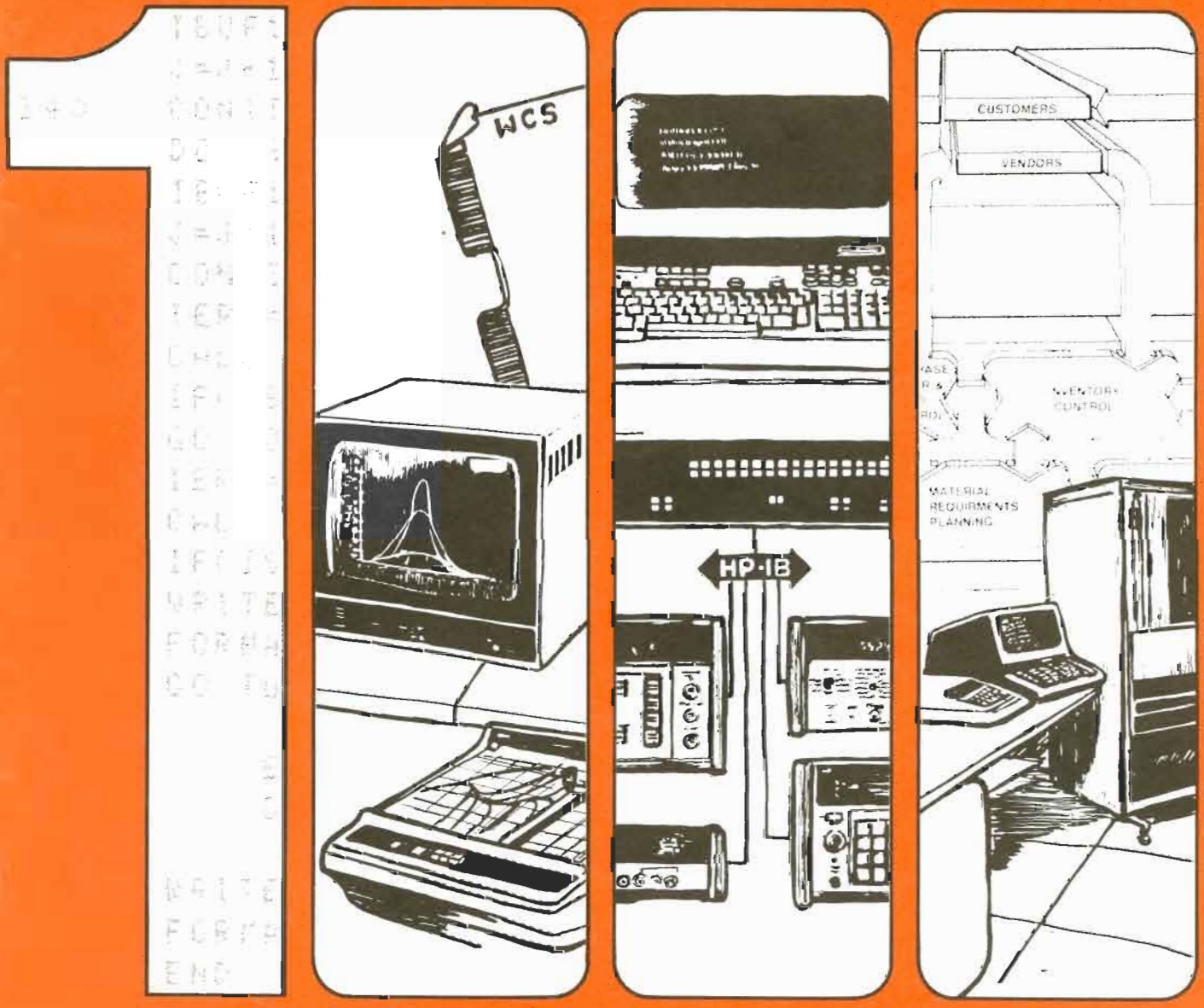
HEWLETT hp PACKARD

Computer Systems

COMMUNICATOR



issue no. 13

HP Computer Museum
www.hpmuseum.net

For research and education purposes only.

EDITOR'S NOTE

Notice anything different about the cover? Things have changed slightly — the COMMUNICATOR has been divided into 3 separate publications for the HP 1000/2000/3000 — and for a good reason. . . . . YOU!

Now, each issue of the COMMUNICATOR 1000 will deal specifically with HP 1000 related products; hardware and software.

The main objective of the publication remains the same however: TO HELP YOU GET THE MOST FROM YOUR HEWLETT-PACKARD COMPUTER SYSTEMS.

In this issue of the COMMUNICATOR 1000, you will see the inclusion of a HARDWARE section, a complete and comprehensive range of SOFTWARE sections, and continual emphasis put on Software and Manual Updates in our BULLETIN Section. For these reasons, and more, we are as excited about the restructuring as we hope you will be.

The COMMUNICATOR 1000 is a quality publication that meets the informational needs of HP Computer Systems users. If you have new ideas, or can suggest changes in the content or format of the COMMUNICATOR 1000, please contact the editor.

Address your correspondence to:

EDITOR

COMPUTER SYSTEMS/COMMUNICATOR 1000
HP Data Systems Division
11000 Wolfe Road
Cupertino, Ca. 95014

CONTENTS

COMPUTATION

- Writing Microprograms: The RTE Micro Debug Editor 1

INSTRUMENTATION

- HP-IB Trekie Article #3 3
- Understanding Your HP-IB Driver (DVR37) 4

OPERATIONS MANAGEMENT

- IMAGE/1000 Data Base Backup on Disc? Yes! 11

OPERATING SYSTEMS

- Know Your RTE — Part 7 12
- High-Speed Interrupt Processing With RTE-II 16
- RTE Performance 20

THE BIT BUCKET (Where all other software information usually goes)

- Writing Subroutines to Use the Fail Error Option in Basic 23
- Keep Pressing Those Soft Keys!! 24
- Software Samantha 25

HARDWARE

- New High Performance Memory Means Increased System Throughput 27

BULLETINS

- "Friendly" Documentation for RTE-M 27
- New Contributed Library Catalog Ready 28
- New Releases For Data Systems LOCUS 28
- Software Updates 32
- Documentation 39
- Training Schedule 42

COMPUTER SERVICE DIVISION INFORMATION

WRITING MICROPROGRAMS: THE RTE MICRO DEBUG EDITOR

Bill Elmore/DSD

- The need for a debug editor
- Creating a microprogram
- Register and parameter values
- Setting breakpoints
  Subroutine and stand alone operation

After determining that some portion of your application should be microprogrammed, writing the microprogram using the RTE Interactive Editor, and then, microassembling, what then? One of the most difficult tasks in creating a workable microprogram is the debugging of the code after it has been written. This is because, unlike most other types of programming, the user doesn't have access to all the registers at a software level with which the microcode can work. In fact, he cannot even halt the computer to check interim values at various stages of the program! Thus one would almost have to resort to the "crash and try again" method of debugging if it were impossible to get any "hooks" into the microcode.

The RTE Micro Debug Editor (MDE), part of the HP 92061A RTE Microprogramming Package, allows the user to execute his microprograms in Writable Control Store (WCS) either in a stand-alone mode or as a subroutine; to set up register and parameter values; and to set breakpoints in the microcode. These are the "hooks" that the programmer needs for debugging, as discussed above. An in-depth discussion of MDE's capabilities is presented in the 21MX M- and E-Series RTE Microprogramming Manuals, but several key functions will be covered here.

There are two methods of actually getting the microprogram into control memory for debugging. A short program may actually be created in control memory by using the MDE RE (replace) Command, and typing in the desired code. The other method, more applicable for longer microprograms, is to create the source code using the RTE Interactive Editor, microassemble the source, and then use the MDE LD (load) Command to load the object code into WCS from disc. When the second method is used, the full power of the RTE Editor (character exchanges, line insertion, tab functions, etc.) may be used in creating the microprogram.

In order to actually execute the microprogram in MDE, one need only type RU,xxxxxx, where xxxxx is the macroinstruction that maps to the microprogram. (For example, RU,105600B.)

Most microprograms make use of certain registers or parameters for retrieval of data from the calling program to use in their calculations. MDE allows the user to set up the software registers and up to ten parameter values using the debug editor's SE (set) and PR (parameter) commands. This allows the user to execute the microprogram in a stand-alone mode and eliminates the need to write software programs to interface with the microprogram for debugging purposes. The SE (set) Command is used to display and alter any of the registers accessible at the microcode level in addition to the software accessible registers (A,B,X,Y,S,E,O), as shown in examples 1 and 2.

The MDE PR (parameter) Command may be used to assign up to ten parameter values following the macroinstruction. Parameters may be assigned as data or as indirect pointers to other parameters. Example 1 shows the user of the PR command.

The ability to set breakpoints, observe and alter the internal registers of the CPU at that breakpoint, and then resume the microprogram, is one of the most powerful tasks that MDE can perform. When a breakpoint is taken, all internal registers (except M,IR, and CNTR in the E-Series computer) are saved, and control is transferred to MDE. At this point, one may inspect and alter the state of the machine and clear or set new breakpoints, and then continue the execution of the microprogram.

In order to set a breakpoint, one need only load WCS with the MDE breakpoint microcode and then set the locations in control memory at which one wishes to inspect the state of the machine. An example of the use of the breakpoint facility is shown in example 2.

Considerations when setting breakpoints are as follows:

- Executioning a breakpoint will modify the contents of the Instruction Register.
- The M-register is not saved across a breakpoint.
- A breakpoint should not be set before data is passed from the T-register after a READ micro-order.
- A breakpoint set on an I/O micro-order will not allow continuation after the breakpoint.

MDE also comes in a subroutine form, which is useful for debugging when the microcode is called from a higher level program. The main program may then call MDE, at which time the user may set breakpoints in order to inspect and alter the state of the machine at execution time.

The ability to create, execute, and debug microprograms on-line in an RTE system can significantly reduce the time for the user to develop microcode for his particular application.

The Micro Debug Editor is an integral part of the HP 92061A RTE Microprogramming Package and is certainly a valuable development tool for anyone using the microprogramming capabilities of the HP 21MX Computers.

EXAMPLE 1: Creating a Program in MDE and use of PR command

```
*ON,FMGR
:RU,MDEP                    <- RUN THE MICRO DEBUG EDITOR PROGRAM
COMPUTER TYPE:  1=21MX,2=21MX E-SERIES
TYPE(1 OR 2)?2
$LU,13

LU#      RANGE     STATUS
13  034000034777  1
$RE,34000B,34003B           <- REPLACE THIS BLOCK OF CODE
34000   LGS         XOR   S3    X
$$READ,NOP,PASS,L,A         <- NEW MICROINSTRUCTION
34000   READ        PASS  L     A
$$/                         <- NEXT LOCATION
34001           STFL  CMPS  A     CNTR
$$NOP,NOP,AND,S1,TAB
34001                 AND   S1    TAB
$$/
34002           STFL  PASS  S11   S1
$$WRTE,MPCK,PASS,TAB,S1
34002   WRTE  MPCK    PASS  TAB   S1
$$/
34003           SRG1  CMPS        MEU
$$READ,RTN,INC,PNM,P
34003   READ  RTN     INC   PNM   P
$$A
$SN,34000B,34003B           <- DISPLAY MICROPROGRAM
34000   READ          PASS  L     A
34001                 AND   S1    TAB
34002   WRTE  MPCK    PASS  TAB   S1
34003   READ  RTN     INC   PNM   P
$SE,A                       <- SET THE A-REGISTER
A   = 0

A   = 0
A   = 377B
A   = 377
A   = A

$PR
P+01= RETURN
P+02= RETURN
P+03= RETURN
P+04= RETURN
P+05= RETURN
P+06= RETURN
P+07= RETURN
P+08= RETURN
P+09= RETURN
P+10= RETURN

P+01= RETURN
P+01= S2525B                <- SET NEXT MEMORY LOCATION FOLLOWING MACRO
P+01= S252S
P+01= A
$RU,10S600B                 <- EXECUTION
RETURN= P+02
$PR
P+01= 12S
P+02= RETURN
P+03= RETURN
P+04= RETURN
P+05= RETURN
P+06= RETURN
P+07= RETURN
P+08= RETURN
P+09= RETURN
P+10= RETURN

P+01= 12S                   <- IT WORKED!
P+01= A
$EX
 $END MDEP
:EX
 $END FMGR
```

