.0 ) * 1.8 + 32.0
C
C   RETURN
C   END
```

Figure 9. Downloaded Subroutine &DGREE

```
NTASKS(10)
DIMENSION ( 20, 3, 5, 20, 2 )
!
```

Figure 10. MCL/50 Task SETUP

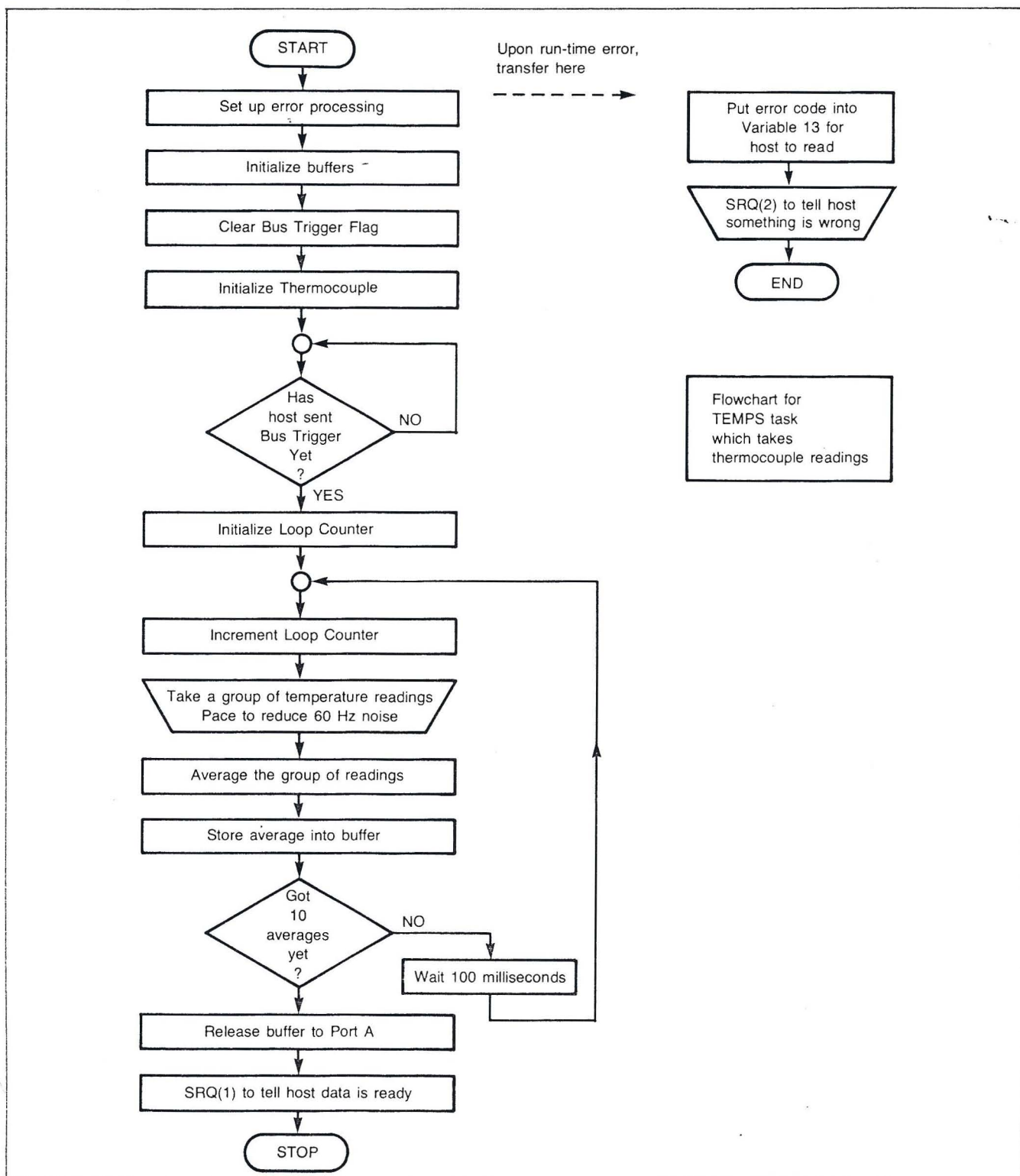


Figure 11. Flowchart for TEMPS Task