EXAMPLE 2: Creating a microprogram, setting breakpoints, and use of the SE command.

```
*ON,FMGR
:RU,EDITR                   <- CREATE MICROPROGRAM SOURCE FILE
SOURCE FILE?
/
EOF
/T;10,15,20,25,30,40        <- SET TABS FOR MICROINSTRUCTION FORMAT
/ MICMXE,L;;;;;;21MX E-SERIES
/ $CODE='M2.1E,REPLACE;;;OBJECT TO DISC

BODY OF MICROPROGRAM        <- USER SELECTED MICROPROGRAM OBJECT FILENAME

/ELC&M2.1E                  <- USER SELECTED MICROPROGRAM SOURCE FILENAME
  LS FILE 2 41
END OF EDIT
:RU,MICRO,2                 <- MICROASSEMBLE MICROPROGRAM
 /MICRO: END
:RU,SRTST,1,S,1             <- CONSOLE LU, NUMBER OF DATA, LIST FLAG (1=LIST)

016440 136075 016336 152742 023501  <- UNSORTED DATA

COMPUTER TYPE:  1=21MX,2=21MX E-SERIES
TYPE(1 OR 2)?2
$LU,13

LU#      RANGE     STATUS
13  034000--034777  1
&LD,'M2.1E                  <- USE FILENAME IN $CODE STATEMENT
$LC,34600B,34417B
$BR,340S2B,34072B           <- LOCATE MDE BREAKPOINT MICROPROGRAM, AND PROVIDE AN UNUSED ENTRY POINT FOR MDE USE, BEFORE SETTING BREAKPOINTS
BREAK 1  34052
BREAK 2  34072
BREAK 3  0
$EX                         <- SET BREAKPOINT IN SWAP MICROINSTRUCTIONS, AND SET BREAKPOINT AT END OF ONE COMPLETE PASS

 START OF SORT
BREAK 340S2                 <- BREAKPOINT IN SWAP MICROINSTRUCTIONS
$SE,L,S6
L   = 16440     S6  = 16336 <- ELEMENTS BEING SWAPPED

L   = 16440
L   = A
$RU
BREAK 34072                 <- AFTER BREAKING AT END OF PASS, REMOVE END OF PASS BREAKPOINT
$CL,34072B
BREAK 1  34052
BREAK 2  0
BREAK 3  0
$RU

BREAK 340S2                 <- BREAKPOINT IN SWAP MICROINSTRUCTIONS
$SE,L,S6
L   = 16336     S6  = 136075<- ELEMENTS BEING SWAPPED

L   = 16336
L   = A
$RU
BREAK 340S2                 <- BREAKPOINT IN SWAP MICROINSTRUCTIONS
$SE,L,S6
L   = 16440     S6  = 152742<- ELEMENTS BEING SWAPPED

L   = 16440
L   = A
$RU
BREAK 34052                 <- BREAKPOINT IN SWAP MICROINSTRUCTIONS
$SE,L,S6
L   = 16336     S6  = 152742<- ELEMENTS BEING SWAPPED

L   = 16336
L   = A
$RU

 END OF SORT                <- NOTE: THESE ARE NEGATIVE NUMBERS
 136075 152742 016336 016440 023501  <- CORRECTLY SORTED DATA
$CL                         <- BE SURE TO REMOVE BREAKPOINTS !
BREAK 1  0
BREAK 2  0
BREAK 3  0
$EX
:EX
 $END FMGR
```

HP-IB TREKIE ARTICLE #3

OPEN COLLECTOR DELAY CONSIDERATIONS

Larry W. Smith/DSD

Now that the HP-IB scheme for interfacing digital equipment is known to the electronic world, the following series of articles titled

"HP-IB TREKIE"

will deal with common and useful things to do with an HP-IB system. The initial thrust and series of articles will deal with performance studies on different areas of HP-IB applications. The intent of these first six articles will be to analyze "WORST-CASE" conditions so the user will have a good idea of how far to push an HP-IB system. It is also intended to give realistic data on system configuration and operation. So, stay tuned to the **COMMUNICATOR** 1000. There's a whole lot of interesting and surprising things to come in the HP-IB area.

To continue our trip down the HP-IB path, let's summarize all the topics covered in the **COMMUNICATOR** 1000:
COMMUNICATOR:

ISSUE 11: HP-CABLE PARAMETERS & DEVICE LOAD CONSIDERATIONS

ISSUE 12: BUS TOPOLOGY & HANDSHAKE ANALYSIS (DON'T MISS THIS ONE)

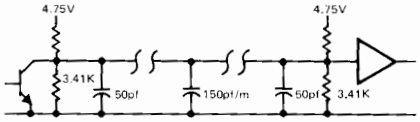
ISSUE 13: OPEN COLLECTOR DELAY CONSIDERATIONS

ISSUE 14: POWERED-OFF DEVICE CONSIDERATIONS

ISSUE 15: DATA SETTLING TIMES & DAV GLITCH

ISSUE 16: SUMMARY

Two of the HP-IB handshake lines use open-collector drivers. It is assumed that these open collector signals will have a very long rise time as compared to the cable propogation delays, so the signal delays may be calculated using lumped circuitry elements rather than transmission line techniques. The equivalent circuit for one driver and one receiver would be as follows:



The driver is represented by the worst-case resistive termination (see last article) and a capacitive load of 60 pf during the signal rise time. Although the standard currently fixes the driver's maximum low-level output voltage at 0.4 volt, it seems reasonable to assume that Schottky devices will be used in critical speed applications. Such drivers would have a low-level output of 0.5 volt maximum and perhaps 0.4 volt typical. The worst-case open collector rise time occurs when this low-level voltage is small, but a minimum voltage is difficult to specify. Therefore, this analysis will assume that 0.4 volt is the worst-case, and a lower voltage yield a slightly slower rise time than calculated below. For example, if 0.3 volt is the correct value, the rise time will be degraded by about 3%. The receivers input impedence is completely represented by a worst-case resistive termination and a 50 pf capacitor. The receiver requires a positive threshold input voltage of 2 volts.

Using the lumped circuit model, the open collector signals have exponential waveforms given by

$$V = VI + (VSS - VI)(1 - E^{-T/RC})$$

where V is the time-varying signal voltage. VI is the initial voltage, 0.4 volt. VSS is the steady-state voltage and is determined by the resistor termination as 2.5 volts (see previous article). R is the A.C. equivalent of all the resistive terminations. For one termination, R is 3.07K in parallel with 3.41K, or 1.62K. Therefore, if N is the total number of devices, R= 1.62/N. C is the total capacitance. If M is the total length of cable in meters, then C= 50N+ 150M.

The exponential equation must be solved for the time required to reach 2 volts, the receiver's threshold:

$$2 = 0.4 + (2.5 - 0.4)(1 - E^{-T/RC})$$

The resulting value for T (in nanoseconds) is

$$\begin{aligned} T &= -RC\ LN\ (1 - 1.6/2.1) \\ &= 1.44RC \\ &= 1.44\ (1.62/N)(50N - 150M) \\ &= 116 + 349\ \frac{M}{N} \end{aligned}$$

For example, a system having ten devices and twenty meters of cable would have a worst-case open collector delay of

$$116 + 349 \times \frac{20}{10} = 814 \text{ NSEC.}$$

Also, a minimum system consisting of two devices and one meter of cable would have 290 NSEC worst-case delay.

UNDERSTANDING YOUR HP-IB DRIVER (DVR37)

Gary Gross/DSD

Lately, we have seen some questions arise concerning the size and power of DVR37, the RTE HP-IB I/O driver. The questions are basically two types:

First we have the new user who asks: "What can DVR37 do in general? How can I use it to solve my instrumentation problem? How well will it handle the devices I will be using on the bus?" If you are this kind of user, we suggest you read through section I of this article and just glance through the latter sections.

Secondly, we have the initiated user who has been working with HP-IB, and wishes to improve and understand HP-IB performance with DVR37. This type of user has been asking: "Does DVR37 work with DMA? How? How effective is it? Is DMA the best way to go for the devices I'm using on the bus?". We suggest that you glance through Section I and read Sections II and III.

The article is broken into three separate sections, the first of which is dedicated to the less initiated user:

I. General capabilities of DVR37

II. DVR37's speed flexibility with data message transfer

III. DVR37's generality with data messages

GENERAL CAPABILITIES OF DVR37

The Hewlett-Packard Interface Bus (HP-IB) is Hewlett-Packard's implementation of IEEE Standard 488-1975 "Digital Interface for Programmable Instrumentation." A standard 2100/21MX HP-IB I/O card (59310B) provides the hardware interface between the CPU and the compatible HP-IB devices. Each 59310B card can support (i.e., identify, control, and service) up to 14 HP-IB devices; with multiple 59310B cards (one CPU I/O slot needed by each) several such clusters can be supported by a single CPU.

The RTE HP-IB driver (DVR37) is the key link between any RTE (II, III, M) system and the 59310B card. It is a general purpose driver used in all RTE/HP-IB configurations irrespective of the particular devices connected to the card. The user can call DVR37 from FORTRAN, BASIC, and ASSEMBLER making it a powerful tool that can be used in a variety of programming environments.

Devices on the bus are designated as talkers (those able to send data, i.e., voltmeters), listeners (those able to receive data, i.e., displays), talker/listeners (those able to do both, i.e., multimeters) and controllers (those able to do both plus control all other devices on the bus). The 59310B card makes the CPU a system controller giving it absolute control of the bus. As such the CPU can actively participate in all data transfer or permit device-device communication without it's intervention.

The actual hardware bus consists of 8 DATA BUS lines (which carry the messages in a bit parallel, byte serial fashion), 3 DATA BYTE TRANSFER CONTROL lines (which provide the asynchronous "handshake" for data transmission), and 5 GENERAL INTERFACE MANAGEMENT lines (which ensure an orderly flow of information on the bus). However, DVR37 offers an automatic addressing mode of operation in which the driver assumes full responsibility for bus protocol. Thus, the user need only be concerned with the actual data to be transferred, its origin, and its destination. This is achieved by allocating a logical unit (LU) number to each device when the system is configured, and thereby permitting HP-IB device I/O to be programmed using standard FORTRAN (READ, WRITE) or BASIC (READ, PRINT) statements and/or RTE EXEC calls.

Control requests (checking status words, clearing devices, etc.) can be handled by all three languages again using the auto-addressing mode.

Although, the HP-IB is generally known as an ASCII bus, binary I/O can be performed by properly configuring the EXEC call control word.

For devices capable of high speed data transfer, DVR37 can be used with the CPU Dual Channel Port Controller (DCPC) for Direct Memory Access (DMA) operation. Transfer rates exceeding those of conventional interrupt driven I/O can be achieved in this manner.

The 59310B card allows the CPU to monitor the service request (SRQ) line (one of the GENERAL INTERFACE MANAGEMENT lines). If the line is pulled, DVR37 performs a serial poll of the connected devices, determines which device initiated the request and schedules the appropriate user written device servicing program.

To better understand the basic operation of RTE/HP-IB systems, the reader is referred to the following:

1. 59310B (HP-IB) Interface Bus Users Guide (HP #59310-90064)
2. DVR37 Reference Manual (HP #59310-90063)
3. HP-IB Article, HP COMMUNICATOR p. 177, Nov. 15, 1975
4. IEEE Standard Digital Interface for Programmable Instrumentation (IEEE-488-1975).

DVR37'S SPEED FLEXIBILITY WITH DATA MESSAGE TRANSFER

There are two routes which may be taken when transporting information via the bus, to or from the computer. Suppose for example, we just set up a particular voltmeter to take a measurement and it sends (to the computer) the following ASCII string:

```
+3.141590E+00 CR LF
```

There are two ways the computer can accept this information. The first method is to interrupt the computer for each word input. That is:

```
I +3 I .1 I 41 I 59 I 0E I +0 I 0CR I LF I
```

where I = an interrupt generated

Each time the central processing unit (CPU) responds to an interrupt it must enter the operating system and incur a slight overhead. As you can see, this overhead will grow rapidly.

The second method we can use to obtain the measurement generates only two interrupts:

```
I 3.14.590E+00CRLF I
```

where I = an interrupt generated

The buffer may be any length, and notice that the overhead does not accumulate with the length of the transfer.

The first transfer method was via the 'interrupt method', the second via 'DMA'. Both have advantages and disadvantages. The interrupt transfer method produces more accumulating system overhead owing to the fact that it must enter the opeating system for each word transferred. However, there are DMA restrictions also. DMA operates on a CPU cycle stealing basis. Simply, this means that DMA is capable of slowing the CPU to a snails pace until an I/O operation is complete. The faster the device, the slower the CPU operates. Also note that there are only two DMA channels in the system.

When selecting the DMA option for speed, one must consider other devices in the system which use DMA also. For example, suppose we have several devices in the system that use DMA (like a DISC and magnetic tape unit). If both DMA channels are currently being utilized by some device, the HP-IB channel will have to wait until a channel becomes available.

We must also consider other devices in the system which use only DMA. If we use DMA to transfer information to or from a very slow device we may tie up a DMA channel for a long period of time. This is obviously not the optimum solution to the problem. The answer here would be to use the interrupt method.

Let's summarize what we have found:

Use DMA when:

1. There is little chance that both DMA channels will be consumed by other I/O devices.
2. For high speed devices with medium or long buffer.
3. For medium speed devices with short buffers.

Use the interrupt method:

1. When it is likely that both DMA channels will be tied up for significant amounts of time.
2. For slow speed devices with long buffers (i.e., line printers).
3. For medium speed devices with long buffers.
4. For very high speed devices (>200 kHz) with short buffers (<60 bytes).

GENERALITY WITH DATA MESSAGES

Let's examine data messages in auto-addressing. Using this method we assign a number to a particular device and send (or receive) data messages to this number which represents the device. For example: suppose we assign the number '20' to a digital voltmeter connected to the bus. Let's first see how we would write to this device:

```
FORTRAN:      WRITE(20,100)
          100 FORMAT("FOR1T3M0A0D0")

BASIC:    155 PRINT #20;"(FOR1T3M0A0D0)"
```

A READ REQUEST MIGHT LOOK LIKE:

```
FORTRAN:      READ(20,101) A     OR READ(20,*)A
          101 FORMAT(E14.6)

BASIC:     95 B$="(E14.6)"       OR 100 READ #20;V
          100 READ #20;A$
          105 CALL DCODE(A$.V2,B$)
```

Using just the read and write statements as shown above, would assume certain characteristics about the driver and the voltmeter addressed (the assumptions will be explained in detail below). In particular, the characteristics would be as follows:

1. If the voltmeter were to generate a require service message during an input or output data message from the computer, we would finish the data message before attempting to service the voltmeter.

2. All data messages are assumed to be via the interrupt method.

3. If this is an input data message to the computer:

   a. Assume the device will generate an end of record (carriage return linefeed) or end or identify (EOI) as (or with) the last character of input data.

   b. Remove all carriage returns from the buffer.

   c. Remove all line feeds from the buffer.

4. If this is an output, send a carriage return linefeed as the record terminator (also assert the EOI line as the line feed is sent).

For every device on the bus (up to 14 devices) there are three words reserved in memory which describe the particular requirements of each one.

The three word area for each configured device is shown below:

```
                                          THIS IS LU <1>
        ********************************************
BEQT1   *S*R*D*I*J*O*P*0*0*0*0*DEVICE ADDRESS 5 BITS *
        ********************************************
BEQT2   *   ALARM PROG ID SEGMENT ADDR               *
        ********************************************
BEQT3   *    SRQ STATUS BYTE                         *
        ********************************************
                          .
                          .
                          .
```

```
                                          THIS IS LU <14>
BEQT1   ********************************************
        *S*R*D*I*J*O*P*0*0*0*0*DEVICE ADDRESS 5 BITS *
        ********************************************
BEQT2   *   ALARM PROG ID SEGMENT ADDR               *
        ********************************************
BEQT3   *    SRQ STATUS BYTE                         *
        ********************************************
```

In the auto addressing mode, DVR37 handles each device on the bus by referring to these three entries.

Specifically, the first word is known as the configuration word for the device. It contains the following information:

S = 0/1 Disable/enable driver to abort a currently active data or control message in order to service a require service message.

R = 0/1 Disable/enable driver to attempt to restart a data or control message request which was aborted, as above. R is functional only if S= 1.

D = 0/1 Disable/enable driver usage of DMA for data messages.

I = 0/1 - ASCII request (INPUT)
read from the bus until one of four conditions are realized:

1. An EOI is generated
2. The input length is reached
3. The device no longer responds

When condition #3 occurs, examine I to determine whether an EOI was expected. If I= 1, the device is out of service so return the error information.

J = 0/1 - ASCII and binary requests (INPUT)

When an EOI is generated, J is examined to determine whether the last byte which was input is valid. If J=0 the last byte is not moved to the user's area.

O = 0/1 - ASCII request (OUTPUT)

O is not examined for an ASCII output request. A carriage return, linefeed (EOI is asserted with the linefeed) are sent as the record terminator unless the '<' (back arrow) is the last byte output.

- Binary request (OUTPUT)

If O= 1 use the EOI line as a record terminator, otherwise do not assert the EOI line.

P = 0/1 - ASCII request (OUTPUT)

P is not used for ASCII output.

- Binary request

P is examined only if O=1. If P=1 the EOI line will be asserted at the same time the last byte is sent. Otherwise (P=0) the EOI line will be asserted after the last byte and sent with a zero byte.

Suppose that we wish to use DMA for one particular device on the bus but not another. With auto addressing, we have this capability, all that is necessary is to set up the configuration word for DMA on this device.

For example: Suppose the LU of our 59309A digital clock is 26. Then the call to set up the configuration word could be as follows:

```
CALL EXEC(3,2500B+26,37000B)
          ----- --
            ↑    ↑
            ↑    ↑ LU OF DEVICE (NOT BUS!)

            ↑    CONFIGURE DRIVER WORD FOR THIS
                 DEVICE ONLY

                                3   7   0   0   0
                              0 011 111 000 0XX XXX
DO NOT ABORT I/O REQUEST FOR SRQ  ↑
0 BECAUSE OF THE ABOVE RESTRICTION  ↑
ENABLE DMA FOR THIS DEVICE            ↑
REQUIRE EOR FROM THIS DEVICE            ↑
REQUIRE IT WITH THE LAST BYTE             ↑
ISSUE EOI TO DEVICE ON OUTPUT (BINARY)      ↑
ISSUE IT WITH THE LAST BYTE   (BINARY)        ↑
THESE ARE ALWAYS ZERO      ....                 ↑
                           ....                   ↑
                           ....                     ↑
                           ....                       ↑
    DON'T CARE                                          ↑
    DON'T CARE                                            ↑
    DON'T CARE                                              ↑
    DON'T CARE                                                ↑
    DON'T CARE                                                  ↑
```

Another call that may be used would be as follows:

```
CALL HPIB(26,21,37000B)
          -- --
             ↑ CONFIGURE DRIVER WORD FOR THIS DEVICE ONLY

           ↑   LU OF THIS DEVICE (NOT BUS!)
```

Now, this particular device is configured for DMA. Whenever we send or receive data to or from this device, we'll now do it with DMA.

DVR37 must determine when the computer has finished sending a data message to the device and when the device has finished sending a data message to the computer:

THE HP-IB HAS TWO STANDARD TYPES OF END OF RECORD (EOR) SIGNALS:

1. The EOR is sent on the eight input output lines (DIO lines) in the form of a carriage return line feed.

2. A separate line known as EOI (end or identify) is asserted signifying end of record.

Imagine the many different devices connectable to the bus and the differences in end of record (EOR) requirements for each. We have the following possibilities:

1. Known buffer length, but we need not send an EOR.

2. Given buffer length, and must give EOR with the last byte transferred.

3. Given buffer length, and must give EOR after the last byte transferred.

WE CAN APPLY A SIMILAR PRINCIPLE TO INPUT:

1. We know the buffer length, but we will receive no EOR.

2. We know the buffer length, and we receive the EOR with the last byte of data from the device.

3. We know the buffer length, and we receive the EOR after the last byte from the device.

4. No given buffer length, but the EOR comes with the last byte from the device.

5. No given buffer length, but EOR comes after the last byte of data from the device.

6. No given buffer length, and no EOR.

Believe it or not, DVR37 can handle all of these situations, which is a big order. We may alert DVR37 to the type of record terminator (if it deviates from the default) by restructuring the configuration word for the device.

LET'S TAKE AN INPUT TO THE COMPUTER FOR EXAMPLE:

We have two types of requests available; ASCII, or binary. ASCII is used most often. The table below demonstrates all of the possible configurations available for ASCII and binary input requests.

The binary input mode is used when a device needs complete control over the data buffer including carriage return, line feed bytes. In this case the EOI line is used as the record terminator. Examining the binary input table below will show you how to configure for various EOI record terminators.

ASCII INPUT CONFIGURATION SHOWING EXAMPLE STRINGS COMING FROM A DEVICE ON THE BUS TO THE COMPUTER

user's desired input string = 1.2
! = EOI generated from device
shaded = information taken from the bus

EXAMPLE DEVICE OUTPUT	*COMMENTS	*DEVICE CONFIGURATION WORD FOR A SUCCESSFUL COMPLETION
1 . 2 CR LF 2 . 3 CR LF ! !	carriage return linefeed and EOI	I=0, or 1, J = 0 or 1
1 . 2 CR LF 2 . 3 CR LF	carriage return	I=0, or 1, J = 0 or 1
1 . 2 LF 2 . 3 LF ! !	linefeed only linefeed and EOI only	I=0, or 1, J = 0 or 1
1 . 2 LF 2 . 3 LF	linefeed only	I=0, or 1 J = 0 or 1
1 . 2 1 . 2 ! !	EOI only with last byte	I=1, J = 1
1 . 2 X 1 . 2 X ! !	EOI only after last byte	I=1, J = 0
1 . 2 1 . 2	no EOR, input length must equal three char.	I=0, J=0 or 1
1 . 2	Timeout situation	I=0, input length > 3

This last case corresponds to when the device fails to respond after the last byte, and the user specifies in the configuration word that this is a legal end of record condition.

BINARY INPUT CONFIGURATION SHOWING EXAMPLE STRINGS COMING FROM A DEVICE ON THE BUS TO THE COMPUTER

User's desired input string = 1.2
! = EOI generated from device
[shaded] = Information taken from the bus

EXAMPLE DEVICE OUTPUT	*COMMENTS	*DEVICE CONFIGURATION WORD FOR A SUCCESSFUL COMPLETION
1 . 2 1 . 2 ! !	EOI only with last byte	I=1, J = 1
1 . 2 X 1 . 2 X ! !	EOI only after last byte	I=1, J = 0
1 . 2 1 . 2	no EOI, input length must equal three char.	I=0, J=0 or 1
1 . 2	Time out situation	I=0, input length > 3

This last case corresponds to when the device fails to respond after the last byte, and the user specifies in the configuration word that this is a legal end of record condition.

user's desired input string = **1 . 2 CR LF**

EXAMPLE DEVICE OUTPUT	*COMMENTS	*DEVICE CONFIGURATION WORD FOR A SUCCESSFUL COMPLETION
1 . 2 CR LF 1 . 2 CR LF 1 ! !	EOI only after last byte	I=1, J=0

This last example illustrates that the carriage return linefeed is not effective in the binary input mode.

ASCII output adds a carriage return and linefeed to your output buffer (it also asserts the EOI line during the linefeed) unless a '<' is the last byte output. In this special case the carriage return linefeed is suppressed. Binary requests are shown below:

Binary output configuration showing example strings transmitted from the computer to a device on the bus

```
                  ↑>>>>>>>B REPRESENTS A BINARY REQUEST
                  ↑
                  ↑ ↑>>>>>THIS SUBSCRIPT REPRESENTS THE 0 BIT VALUE
                  ↑ ↑
                  ↑ ↑ ↑>>>THIS SUBSCRIPT REPRESENTS THE P BIT VALUE
                  ↑ ↑ ↑
OUTPUT STRING   B(0,0)    B(1,0)     B(1,1)
   ABC            ABC       ABC0       ABC
                                !          !
! = AN EOI GENERATED
```

Now remember that most devices will default to the standard configuration word, in the ASCII mode, however, you have this option available to you if you wish to use it.

COMPUTER SYSTEM ERROR CHECKING

Beside the normal device status byte error checking, which is standard with HP-IB, we also have a somewhat more sophisticated error checking system in the computer. Suppose for example, that we are performing a FORTRAN read from a particular voltmeter (LU #20) on the bus:

```
      READ(20,100)A
  100 FORMAT(E14.6)
```

During the read the voltmeter just stops giving the computer data. We say the device "TIMES OUT". We can specify a certain amount of time which we'll wait for a device to respond. If the device doesn't meet the requirement, the system returns to the next statement in the program anyway,

with information about the device. Let's look at FORTRAN for example. To check for errors after a data message request (like the one shown above) we could enter the following statements:

```
      IERR=IBERR(20)
      IF(IERR.NE.0) WRITE(1,200) IERR
  200 FORMAT(/"HP-IB VOLTMETER ERROR NUMBER:"I6)
```

Some of the errors we can recognize are shown below:

Data message device 'TIME OUT'
Interface clear detected during a data message
Require service message has aborted a data message
Non-existent require service message alarm program

We also have the ability to automatically schedule an independent program as the result of a require service message. Another feature which can be used is direct I/O. This enables us to change controllers and allow devices to talk between themselves independent of the computer. We allow multiple HP-IB ports operating independently of one another. The require service message response conducts a serial poll automatically controlled by the driver. So, you can see that DVR37 is a sophisticated system controller which can be used to simplify possibly complex tasks. Yes, mass transit is here to stay!

SUMMARY

LET'S SUMMARIZE WHAT WE'VE COVERED IN THIS SECTION:

First we demonstrated how to perform simple data messages using auto addressing.

Second we showed that the data messages sent or received defaulted to a certain format defined by the devices configuration word. Namely, no DMA was used, and the record termination followed a standard format.

Third we displayed how to modify the configuration word for each device. Most important, how to specify DMA for the device.

Fourth we showed the difference between ASCII and binary requests and why a device might use them.

Fifth we showed how to set up the configuration word so that we may communicate with various types of devices which may not use the standard configuration.

Sixth we showed you some example error checking by the system, over and above the standard error checking done on the HP-IB.