```
TASK(2)
    V13 = 0
    ONERROR ( 13 )
    REWIND ( B2 )          REWIND ( B3 )
    IF BT = 1 ENDIF
    IN ( B3 )
    RANGE ( 2, 1 ) 50    RANGE ( 2, 16 ) 1000    REF ( 2, 16 )
LABEL(1)
    IF BT <> 1 THEN
        PAUSE
        GOTO (1)
    ENDIF
    V1 = 0
LABEL(2)
    V1 = V1 + 1
    REWIND ( B1 )
    PACE ( 0, 4, 167 )
    REF ( 2, 16 )
    CLB ( 2 )
    REPEAT (4)
        IN (B1)
        WPACE
        ETEMP (2,1)
    NEXT
    V2 = 0
    AAV ( B1, 4, V2 )
    CALL DGREE ( V2, V3 )
    IN ( B2 )
    OUT ( V3 )
    ECHO (2)
    IF V1 >= 10 THEN    GOTO ( 3 )    ENDIF
    CTIMER
    PTIMER ( 0, 0, 100 )
    GOTO (2)
```

Figure 12. MCL/50 Code for TEMPS Task (Sheet 1 of 2)


```

LABEL(3)

      RELEASE ( B2, A )
      SRQ(1)
      STOP

LABEL(13)
      V13 = ERROR
      SRQ(2)

      !
    
```

Figure 12. MCL/50 Code for TEMPS Task (Sheet 2 of 2)

```

REL ,%DGREE
LIB,$QLIB
LIB,$MLIB1
LIB,$MLIB2
LIB,$MLIB3
MCL,DGREE
EN
    
```

Figure 13. *DGREE Command file for use with the LINKR Program

System Performance

The amount of time necessary to transfer data from the memory of the HP2250A to the memory of the host computer is shown in Figure 14, for both the F-series and the L-Series HP 1000 computers. The performance measurements were made on quiescent (nothing else happening) systems.

READ and WRITE statements in FORTRAN incur formatting overhead each time they execute. This overhead for I/O transfers can be avoided with the use of EXEC calls. EXEC is included in all RTE operating systems, and performs a variety of control functions. The form of an EXEC call is shown below.