These are just some of the features of DVR37, we also have the option of using most of the other standard message commands which are familiar to most calculator users:

Data messages
Trigger messages
Clear messages
Remote messages
Local messages
Require service message response
Status byte and bit messages

IMAGE/1000 DATA BASE BACKUP ON DISC? YES!

Gary Gubitz/DSD

The IMAGE/1000 Data Base Utility System routines provide data base backup capability from disc to magnetic tape. Used in conjunction with the Spool Monitor, these utility routines can also provide disc to disc backup of IMAGE/1000 data bases with no change in the existing software.

The article describes the use of the IMAGE utility routines and the Spool Monitor in saving (and restoring) a data base on (from) magnetic tape or disc.

I. DATA BASE UTILITY SYSTEM ROUTINES

IMAGE/1000 data base backup capability is provided by routines DBSTR, DBRST, DBULD and DBLOD of the Data Base Utility System. Briefly, their functions are as follows:

DBSTR - Copies the data base root file and data base entries to magnetic tape, sector by sector.

DBRST - Restores a root file and associated data base entries to disc files from a magnetic tape file created by DBSTR. Unlike DBULD and DBLOD, DBRST only restores the contents of the files; no modification of the data base structure is allowed.

DBULD - Copies data entries from a data base to a magnetic tape file. Unloading the data base with this routine allows the user to reload the data base into a different data base structure.

DBLOD - Loads data entries into a disc file from a magnetic tape file created by the DBULD routine. DBLOD users can restore the data to the same data base structure or create a new data base structure using a new data base schema.

NOTE that the DBSTR and DBRST tend to be much faster than DBULD and DBLOD because no restructuring of the data base is performed.

II. USING DBUS ROUTINES

To execute DBSTR, DBRST, DBULD or DBLOD, the user enters the directive:

:RU,DBXXX,p1,p2 where
XXX is either STR,RST,ULD or LOD,
p1 is the console logical unit (default= 1) and
p2 is the magnetic tape device logical unit (default= 8)

The user is then prompted for the data base name, security code and level access word before the save or restore is performed.

III. DATA BASE BACKUP TO DISC

Used in conjunction with the Spool Monitor, DBSTR, DBRST, DBULD and DBLOD can also save (and restore) data bases on (from) FMP disc files. For disc backup, the following steps should be performed:

1. Be sure that the spool program has been generated in the system and that the EQT, DRT and Interrupt Table entries have been properly assigned for spooling.

2. Create the backup file (type 3 or greater) on the desired disc cartridge. File creation can be done as part of the spool job (Step 3).

3. Create a spool job which a) performs an LU switch, associating the magnetic tape LU with the backup file and b) executes the appropriate utility routine.

4. Run the spool job using the Spool Monitor program JOB.

   ***RU,JOB,filename** where
   filename is the name of the file containing the job to be spooled

IV. EXAMPLE

The following example illustrates the use of the Spool Monitor to perform data base disc backup:

The following job is created as a FMP file called SAVE:

```
:JO
:CR,BKUP::10:3:128
:LU,8,BKUP,BOST
:RU,DBSTR,1,8
:EO
```

The spool job can then be run with the command:

```
*RU,JOB,SAVE
```

(A type 3 (extendable) file called BKUP is created on CR010. The magnetic tape LU(8) is associated with the file BKUP. The utility routine DBSTR is then run, copying the data base to the disc file, rather than the magnetic tape.)

To restore the data base, a job RSTOR is created as a FMP file.

```
:JO
:LU,8,BKUP,BOST
:RU,DBRST,1,8
:EO
```

The job can be run with the command:

```
*RU,JOB,RSTOR
```

(The LU switch is made and DBRST is run, restoring the data base from the backup file on disc, rather than magnetic tape.)

V. REFERENCES

IMAGE/1000 Data Base Management System Reference Manual (92063-90001)
Batch-Spool Monitor Reference Manual (92060-90013)

KNOW YOUR RTE — PART 7

Mr. RTE

This is the 7th of a series of articles in the **COMMUNICATOR** dealing with the inner works of HP's RTE systems. These articles go into some detail on how the system works; therefore, you should have already read and become familiar with the material in the RTE reference manual to your system.

In this issue we are going to cover powerfail-auto restart.

In general, what we want to do at powerfail is to save the current machine state, and when power is restored, to restore the saved machine state.

Before we go into details of the powerfail routine, we are going to expound on some little known facts about the 2100 series computers, which make powerfail recovery hard. We don't want to convey the idea the 2100 series CPU is unique in this respect. Powerfail recovery is hard in any environment regardless of the CPU.

Along the way, we will mention why the system does some of the things it does in the way it does them. First there is the write only memory problem.

WRITE ONLY MEMORY (WOM)

Write only memory is memory that may be written but not read. If this memory is volatile (i.e., it is lost when power fails) it becomes a concern of the power-fail software. "Where does the 2100 series CPU have WOM?" you say. Consider:

```
OTA 5
```

This instruction sets the memory protect (MP) fence register to the contents of the A register. The MP fence register is WOM because there is no way of reading back its contents. The fact that the hardware may interrogate the register from time to time to do MP checks does not alter the fact that, from the point of view of the powerfail routine, the MP fence register is WOM. There are other cases of WOM and, in fact, they are a growing group. Another case that comes to mind is the 12966 interface card which is used with the 2640 series terminals via DVR05. This card has a RAM on it which, to the software, looks like a WOM. The RAM contains information which the card uses to identify 'special' characters which by definition, will cause an interrupt when received from the terminal.

For timing reasons it is not desirable to send these special character flags to the interface each time it is accessed and we cannot read them.

Solutions to these two problems, as is generally the case in the gray world of powerfail recovery, are unique to the problems. In the case of the MP fence register the system always sets the MP fence as follows:

```
STA FENCE
OTA 5
```

where FENCE is a base page location (see your RTE-II/III manual Appendix A). This allows the powerfail routine to restore it on power up. It is important to note that the order is important because a powerfail interrupt may occur between the two instructions.

The solution for DVR05 is to test the flag register (SFS) on the card. The driver always completes with the flag clear and the hardware POPIO which is generated on power up sets all interface flags. Thus, if the flag is set the 'special' character RAM must be reset.

The other sides of the coin is the read only memory (ROM) which is volatile.

READ ONLY VOLATILE MEMORY

The volatile (i.e., it is lost on powerfail) ROM comes in two flavors in the 2100 series CPUs.

Flavor one is passively volatile. This is best illustrated by the following instruction:

```
LIA 5
```

This instruction reads the violation register which should (provided we ignore memory parity errors) contain the address of the instruction causing the MP violation. The problem is that there is no way of restoring it when power is restored.

Flavor two is maliciously volatile. This is illustrated by:

```
LIA 4
```

This instruction fetches the central interrupt register (CIR), in other words, the select code of the last device to interrupt. The system uses this to dispatch interrupts to drivers, time base, memory protect etc. The problem is that not only is the register volatile, it is lost by the powerfail interrupt itself.

The solutions to the first problem depends on the CIR and is coded as follows:

```
      in the power down routine

      EXT $PWR5
      .
      .
      .
      LIA 5
      .
      .
      .
      STA $PWR5
      .
      .
      .
      in the system EXEC module
      .
      .
      .
      ENT $PWR5
      .
      .
      .
$RQST LIB 5       GET VIOLATION ADDR
      LIA 4       DO NOT REARRANGE!!!
      CPA D4      POWER FAIL?
      LDB $PWR5   YES,USE SAVED ADDR
      .
      .
      .
```

We will only have indecision at $RQST if the powerfail occurred after the LIA 4 in RTIOC returned a 5 and before the LIB 5 at $RQST. In this case, the LIA 4 at $RQST+ 1 will return 4 and the power down code insures that $PWR5 is what is needed. Note that if the code were:

```
$RQST LIA 4
      LIB 5
      CPA D4
      LDB $PWR5
      .
      .
      .
```

A powerfail interrupt immediately after the LIA 4 would not be detected and the wrong value would be used. On the other hand, the code ordering used may be interrupted by powerfail at any point.

The solution to the loss of the CIR by the powerfail interrupt is a little more involved and requires a little more detail on what powerfail recovery does. In brief powerfail recovery:

a. restores the system state, except for I/O, as near as is possible.

b. Starts the system clock and sets a time out on the powerfail EQT, said time out to occur in one tick.

c. Returns to the point of power failure.

d. On time out entry, proceeds to restore I/O.

e. After all I/O is restored the routine 'AUTOR' is aborted and then scheduled.

The CIR is used by RTIOC to dispatch interrupts and the consequence of it being 4 (which it will be because of the power up interrupt) are that RTIOC will consider the interrupt illegal, will issue an ILLEGAL INTERRUPT message and then return to the point of interrupt. So what's the "diffemence"? Well, the correctness of this action depends on what the actual interrupt was for. We are going to restart all I/O so this action is all right for I/O interrupts. What we have left is the MP interrupt. We would like to either save the fact that it was an MP interrupt or to re-execute the violating instructions and thus get the registers straightened out again. The problem is that:

a. The P-register stored at $CIC by a MP interrupt is not the violation address. (It, in fact, is different on different CPUs in the 2100 series.)

b. Because the CIR is lost, we must figure out that a MP was being processed independently of the CIR.

When we have figured out that a MP is being processed, we want to save the violation address in $CIC so that the return to interrupt will go to the violation address. We use the following code:

```
.
.
EXT     $PWR5,$CIC
.
.
LIA 5
LIB 5
CPA B
STA $CIC
STA $PWR5
.
```

The feature we are using is that the violation register follows the P register until a violation occurred at which point it is 'frozen' until a STC 5 is executed. This means that if a MP interrupt was not being processed the A and B registers will contain the addresses of the LIA and the LIB instructions respectively, whereas if a MP interrupt is being processed A and B will both equal the violation address and we save this in $CIC so that after the illegal interrupt message, RTIOC will return to the violating instruction.

For those among you who want to run out and try to get this message by powerfailing your system, be advised that a "quiet" system is susceptible to this problem about .35% of the time; i.e., within about 300 powerfails you should see it.

Now we can consider the powerfail routine in detail.

POWER DOWN

When powerfails the A,B,E,O,S and if MX series, X and Y registers are saved (NOTE on MX series CPUs LIA 6 returns – 1 and other 2100 series machines it returns 0). In addition, $CIC and $PWR5 are set as shown above and the state of the interrupt system is saved. The word counts on DCPC channels are saved as, in RTE-III, are the memory MAP registers.

POWER UP

On power up, because it takes a while to get out of the power up routine, a switch is set so that a power failure in the up processor will just halt - without saving (destroying) any registers. Note all power up code must be able to stop at any point and start over from the top.

After the switch is set, the CPU is notified that it may process a power down at any time (STC 4) and the EQT area is search for DVP43's EQT (EQT2 points at IP43). If this EQT is not found or if the power down code was not executed (i.e., powerfailed with the CPU halted), the powerfail routine halts (HLT 4, C). Once this EQT is found, its timeout word (EQT15) is set to – 1 and, if a powerfail time is not in hand, the current system time is saved as the time of powerfail (this is printed out by AUTOR). The 'time in hand' flag will be cleared only when AUTOR completes, so power failure before that time will report the same time.

At this time $SCLK in the RTIME module is called to start the system clock.

This routine, while not in the powerfail package is a powerfail routine and is charged with setting up the time base to interrupt every 10 ms with the first interrupt to occur immediately.

We now have a problem. There is an interrupt pending and we may need to turn on interrupts before exiting to the point of the powerfail interrupt.

Further we must reset a switch so that a subsequent power down interrupt will be handled correctly without allowing a power down interrupt to occur before we exit.

First let's figure out if the interrupt system is to be on or off when we exit. Figure 1 shows the relationship of the interrupt system, MPTFL (a base page word) and the privileged interrupt fence card. This figure shows all the possible states the powerfail may have occurred in and how the powerfail routine restores the system for each state. Note that for privileged systems the system turns the interrupt system on, as shown by the dotted line, while powerfail will leave it off. All I/O is dead anyway so this is of no importance. We must, however, set up the privileged card as shown in case the interrupt system does follow the dotted line when the powerfail routine returns. The interrupt system is to be turned on only if it was on at powerfail (this is checked on the way down by SFS 0) and MPTFL (on the system base page) shows it to be on.

The code used to exit from the powerfail up interrupt is:

```
$POWR NOP
      SFC 4           test for down or up
      JMP UP          -it is going UP-
*
      JMP DOWN,I      -it is going down-
DOWN  DEF DWN
*
      STF 0           turn on interrupt system
SW2   STC 5           (CLF 0 if interrupts to be off)
      JMP PSAVE,I     (JRS in RTE III)
*
DOWNI DEF DOWN
DWAIT DEF WAIT
*
EXIT2 LDA ASAVE
      LDB BSAVE
      JSB DOWNI,I
*
*    DOWN ROUTINE
*
DWN   STA ASAVE
      .
WAIT  CLC 4
      HLT 0
*
UP    LDA DWAIT
      STA DOWN
      .
      JMP EXIT2
```

This shows the complete coding of the down switch at DOWN but not the details of the SW2 switch. The instruction at SW2 is STC 5 if the interrupt system is to be on an exit and CLF 0 if it is to be off. The powerfail routine takes advantage of the fact that the following instructions are not interruptable (JMP,I;JSB,I;STF;CLF;STC). Note that the JSB DOWNI,I resets the switch at DOWN to point to the down code while not using any registers and while holding off a possible powerfail interrupt.