```
CALL EXEC ( I, LU, BUFFER, LENGTH )
```

where:

- I = Which operation to perform (1 = read, 2 = write)
- LU = LU of the device to be read
- BUFFER = Variable or array in the user program for the data to be transferred
- LENGTH = How many words of data to transfer

The HP 2250A will return integer data to an HP 1000 in the HP 1000's internal format, so no extra formatting of the data returned by an EXEC call is required. If I/O transfer rates must be maximized, then the use of EXEC calls in user application programs is appropriate.

Execution times for specific MCL/50 commands can be found in the HP 2250A Performance Guide. However, there is a simple method of determining how long any given task will take to execute. Figure 15 shows how to determine the time necessary to read 100 digital channels.

HP 2250A/HP 1000

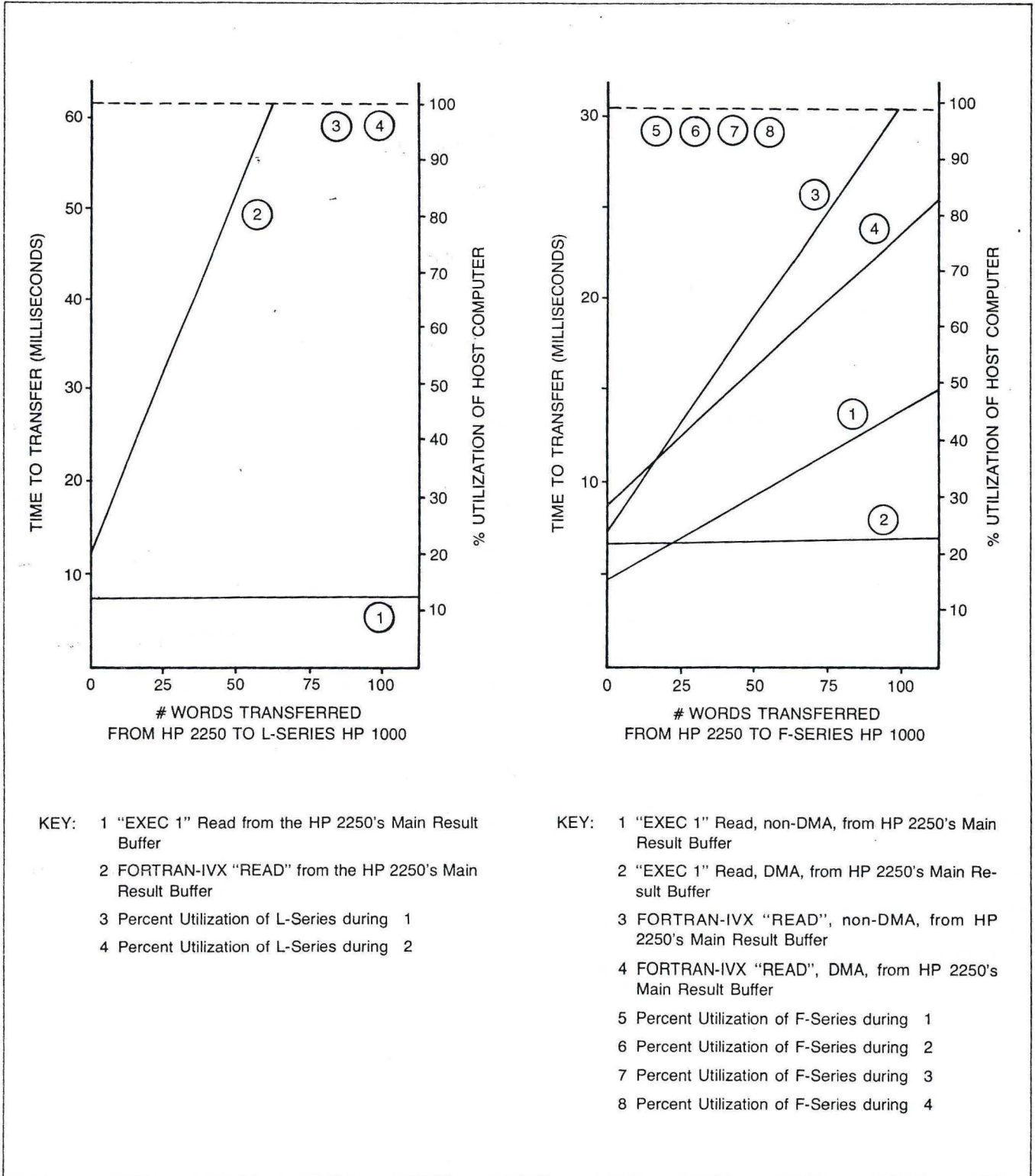


Figure 14. Transfer Rate for I/O between the HP 2250A and the HP 1000 F- and L-Series Computers

MCL/50 Task	Comments
DIMENSION (10, 0)	Set up for 10 variables