RECOVERING THE I/O SYSTEM

We have now covered all the powerfail recovery except the I/O system. The philosophy behind the powerfail routine was to write no code respecting any given I/O device. If any device does need additional help, the code to help it should be in its driver. It should also be noted that the system cannot be expected to recover a device about which it has no knowledge. Before the clock was restarted, the time out word in the powerfail EQT was set to −1. This causes the system to time out this entry and, since, the S bit (EQT4) was also set, to enter the powerfail routine at entry point CS43. The code at CS43 sets up to force another time out and then scans the EQT tables looking for active, UP, or down devices. The actions taken in each case are as follows:

DEVICE DOWN

CALLS '$UPIO' to up the device. Any pending I/O will be send to the driver which will make a new UP/DOWN test etc.

DEVICE BUSY

If the P bit is set the driver is called at its I.XX entry point with EQT5 showing the device busy and A=the device select code. This is the only I.XX call possible with EQT5 showing busy, thus the driver knows that power up is in progress.

If the P bit is not set, EQT5 is set to show the device down and the DEVICE down action is taken. This will effectively restart the I/O in progress at power failure and is exactly the right thing to do for a disc and serves fairly well for most TTY type devices. Note that the first post-powerfail call that DVR05 sees is going to be one to start a new I/O request (or to restart an old, which is the same thing to the driver), thus its SFS test will be soon enough to be effective.

DEVICE UP

The device is put down and $UPIO called to up it. This takes care of down LU's on a given device. If there are no outstanding requests, no action is taken by $UPIO other than clearing the down bit.

Since $UPIO does not return, the powerfail routine regains control by a new time out after a $UPIO call.

After all EQT's are processed the powerfail routine aborts, by name, the AUTOR program. This is done in case it is still active from a previous power failure. On the next time out entry AUTOR is scheduled.

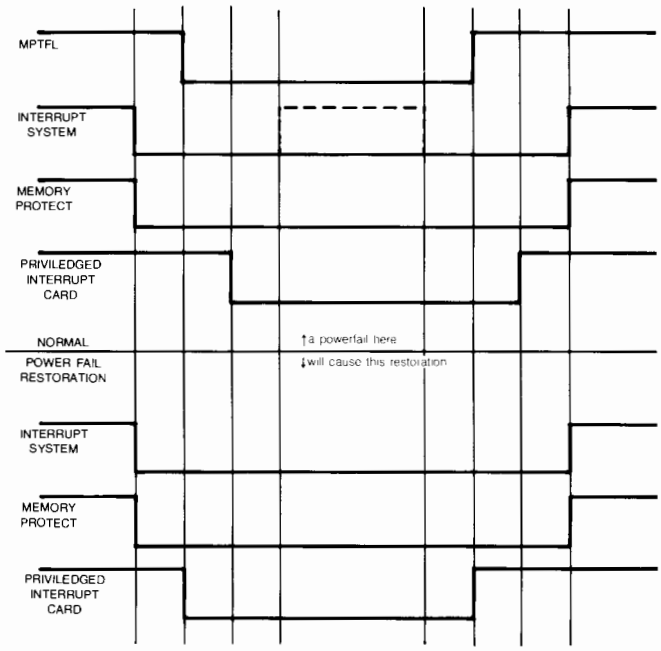
AUTOR

AUTOR is a user program written in FORTRAN and designed to be user modified. The supplied version scans the LU table (via EXEC 13 requests) looking for DVR00 and DVR05 subchannel 0 devices. To each of these it sends a powerfail message. AUTOR also calls the powerfail LU to get the powerfail time which it puts in the powerfail message. The message has two purposes. First, it notifies the user at each terminal that power was in fact lost and that the last line transferred may be in error. Second, if the terminal was inactive at the power failure, this call allows the driver to reenable the terminal so that it may respond to user attention (i.e., he hit a key) requests.

After all terminals have been notified AUTOR calls the powerfail LU for the powerfail time a second time. This second call resets the powerfail time in hand switch and signals complete recovery to the powerfail routine.

This concludes our discussion of the powerfail autor restart software. To get the most from this discussion, we suggest you reread it.

Figure 1



HIGH-SPEED INTERRUPT PROCESSING WITH RTE-II

David V. Fitterman
U.S. Geological Survey

Introduction

RTE-II users may on occasion find they need interrupt processing capabilities faster than the 1-2 milliseconds response time of the RTE-II system. The first solution which comes to mind is the use of a privileged interrupt card. With the privileged interrupt card assigned to select code 10, the interrupt can be recognized in 110 machine cycles or 220 microseconds. This approach has two drawbacks. First, it requires that two I/O slots be dedicated to the peripheral. Second, for some applications the privileged interrupt card will not be fast enough. We will now look at an alternative approach which can give the fastest possible interrupt processing. The price which must be paid, however, is the loss of multiprogramming capability while the high speed interrupt processing is taking place.

General Approach

The program is written as though it were a privileged routine, that is, the memory protect and interrupt hardware are disabled by a call to $LIBR. The contents of the trap cells corresponding to the peripherals being used are modified to jump to user supplied interrupt processing routines. Unused peripherals, such as the time base generator, are turned off. Once this is done the involved peripherals and interrupt system can be turned on, and high speed interrupt processing started. When the processing is completed, and the user is ready to restore the RTE system, the interrupt system is turned off, trap cells are restored to their original contents, any peripherals used are cleared, the time base generator is restarted, and finally memory protect and the interrupt system are turned on. Note that the real-time clock will be slow by the length of time the privileged routine was running. The outline of the general procedure for disabling and reenabling RTE-II is shown in Figure 1.

Specific Example

The technique described above was developed for a data transcription application. Geomagnetic variation data from portable magnetometers are recorded on magnetic cassettes using recorders manufactured by Sea Data Corp., Newton, MA. To achieve high packing density on the cassettes (1 million 12 bit words per 300 foot cassette) interrecord gaps are too small for the cassette reader transport to be stopped between records without passing over some data. This requires that the cassette reader be started and not stopped until all of the data has been transcribed to 9-track magnetic tape. The cassette reader presents a word to the interface every 500 microseconds except for the last word in a record which will be present for only 175 microseconds. Record gaps, occurring every 27-30 words depending upon the recorder format, are 1.25 milliseconds long giving a mean time between interrupts on the order of 530 microseconds. The listing of the program developed to do the data transcription is shown in Figure 2. Interrupts from the cassette are processed by routine CASS which places the data into a buffer. Ten buffers are used to form a ring structure. This many buffers are required to allow rewriting of tape records when parity/timing errors are encountered without losing any data coming from the cassette. The first word of each buffer is the length of the buffer, the second word points to the first word of the next buffer in the ring, and the third word contains the actual number of words stored in the buffer.

When CASS has completely filled a buffer it sets the sign bit of the first word (BUFi) of that buffer and moves on to the next buffer in the ring. The main program waits in a loop labelled IDLE for a buffer to be filled, an end of file mark to be encountered by CASS, or the operator to signal with the switch register that all of the data has been read. When any of these conditions are encountered the main program exits from the loop and writes a buffer onto tape. After satisfactorily writing the tape record, the sign bit of word BUFi is zeroed so that the buffer is available for filling by CASS. If a parity/timing error is detected during the tape write, the tape is backspaced over the bad record, a 4 inch gap is written, and the record is rewritten. By having 10 buffers more than a dozen consecutive bad writes can be accommodated without running out of storage for incoming data.

This program has one sophistication not mentioned in the previous section. Since it takes about 7 minutes to transcribe one cassette, the resulting delay in the real-time clock might be of concern to some users. This problem was solved by resetting the time base generator to tick every 10 seconds and counting the number of ticks. When transcription is complete the time base generator is reset to a 10 millisecond tick interval and the number of missed 10 second beats is returned to the father program to aid the operator in resetting the real-time clock. In applications where data transmissions take place infrequently and for only a short period of time, keeping track of the lost time would not be necessary. An example of this would be a real-time measurement program which is scheduled once an hour and takes less than a second to execute.

Conclusions

A procedure for setting up a non-RTE interrupt environment on an RTE system has been shown. The method provides interrupt processing speed limited only by the duration of the interrupt routine itself. The method could also be used to circumvent writing an RTE driver for special purpose equipment. The disadvantages of the method are that multiprogramming capability is lost, and the real-time clock will be slow.

Figure Captions

Figure 1. Outline of Procedure For Setting Up a Non-RTE Interrupt Environment on an RTE-II System

Figure 2. Listing of Data Transcription Program

Figure 1

```
      NAM BYRTE,3,80
      ENT BYRTE
      EXT $LIBR,$LIBX,EXEC
      ORB
APER1 DEF IPER1     INTERRUPT LINK ADDRESS
      ORG
BYRTE NOP
      JSB $LIBR     TURN OFF INTERRUPTS
      NOP           AND MEMORY PROTECT
      CLC 10B,C     TURN OFF TIME BASE GENERATOR
      CLC 11B,C     AND ALL OTHER DEVICES
       .
       .
       .
      CLC 21B,C
      LDA PER1      SAVE AND MODIFY TRAP CELLS
      STA SYS1
      LDA TCEL1
      STA PER1
      STC PER1,C    ARM PERIPHERAL
      STF 0         TURN ON INTERRUPT SYSTEM
*****************************************************
*
*                   MAIN PROGRAM FOR DATA HANDLING
*
*****************************************************
*
IPER1 NOP           ENTRY TO INTERRUPT
       .            PROCESSING ROUTINE
       .
      STC PER1,C    REARM PERIPHERAL
      JMP IPER1,C   RETURN
*
*****************************************************
*
*                   RESTORE RTE SYSTEM
*
      CLF 0         TURN OFF INTERRUPTS
      LDS SYS1      RESTORE TRAP CELLS
      STA PER1
      STF PER1      SET FLAG AND CLEAR CONTROL OF
      CLC PER1      ANY PERIPHERALS USED INCLUDING
*                   DMA CHANNELS TO PREVENT
*                   ILLEGAL INTERRUPTS
      STC TBG       TURN ON THE TIME BASE GENERATOR
      JSB $LIBX     TURN ON MEMORY PROTECT
      DEF ATURN     AND INTERRUPT SYSTEM
```

```
RTURN JSB EXEC      TERMINATE PROGRAM
      DEF *+2
      DEF COD6
ATURN DEF RTURN
PER1  EQU 13B       S.C.  OF PERIPHERAL
TBG   EQU 10B       S.C.  OF TIME BASE GENERATOR
SYS1  BSS 1         SAVE AREA FOR SYSTEM TRAP CELL
TCEL1 JSB APER1,I   MODIFIED TRAP CELL INSTRUCTION
      END BYRTE
```

Figure 2

```
0001  ASMB,L,T,C
0002        NAM PIRAT,3,80
0003        ENT PIRAT
0004        EXT EXEC,$LIBR,$LIBX,RMPAR,PRTN
0005  *
0006  * PROGRAM TO PIRATE AWAY RTE SYSTEM, SET UP A STAND
0007  * ALONE INTERRUPT ENVIRONMENT, TRANSCRIBE CASSETTE
0008  * TO TAPE, AND RESTORE THE RTE SYSTEM
0009  *
0010  *WRITTEN BY D. V. FITTERMAN
0011  *       BRANCH OF ELECTROMAGNETISM AND GEOMAGNETISM
0012  *       U. S. GEOLOGICAL SURVEY
0013  *       DENVER, COLORADO  80225
0014  *       JANUARY 1976
0015  *
0016        ORB
0017  ATAPE DEF TAPE        ADDRESSES
0018  ACASS DEF CASS          FOR
0019  ADMA  DEF DMA             INTERRUPT
0020  ATIME DEF TIME              ROUTINES
0021        ORR
0022  PIRAT NOP
0023        JSP RMPAR     PICK UP # OF WORDS/TAPE
0024        DEF *+2       RECORD
0025        DEF WDTP
0026        JSB $LIBR     TURN OF INTERRUPTS
0027        NOP           AND MEMORY PROTECT
0028        HLT 76B
0029  CLC6  CLC 6B,C      TURN OFF PERIPHERALS
0030        ISZ CLC6      INCREMENT INSTRUCTION
0031        ISZ NPER      COUNT NUMBER OF PERIPHERALS
0032        JMP CLC6      LOOP
0033        LDA 6B        SAVE & MODIFY
0034        STA SYS6        TRAP
0035        LDA TCDMA         CELLS FOR
0036        STA 6B                      DMA
0037        LDA 10B
0038        STA SYS10
0039        LDA TCTBG
0040        STA 10B                     TBG
0041        LDA 14B
0042        STA SYS14
0043        LDA TCTAP
0044        STA 14B                     MAG TAPE
0045        LDA 16B
0046        STA SYS16
0047        LDA TCCAS
0048        STA 16B                     CASSETTE
0049        LDA .105      SET TBG TO
0050        OTA TBG       TEN SECOND TICKS
0051        STC TBG       TURN ON TBG
0052        LIA 1B        SAVE SYSTEM
0053        STA SYSWR     SWITCH REGISTER
0054        CLA           CLEAR THE
0055        OTA 1B        SWITCH REGISTER
0056        STF 0         TURN ON INTERRUPTS
0057        LDB SEL0      SELECT TAPE
0058        JSB REJ       UNIT ZERO
0059        LDX =B0       ZERO "X"
0060        LDB PNT10     RESET BUFFER
0061  SET   LDA WDTP
0062        STA B,I       SET BUF# TO WORDS/TAPE RECORD
0063        ADB =D2       COMPUTE ADDRESS OF CNT #
0064        CLA
0065        STA B,I       ZERO CNT#
```

```
0066          ADB =D-1      COMPUTE ADDRESS OF PNT#
0067          LDB B,I       LOAD ADDRESS OF BUF#+1
0068          ISZ TEN       DONE?
0069          JMP SET       NO, LOOP
0070          LDA PNT10     YES, RESET BUFFER
0071          STA BUFR      2-B FILLED
0072          STA BOUT      AND 2-B OUTPUT
0073          STC PERI,C    TURN ON PERIPHERAL
0074  IDLE    LDA BOUT,I
0075          SSA           USE BIT = ZERO?
0076          JMP WRITE     NO, OUTPUT BUFFER
0077          LDA EOF       YES
0078          SSA           EOF = 1'S?
0079          JMP EXIT      YES, RESTORE SYSTEM
0080          LIA 1B        NO, LOAD DISPLAY REGISTER
0081          SLA,RSS       B0 = ONE?
0082          JMP IDLE      NO, IDLE SOME MORE
0083          LDB BUFR      YES, LOAD COUNT
0084          ADB =B2       OF PRESENT
0085          LDA B,I             BUFFER
0086          SZA           COUNT = ZERO?
0087          JMP WRITE     NO, OUTPUT BUFFER
0088          JMP EXIT      YES, EXIT
0089  WRITE   ISZ TPRC      INCREMENT # OF TAPE RECORDS
0090          CCA           SET DMA STATUS WORD
0091          STA DMASW
0092          LDA CW1       OUTPUT 1ST DMA
0093          OTA 6B          CONTROL WORD
0094          CLC 2B        PREPARE FOR 2ND DMA CONTROL WORD
0095          LDA BOUT      COMPUTE STARTING
0096          ADA =B3         ADDRESS OF TRANSFER
0097          OTA 2B        OUTPUT SECOND CONTROL WORD
0098          ADA =D-1
0099          LDB A,I       LOAD WORD COUNT
0100          CMB,INB       NEGATE
0101          STC 2B        PREPARE FOR 3RD CONTROL WORD
0102          OTB 2B        OUTPUT 3RD CONTROL WORD
0103          LDB WCC       LOAD WRITE COMMAND
0104          JSB REJ       REJECTED?
0105          STC CC,C      START TAPE
0106          STC DC,C      START DATA
0107          STC 6B,C      START DMA
0108          LDA DMASW     WAIT FOR DMA TO FINISH
0109          SZA
0110          JMP *-2
0111          SFC CC        WAIT FOR TAPE INTERRUPT
0112          JMP *-1       BEFORE TESTING TAPE STATUS WORD
0113          LDA TPSWD     LOAD TAPE STATUS WORD
0114          SZA           STATUS WORD = ZERO?
0115          JMP ERROR     NO, BACKSPACE, GAP & REWRITE TAPE
0116          LDB BOUT      YES, LOAD FIRST WORD
0117          LDA B,I         OF OUTPUT BUFFER
0118          ELA,CLE,ERA   CLEAR SIGN BIT
0119          STA B,I       STORE
0120          INB           CHAIN TO NEXT
0121          LDA B,I         BUFFER
0122          STA BOUT      STORE ADDRESS IN BOUT
0123          INB           INCREMENT TO WORD COUNT
0124          CLA             ZERO AND
0125          STA B,I           STORE IN COUNT WORD
0126          JMP IDLE      GO IDLE AGAIN
0127  ERROR   ISZ BADTP     INCREMENT # OF BAD TAPE WRITES
0128          LDB BSR       BACK SPACE A RECORD
0129          JSB REJ       REJECTED?
0130          STC CC,C      START TAPE
0131          LDB GAP       WRITE A GAP
0132          JSB REJ       REJECTED?
0133          STC CC,C      START TAPE
0134          JMP WRITE+1   TRY TO WRITE RECORD AGAIN
0135  *
0136  *  TAPE INTERRUPT ROUTINE
0137  *
0138  TAPE    NOP
0139          CLC DC,C      TURN OFF DC, START EOR SEQUENCE
0140          STA TPSVA     SAVE "A"
0141          LIA CC        INPUT AND STORE TAPE
0142          STA TPSWD     STATUS WORD
0143          LDA TPSVA     RESTORE "A"
0144          CLC CC,C      TURN OFF TAPE UNIT
0145          JMP TAPE.I
0146  *
0147  * CASSETTE INTERRUPT ROUTINE
0148  *
0149  CASS    NOP
0150          CLF 0         TURN OFF INTERRUPTS
0151          STA CASA      SAVE "A"
0152          STB CASB      SAVE "B"
0153          ERA,ALS       "E" - A15
0154          SOC           "0" SET?
0155          INA           YES, INCREMENT "A"
0156          STA CASEO     SAVE "E" AND "0"
0157          ISX           INCREMENT "X"
0158          LDB BUFR
0159          ADB =B2       INCREMENT WORD COUNT
0160          ISZ B,I           COUNT
0161          LIA PERI      INPUT DAT WORD
0162          AND =B177     REMOVE TTY PARITY BIT
0163          SAX B,I       STORE WORD
0164          SLA,RSS       A0 = ONE?
0165          JMP FUL       NO, DATA WORD
0166          ISZ CASRC     YES, MESSAGE WORD INCREMENT CASRC
0167          TBS =B6 A     PARITY ERROR?
0168          JMP *+2       NO, TEST FOR DATA ERROR
0169          JMP ERR       YES, INCREMENT BADRC
0170          TSB =B10 A    DATA ERROR #1?
0171          JMP ERR       YES, INCREMENT BADRC
0172          TBS =B20 A    NO, DATA ERROR #2?
0173          JMP ERR       YES, INCREMENT BADRC
0174          TBS =B40 A    NO, DATA ERROR #3?
0175          JMP ERR       YES, INCREMENT BADRC
0176          JMP ERR+1     NO, TEST FOR END OF FILE
0177  ERR     ISZ BADRC
0178          TBS =B100 A   END OF FILE?
0179          RSS           YES
0180          JMP FUL       NO, TEST IF BUFFER IS FULL
0181          CCA           SET EOF
0182          STA EOF             WORD
0183          LDA BOUT,I    SET
0184          ELA              USE
0185          CCE                BIT
0186          ERA                   OF
0187          STA BOUT,I               BOUT,I
0188  RESTR   LDA CASEO     RESTORE
0189          CLO
0190          SLA,ELA       "E"
0191          STF 1B            AND "0"
0192          LDA CASA      "A"
0193          LDB CASB          AND "B"
0194          STC PERI,C    ARM  THE PERIPHERAL
0195          STF 0         TURN ON INTERRUPTS
0196          JMP CASS,I    RETURN
0197  FUL     CXA           "X" TO "A"
0198          CPA BUFR,I    "A" = BUFFER SIZE?
0199          RSS           YES, SKIP AND CHAIN
0200          JMP RESTR     NO, RESTORE
0201          LDX =B0       CLEAR "X", BUFFER COUNTER
0202          LDA BOUT,I    SET
0203          ELA              USE
0204          CCE                BIT
0205          ERA                   OF
0206          STA BOUT,I               BOUT,I
0207          LDB BUFR      CHAIN TO
0208          INB             NEXT
0209          LDA B,I           BUFR
0210          STA BUFR            BUFFER
0211          LDB A,I       LOAD BUF#
0212          SSB,RSS       USE BIT SET?
0213          JMP RESTR     NO, ALL'S O.K.  RESTORE!
0214          HLT 77B       YES, CAUGHT YOUR OWN TAIL!!!
0215          JMP EXIT      RESTORE SYSTEM
0216  *
0217  * DMA INTERRUPT ROUTINE
```

```
0218  *
0219  DMA    NOP
0220         STA DMAA    SAVE "A"
0221         CLA         ZERO DMA STATUS
0222         STA DMASW     WORD
0223         LDA DMAA    RESTORE "A"
0224         CLC 6B,C    CLEAR DMA CHANNEL
0225         JMP DMA,I
0226  *
0227  * TIME BASE GENERATOR INTERRUPT ROUTINE
0228  *
0229  TIME   NOP
0230         ISZ TICK    INCREMENT LOST TICKS
0231         NOP
0232         STC TBG,C
0233         JMP TIME,I
0234  *
0235  * REJECT SUBROUTINE FOR TAPE DRIVE
0236  *
0237  REJ    NOP
0238         OTB CC      OUTPUT COMMAND WORD
0239         LIA CC      LOAD STATUS WORD
0240         RAR,RAR     BIT 3 ON?
0241         RAR,SLA
0242         JMP REJ+1   YES, TRY AGAIN
0243         JMP REJ,I   NO, RETURN
0244  *
0245  * RESTORE RTE SYSTEM
0246  *
0247  EXIT   CLF 0        TURN OFF INTERRUPTS
0248         LDA SYS6    RESTORE TRAP CELLS
0249         STA 6B
0250         LDA SYS10
0251         STA 10B
0252         LDA SYS14
0253         STA 14B
0254         LDA SYS16
0255         STA 16B
0256         LDA SYSWR   RESTORE SYSTEM
0257         OTA 1B      SWITCH REGISTER
0258         STF 6B      SET FLAGS OF PERIPHERALS USED
0259         STF 13B
0260         STF 14B
0261         STF 16B
0262         CLC 6B      CLEAR CONTROL OF PERIPHERALS USED
0263         CLC 13B
0264         CLC 14B
0265         CLC 16B
0266         LDA .10MS   RESTORE TBG
0267         OTA TBG     TO 10MS TICKS
0268         STC TBG     TURN ON TIME-BASE GENERATOR
0269         JSB $LIBX   TURN ON INTERRUPTS
0270         DEF ATURN   AND MEMORY PROTECT
0271  RTURN  JSB PRTN    PASS BACK ERROR PARAMETERS TO FATHER
0272         DEF *+2
0273         DEF CASRC
0274         JSB EXEC    TERMINATE PROGRAM
0275         DEF *+2
0276         DEF COD6
0277  ATURN  DEF RTURN
0278  A      EQU 0B
0279  B      EQU 1B
0280  DC     EQU 13B     DATA CHANNEL
0281  CC     EQU 14B     COMMAND CHANNEL
0282  PERI   EQU 16B     SELECT CODE OF PERIPHERAL
0283  TBG    EQU 10B
0284  NPER   DEC -11     -# OF S.C.'S
0285  TEN    DEC -10     -# OF BUFFERS IN RING
0286  .10MX  OCT 2       TBG 10MS CONTROL WORD
0287  .10S   OCT 5       TBG 10S  CONTROL WORD
0288  *
0289  * MODIFIED TRAP CELL INSTRUCTIONS
0290  *
0291  TCTBG JSB ATIME,I
0292  TCDMA JSB ADMA,I
0293  TCTAP JSB ATAPE,I
0294  TCCAS JSB ACASS,I
0295  *
0296  * STORAGE FOR SYSTEM TRAP CELLS
0297  *
0298  SYS6   BSS 1       DMA
0299  SYS10  BSS 1       TBG
0300  SYS14  BSS 1       TAPE
0301  SYS16  BSS 1       PERIPHERAL
0302  SYSWR  BSS 1       SYSTEM SWITCH REGISTER
0303  *
0304  * TAPE CONTROLLER COMMANDS
0305  *
0306  GAP    OCT 15      WRITE 4 INCHES BLANK TAPE
0307  WCC    OCT 31      WRITE ONE RECORD
0308  BSR    OCT 41      BACKSPACE ONE RECORD
0309  SEL0   OCT 1400    SECT UNIT 0
0310  CW1    OCT 20013   1ST DMA CONTROL WORD
0311  COD6   DEC 6
0312  *
0313  * BUFFER AREA
0314  *
0315  DMASW  OCT 0       DMA STATUS WORD
0316  TPSWD  BSS 1       TAPE STATUS WORD
0317  TPSVA  BSS 1       TAPE INTERRUPT SAVE AREA "A"
0318  CASA   BSS 1       CASSETTE INTERRUPT SAVE AREA "A"
0319  CASB   BSS 1                                    "B"
0320  CASEO  BSS 1                                  "E" & "O"
0321  DMAA   BSS 1       DMA INTERRUPT SAVE AREA "A"
0322  EOF    OCT 0       END OF FILE FLAG
0323  WDTP   BSS 5       # OF WORDS/TAPE RECORD
0324  CASRC  OCT 0       # OF CASSETTE RECORDS TRANSCRIBED
0325  TPRC   OCT 0       # OF TAPE RECORDS WRITTEN
0326  BADRC  OCT 0       # OF BAD CASSETTE RECORDS READ
0327  BADTP  OCT 0       # OF BAD TAPE WRITES
0328  TICK   OCT -1      # OF TBG TICKS LOST
0329  BUFR   BSS 1       ADDRESS OF BUFFER BEING FILLED
0330  BOUT   BSS 1       ADDRESS OF BUFFER TO BE OUTPUT
0331  BUF1   BSS 1
0332  PNT1   DEF BUF2
0333  CNT1   BSS 1022
0334  BUF2   BSS 1
0335  PNT2   DEF BUF3
0336  CNT2   BSS 1022
0337  BUF3   BSS 1
0338  PNT3   DEF BUF4
0339  CNT3   BSS 1022
0340  BUF4   BSS 1
0341  PNT4   DEF BUF5
0342  CNT4   BSS 1022
0343  BUF5   BSS 1
0344  PNT5   DEF BUF6
0345  CNT5   BSS 1022
0346  BUF6   BSS 1
0347  PNT6   DEF BUF7
0348  CNT6   BSS 1022
0349  BUF7   BSS 1
0350  PNT7   DEF BUF8
0351  CNT7   BSS 1022
0352  BUF8   BSS 1
0353  PNT8   DEF BUF9
0354  CNT8   BSS 1022
0355  BUF9   BSS 1
0356  PNT9   DEF BUF10
0357  CNT9   BSS 1022
0358  BUF10  BSS 1
0359  PNT10  DEF BUF1
0360  CNT10  BSS 1022
0361         END PIRAT
```