CTIMER	Reset the HP 2250 timer
BLOCK	Take all the readings on one channel instead of adjacent channels (saves needing 100 physical channels)
DI (5, 1, 100)	Put 100 readings into the main result buffer
IN (V1) RTIMER !	V1 — hours, V2 — seconds, V3 — millisecc, V4 — microsec that have elapsed since the CTIMER command

Figure 15. How to perform your own MCL/50 Execution Speed Benchmark

To run the above experiment, first determine the overhead time that is involved by executing the above task without the BLOCK and DI(5, 1, 100) commands. Then, include them and run the task again, and subtract the overhead time to get the final result.

This experiment was done for the following commands:

1. DI — Digital point input
2. FI — Digital field input
3. DO — Digital point output
4. FO — Digital field output
5. AI — Analog input

The results of these measurements for successive numbers of channels are shown in Figure 16.

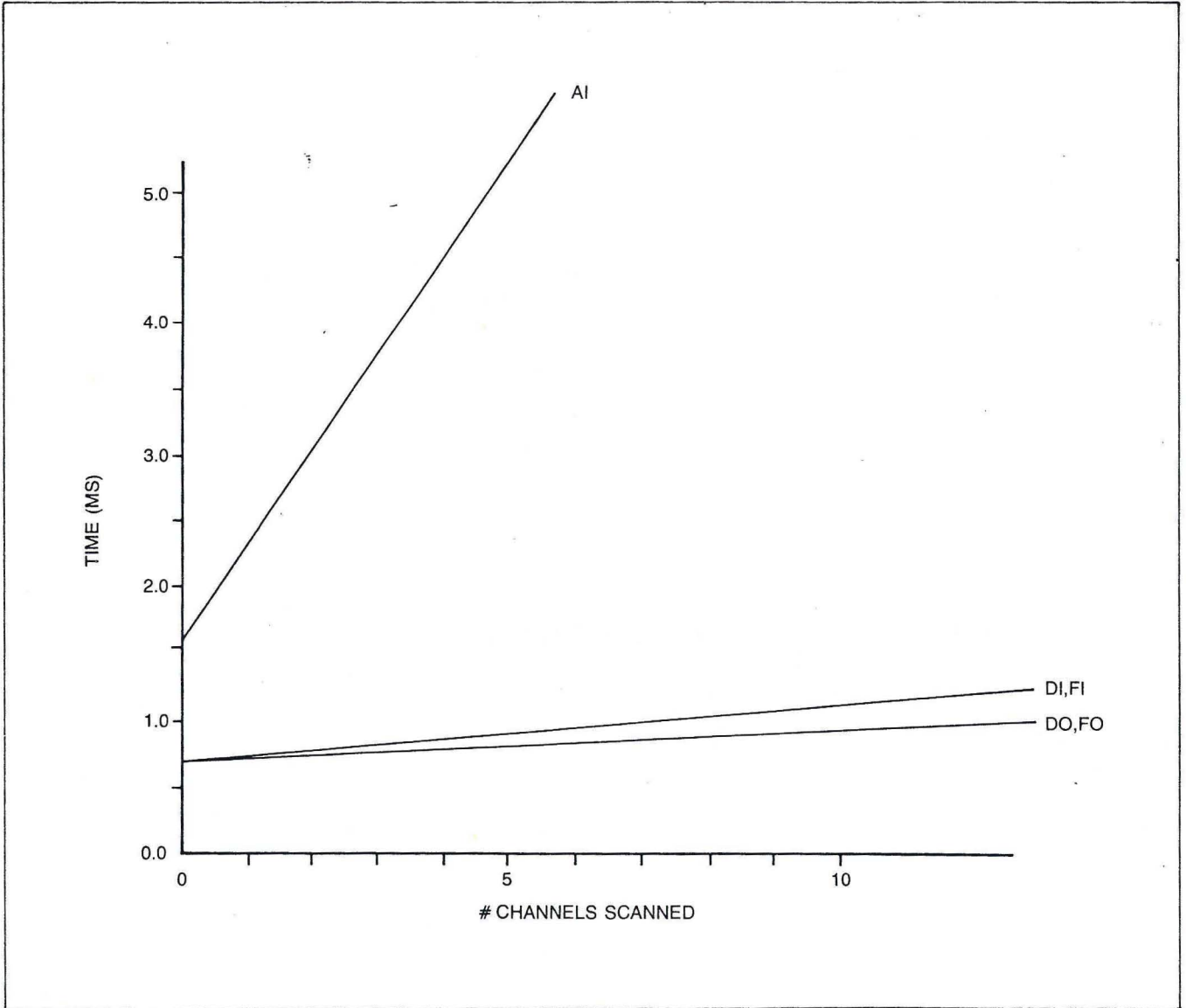


Figure 16. MCL/50 Execution Speed Measurements