RTE PERFORMANCE

Lyle Weiman/DSD

Do you have a performance criteria for a proposed system, and want to know if RTE can handle it? Have you tried to find out whether the system can usefully be expanded (e.g., more terminals) and been unable to find out? Are you designing a system, and need to know the most efficient way to utilize your RTE?

This article provides measurement data for some RTE II and FORTRAN-support library functions (SIN, COS, ATAN, etc.), describes how the measurements were made, and will help you make measurements of your own if you don't find what you want in this data.

You will use different measurement techniques, depending on whether the system can always immediately perform the request you wish to measure (e.g., library functions such as SIN, COS, etc.), or whether it must wait for resources to become available (DMA, system available memory, disc space, or memory partition). I/O measurements are typical of the latter.

In the first case, you simply write a program which requests the function. You put this in a loop, to be executed a large number of times, typically 1000 to 10,000. You choose the number such that overall elapsed time is around a minute to two minutes, so that the 10 ms uncertainty in time interval measurement (you use the RTE Time-of-day call to measure the elapsed time interval) is small compared to the elapsed time. Divide the elapsed time by the loop count to get the request service time. In cases such as library functions, the execution time depends on argument value, so you will want to select some typical and worst-case argument values. You must subtract the RTE time-base clock service overhead from this figure.

You measure TBG time with the program shown in Figure 1 (SYSOH), and you run it when the system is not running any user programs.

Figure 1

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

0001  FTN4,L
0002  C     6/03/76 WEIMAN
0003        PROGRAM SYSOH
0004  C
0005  C
0006  C     MEASURES BASIC SYSTEM OVERHEAD ON RTE SYSTEMS.
0007  C     MAY BE RUN ALONE, OR USED IN CONJUNCTION WITH
0008  C     ANOTHER PROGRAM TO MEASURE THE SYSTEM OVERHEAD
0009  C     THAT PROGRAM INTRODUCES.
0010  C
0011  C
0012  C
0013  C     SCHEDULE PARAMETERS:
0014  C             #1=TTY LU FOR MESSAGE OUTPUT.
0015  C             #2=CPU: <2 = 2100
0016  C                      2 = 21MX
0017  C                     >2 = 21XE
0018  C
0019  C     SEQUENCE OF OPERATIONS:
0020  C     1)GET START TIME
0021  C     2)EXECUTE A FIXED TIME'S WORTH OF INSTRUCTIONS.
0022  C     3)GET FINISHED TIME.
0023  C     4)PRINT THE DIFFERENCE BETWEEN ELAPSED TIME AND
0024  C       EXECUTION TIME, AS A PERCENTAGE OF ELAPSED TIME.
0025  C
0026        INTEGER STIME(5),FTIME(5),IPRAM(5),IOFF,NTIME,LU
0027        INTEGER STIME1,STIME2,STIME3,STIME4,STIME5
0028        INTEGER FTIME1,FTIME2,FTIME3,FTIME4,FTIME5
0029        INTEGER CPU
0030        REAL XTIME
0031  C     EXECUTION TIME = DATA STORED IN "XTIME"
0032        EQUIVALENCE (IPRAM(1),LU)
0033        EQUIVALENCE (IPRAM(2),CPU)
0034        EQUIVILENCE (IPRAM(3),NCHAR)
0035        EQUIVALENCE (FTIME(1),FTIME1)
0036        EQUIVALENCE (FTIME(2),FTIME2),  (FTIME(3),FTIME3)
0037        EQUIVALENCE (FTIME(4),FTIME4),  (FTIME(5),FTIME5)
0038        EQUIVALENCE (STIME(1),STIME1),  (STIME(2),STIME2)
0039        EQUIVALENCE (STIME(3),STIME3),  (STIME(4),STIME4)
0040        EQUIVALENCE (STIME(5),STIME5)
0041        DATA XTIME/56.85568/
0042  C
0043  C
0044  C
0045  1     FORMAT(" CPU IS 2100A")
0046  2     FORMAT(" CPU IS 21MX")
0047  3     FORMAT(" CPU IS 21XE")
0048  C
0049  C     GET SCHEDULE PARAMETERS
0050        CALL PMPAR(IPRAM)
0051  C     DEFAULT TTY-LU
0052        IF(LU .LT. 1) LU=1
0053  C
0054  C     USE PROPER EXECUTION TIME FOR COMPUTER BEING USED.
0055  C
0056        IF(CPU .LT. 2) WRITE(LU,1)
0057        IF(CPU .LT. 2) XTIME=56.85568
0058        IF(CPU .EQ. 2) XTIME=64.40780
0059        IF(CPU .EQ. 2) WRITE(LU,2)
0060        IF(CPU .GT. 2) WRITE(LU,3)
0061        IF(CPU .GT. 2) XTIME=104.7320
0062  C
0063  C
0064  C
0065  C     GET START TIME
0066  500   CALL EXEC(11,STIME)
0067  C     LOOP
0068        DO 1000 I=1,1000
0069        DO 1000 J=1,1000
0070        DO 1000 L=1,9
0071  1000  CONTINUE
0072  C     GET FINISHED TIME.
0073        CALL EXEC(11,FTIME)
0074  C
0075  C     COMPUTE ELAPSED TIME
0076  C
0077        ETIME=(FTIME/STIME)*.01 +(FTIME2-STIME2) +
0078       1 (FTIME3-STIME3)*60. +(FTIME4-STIME4)*3600.
0079        IF(FTIME5 .NE. STIME5) ETIME=ETIME+86400.
0080  C
0081  C     PRINT ELAPSED TIME  CPU LOAD AS PERCENTAGE
0082  C     OF ELAPSED TIME.
0083  2000  FORMAT(" ELAPSED TIME="F8.2"SECS.CPU LOAD="F6.3"%")
0084        CPULOD=(ETIME-XTIME)/ETIME *100.0
0085        WRITE(LU,2000) ETIME, CPULOG
0086  2400  CONTINUE
0087  C
0088  C
```

```
0089  2700  CONTINUE
0090        CALL EXEC(3,1100B+LU,-1)
0091  3000  END

**  NO ERRORS**      PROGRAM = 00304      COMMON = 00000
```

Figure 2

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

0001  FTN4,L
0002  C     5/17/76 WEIMAN
0003        PROGRAM OVRHD
0004        COMMON IBUSY,ICOUNT,ICNTR2,IOPT,ISTOP
0005  C
0006  C
0007  C     PROGRAM TO MAKE SYSTEM OVERHEAD MEASUREMENTS
0008  C     ON RTE-II SYSTEMS
0009  C     COMMON COMMUNICATION:  IBUSY=FLAG, SET BY
0010  C     OVRHD WHEN IT WANTS THE FOREGROUND PROGRAM
0011  C     TO DO SOMETHING.  IT IS CLEARED WHEN THAT
0012  C     TASK IS DONE.
0013  C     ICOUNT,ICNTR2 FORM A TWO-WORD COUNTER
0014  C     IOPT=NUMBER OF WORDS/CHARACTERS
0015  C
0016  C
0017  C     SCHEDULE PARAMETERS:
0018  C             #1=TTY LU FOR MESSAGE OUTPUT.
0019  C             #2=CPU: <2 = 2100
0020  C                      2 = 21MX
0021  C                     >2 = 21XE
0022  C            #3= NUMBER OF CHARACTERS PRINTED.
0023  C              + = WORDS, -=CHARS.
0024  C     SEQUENCE OF OPERATIONS:
0025  C     1)GET START TIME
0026  C     2)EXECUTE A FIXED TIME'S WORTH OF INSTRUCTIONS.
0027  C     3)GET FINISHED TIME.
0028  C     4)PRINT THE DIFFERENCE BETWEEN ELAPSED TIME
0029  C       AND  EXECUTION TIME, AS A PERCENTAGE OF
0030  C       ELAPSED TIME.
0031        INTEGER STIME(5),FTIME(5),IPRAM(5),LU
0032        INTEGER STIME1,STIME2,STIME3,STIME4,STIME5
0033        INTEGER FTIME1,FTIME2,FTIME3,FTIME4,FTIME5
0034        INTEGER IPROG(3)
0035        INTEGER CPU
0036        REAL CLOCK,XTIME
0037  C     EXECUTION TIME = DATA STORED IN "XTIME"
0038        EQUIVALENCE (IPRAM(1),LU)
0039        EQUIVALENCE (IPRAM(2),CPU)
0040        EQUIVALENCE (IPRAM(3),NCHAR)
0041        EQUIVALENCE (FTIME(1),FTIME1)
0042        EQUIVALENCE (FTIME(2),FTIME2), (FTIME(3),FTIME3)
0043        EQUIVALENCE (FTIME(4),FTIME4), (FTIME(5),FTIME5)
0044        EQUIVALENCE (STIME(1),STIME1), (STIME(2),STIME2)
0045        EQUIVALENCE (STIME(3),STIME3), (STIME(4),STIME4)
0046        EQUIVALENCE (STIME(5),STIME5)
0047        DATA XTIME/56.85568/
0048  C
0049  C     SET THE FOREGROUND PROGRAM'S NAME
0050  C
0051        DATA IPROG/2HT0,2H00,2H2 /
0052  C
0053  C
0054  C     GET SCHEDULE PARAMETERS
0055        CALL PMPAR(IPRAM)
0056  1     FORMAT(" CPU IS 2100A")
0057  2     FORMAT(" CPU IS 21MX")
0058  3     FORMAT(" CPU IS 21XE")
0059  C     DEFAULT TTY LU
0060        IF(LU .LT. 1) LU=1
0061        IOPT=NCHAR
0062  401   FORMAT(" # CHARACTERS="I5)
0063        WRITE(LU,401) IOPT
0064  C
0065  C     USE PROPER EXECUTION TIME FOR COMPUTER BEING
0066  C     USED.
0067        IF(CPU .GE. 2) GOTO 5
0068  C                    COMPUTER IS 2100
0069        WRITE(LU,1)
0070        XTIME=56.85568
0071        GOTO 15
0072  5     CONTINUE
0073        IF(CPU .GT. 2) GOTO 6
0074  C                    COMPUTER IS 21MX
0075        WRITE(LU,2)
0076        XTIME=64.40780
0077        GOTO 15
0078  6     CONTINUE
0079  C                    COMPUTER IS 21MX-E SERIES
0080        WRITE(LU,3)
0081        XTIME=34.91070
0082  15    CONTINUE
0083  C
0084  C
0085  C     SCHEDULE "SLAVE" TASK PROGRAM
0086  C
0087        IBUSY=0
0088        ISTOP=0
0089  100   CALL EXEC(10,IPROG,IOPT)
0090  C
0091  C     WAIT FOR IT TO COME IN FROM THE DISC
0092  C
0093        IF(IBUSY .EQ. 0) GOTO 100
0094  C
0095  C     IF PROGRAM HAS ALREADY FINISHED, SKIP WAIT
0096  C     LOOP.
0097        IF(IBUSY .EQ. -2) GOTO 1050
0098  C
0099  C
0100  C     GET START TIME
0101  500   CALL EXEC(11,STIME)
0102  C     LOOP
0103        DO 1000 I=1,1000
0104        DO 1000 J=1,1000
0105        DO 1000 L=1,3
0106  1000  CONTINUE
0107  C     GET FINISHED TIME.
0108        CALL EXEC(11,FTIME)
0109  C     GET # OF COUNTS
0110  C
0111  C     CONVERT NUMBER USING "ICOUNT" AS LOW 16 BITS,
0112  C     AND "ICNTR2" AS HIGH 15 BITS.
0113  C
0114  1050  CONTINUE
0115        IIJ=ICOUNT
0116        IIK=ICNTR2
0117  C
0118  C     IF PROGRAM ALREADY STOPPED, SKIP WAIT
0119  C
0120        IF(IBUSY .EQ. -2) GOTO 1020
0121  C
0122  C     SIGNAL PROGRAM TO TERMINATE
0123  C
0124        ISTOP=-1
0125  1010  IF(STOP .GE. 0) GOTO 1010
0126  C
0127  C     CONVERT COUNTERS
0128  C
0129  1020  CONTINUE
0130        III=IAND(IIJ,77777B)
0131        FTN=III
0132        IF(IIJ .LT. 0) FIN=FIN+32768.0
0133        FIN=FIN+IIK*65536.0
0134  C
0135  C     COMPUTE ELAPSED TIME
0136  C
0137        FTIME=(FTIME-STIME)*.01 +(FTIME2-STIME2) +
0138       1 (FTIME3-STIME3)*60. +(FTIME4-STIME4)*3600.
0139        IF(FTIME5 .NE. STIME5) ETIME=ETIME+86400.
0140  C
```

```
0141  C     PRINT ELAPSED TIME,CPU LOAD AS PERCENTAGE
0142  C     OF ELAPSED TIME, AND NUMBER OF EVENTS.
0143  2000  FORMAT(" ELAPSED TIME="F8.2"SECS.CPU
0144        LOAD="F6.3 "% #EVENTS="F8.0)
0145        CPULOD=(ETIME-XTIME)/ETIME *100.0
0146        WRITE(LU,2000) ETIME,CPULOG,FIN
0147  2400  CONTINUE
0148  C
0149  C
0150  C
0151        CALL EXEC(3,1100B+LU,-1)
0152        END
0153        END$

**  NO ERRORS**     PROGRAM = 00402     COMMON = 00005
```

Figure 3

```
0001                ASMB,L
0002  00000                 NAM T0002,2,70 5 17 76 3:30 PM
0003                        SUP
0004                        COM BUSY,COUNT,CNTR2,IOPT,ISTOP
0005                        EXT EXEC
0006  00000         T0002 EQU *
0007  00000 002400         CLA         ZERO THE
0008  00001 072001C        STA COUNT   COUNTER
0009  00002 072002C        STA CNTR2
0010  00003 002004         INA
0011  00004 072000C        STA BUSY
0012  00005         LOOP  EQU *
0013  00005 062004C        LDA ISTOP
0014  00006 002020         SSA
0015  00007 026023R        JMP STOP
0016  00010 016001X        JSB EXEC    OUTPUT SOME
0017  00011 000016R        DEF *+5
0018  00012 000032R        DEF D2        CHARACTERS
0019  00013 000031R        DEF D1
0020  00014 000033P        DEF MSG
0021  00015 000003C        DEF IOPT
0022  00016 036001C        IS7 COUNT   INCREMENT THE COUNTER
0023  00017 026005R        JMP LOOP
0024  00020 036002C        ISZ CNTR2   ROLLED OVER. BUMP
                                       OTHER COUNTER
0025  00021 000000         NOP
0026  00022 026005R        JMP LOOP
0027*
0028  00023         STOP  EQU *
0029  00023 002400         CLA         SET "STOPPED"
0030  00024 072004C        STA ISTOP    FLAG
0031  00025 016001X        JSB EXEC    TERMINATE
0032  00026 000030R        DEF *+2
0033  00027 000030R        DEF D6
0034  00030 000006  D6     DEC 6
0035  00031 000001  D1     DEC 1
0036  00032 000002  D2     DEC 2
0037  00033 040523  MSG    ASC 28,ASASDFASDFASDFASDFASDF
                           ASDFASDFASDFASDF
0038                       END T0002
**  NO ERRORS *TOTAL **RTE ASMB 760924**
```

This program is your basic work-horse measurement tool. The principal of operation is simple: when RTE is not updating the system clock, responding to other interrupts and running other user programs, it can run SYSOH. Usually, you'll want this program to occupy a partition which is otherwise unused so that no swapping takes place. Looking at the listing (Fig. 1), lines 68 thru 71, notice the three imbedded DO-loops. Execution of these loops consumes a fixed number of instruction cycles, and therefore will always take the same amount of time, *unless the system is doing something else*. (Time Base Generator service, interrupt processing, other user programs, etc.). This time interval has been calculated for 2100, 21MX and 21MX-E computers, and coded into the program (see variable XTIME, lines 57 thru 61). Using the system's own time-keeping, SYSOH subtracts the amount of time these DO-loops were executing from the actual elapsed time, divides by the elapsed time, converts this figure to a percentage (line 77 thru 79) and prints it. The percentage is the percentage of CPU time RTE was doing "something else." You control what that "something else" is when you set up the experiment. When the system is otherwise quiet, only TBG service is being done.

You should run this program with the system quiet and record the CPU utilization in percent. Since TBG service is always being done, you must subtract this number (in percent) from all your other measurement data.

The next step is to actually measure something. Remember, this technique can only be used when the system service to be measured consists of some amount of CPU time, and some amount of "wait time". Let's take I/O, for example. Suppose you want to know how much CPU time is consumed outputting characters to a terminal. Some modifications to your basic measurement program are required, and these are shown in Figure 2 (OVRHD). Another program is required to actually perform the service, in this case output to a terminal. This program is shown in Figure 3 (T0002).

T0002 has a higher priority than OVRHD. Both programs must be in memory at the same time for this measurement, so OVRHD is loaded in background, T0002 is a real-time (foreground) program. In RTE III, any two partitions will do. The important thing is to be sure the two programs never swap each other. The two programs communicate via background *system* COMMON (note: T0002 is loaded using reverse COMMON. Background System COMMON *must* be used local COMMON will not do). The communication consists of two counters, start/stop flags, and the number of characters/words to send (IOPT).

The counters are used to count the number of EXEC calls. The start flag is used to signal when the program has started. The "STOP" flag is used to signal T0002 that it should turn itself off (alternatively, you could simply "abort" it via EXEC call).

The example shown in Figure 2 includes code to schedule the "son" program (the one which does the I/O), wait for it to set its "start" flag, signalling that it has been read in from the disc and begun to execute, (lines 87 thru 93) and code to signal for it to "stop". It converts the two event-counter words into what is basically a 24-bit integer (lines 115 thru 133), although floating-point arithmetic is used. The results are printed as elapsed time, CPU utilization in percent, and number of "events", i.e., number of times EXEC was called.

In order to calculate the CPU overhead, subtract the "quiet system" overhead (in percent), multiply by elapsed time to get time (in seconds) occupied by I/O processing. Divide by the number of events to get CPU time per EXEC call. Divide the overall elapsed time (less clock overhead) by number of events to get elapsed time per EXEC call.

Before you run off to use this information, you should realize that the measurement you just made includes the I/O initiation time, all the interrupts and I/O completion. If your device uses DMA and requires no interrupts on a character or word basis, you'll need to determine how much overhead is required for the various parts. You do this by repeating the measurements above for various word/character lengths, say, 1,51 and 101.

CPU (sec.) = Initiation+#interruptsXinterrupt times+completion
= I+nc+compl

where n = # interrupt
c = continuation (interrupt) processing time

Since initiation and completion times seldom need to be separated out, we will just lump them together under I. So with the data from various character/word lengths, you can solve for the two unknowns (I and C). You only need two data points for this, but I suggest three, in order to see how accurate your measurements were. The values calculated for I an C using all three data points should agree within about 5%. If not, the device you're using may be interrupting more often than you think.

You can use this technique to measure I/O (regular, class buffered and re-entrant), optimize disc throughput, etc.

ACCURACY AND REPEATABLILITY

The time intervals you are measuring are accurate to 10 ms., since the error in the TBG itself is insignificant compared to the others.

If your elapsed time interval is, say, 100 seconds, then that is an error of .01%. Your count may be off by one, depending on when the "father" sampled it. It is wise to make several runs and take the average.

Each installation and each RTE is slightly different, and those who strive for the ultimate accuracy in these measurements should be aware that under actual, dynamic operation, an RTE can reasonably be expected to exhibit more overhead than the sum you'll get by adding up the overhead measured for each task measured separately. This is because, with more programs, the lists are longer, programs compete for DMA and S.A.M. resources, etc., and these effects are not visible when measured in a "quiet" system, even though they are generally small.

WRITING SUBROUTINES TO USE THE FAIL ERROR OPTION IN BASIC

Jim Bridges/DSD

The BASIC manual (92101-90016, section 6-4) describes the use of the FAIL ERROR option with certain HP-supplied subroutines. It is easy to make use of this option in subroutines the user may write. The following example is taken from the BASIC manual:

```
100 CALL TRNON (200,122536) FAIL: GO TO 9000
```

TRNON (and other HP-supplied routines) can generate error messages of the form:

```
ERROR n IN LINE xxx
```
(where n is an error code)

Without the FAIL ERROR option, the program would abort upon the error. By using the FAIL ERROR option, the BASIC program can process the error. However, the error message will still be printed and there is no way to avoid this message as long as you are using the HP routines. But, the message can be avoided in subroutines the user may write for himself.

The key to FAIL ERROR processing is the location ERRCD, which is in the program CALSB (included in 92101-12003). CALSB is the parameter passer linkage between BASIC overlays and the program written in the BASIC language. After completion of a BASIC CALL statement, the value in ERRCD is passed back to the interpreter and is checked: if ERRCD is non-zero, then the FAIL ERROR branch is taken. Note that the FAIL ERROR branch must be included in the CALL statement if ERRCD is non-zero: else the message

```
SUB. OR FUNCT. TERMINATED ABNORMALLY IN LINE xxx
```

will be printed and the BASIC program aborted.

The following is a sample subroutine by which the user might set a value in ERRCD:

```
ASMB,R,L
      NAM PASS,7 SUBROUTINE TO SET ERRCD
      ENT PASS
      EXT .ENTR,ERRCD
ICODE BSS 1
PASS  NOP
      JSB .ENTR
      DEF ICODE
      LDA ICODE,I
      STA ERRCD
      JMP PASS,I
      END
```

Here is a FORTRAN subroutine which uses PASS and a sample BASIC program to call it:

```
FTN4,L
      SUBROUTINE TRYIT (IFLAG,ICODE)
C IF  IFLAG # 0, THEN SET ERRCD TO ICODE
      IF (IFLAG.EQ.0) GO TO 900
      CALL PASS (ICODE)
900   RETURN
      END
```

```
10 PRINT "INPUT A,N (A#0:TAKE ERR EXIT/N=ERRCD)";
20 INPUT A,N
30 CALL TRYIT (A,N) FAIL: GO TO 50
40 STOP
50 PRINT "ERRCD IS NOW ";IERR(0)
60 END
```

Notice that ERRCD is examined in BASIC with the function IERR, which has a single dummy parameter (not used but necessary). Also that ERRCD may be set in the BASIC program by the function SERR (which also has a single dummy parameter). IERR and SERR are explained in the BASIC manual.

KEEP PRESSING THOSE SOFT KEYS !!!

Gary Gubitz/DSD

RTE support for the "user-programmable soft keys" of the HP 2645A Display Station has arrived! Distributed with all HP 1000 systems after April 1, are two utilities which will greatly simplify the task of programming these Special Function keys.

In issue #12 of the Communicator you may remember an article "Start Pressing Those Soft Keys!!!", which described applications for soft keys that would help make your system friendlier. The same article also described a Contributed Library program which allowed one to get their hands on the real potential behind this soft key feature. Well, we went on to enhance this Contributed Library program, and thus created two super utilities that are used in conjunction with the 2645A Display Station. They are:

1. KEYS - a program that provides a simple operator interface for generating command sets in a standard format that will program the 2645A soft keys.

2. KYDMP - a program that provides the capability of outputting a soft key command set, created by the KEYS program, from a discfile or a 2645A mini-cartridge file or LU to a 2645A Display Station to program its soft keys.

When run, KEYS can perform the following functions:

1. Create a new soft key command set.

2. Modify a command set that exists in a file or a logical unit number.

3. List the eight soft key labels, types, and command strings of the current command set or one that exists in a file or a logical unit number.

4. Output the current soft key command set or one that exists in a file or a logical unit number to a 2645A to program its soft keys.

5. Output a soft key command set to a) a disc or mini-cartridge file, or b) a 2645A CTU logical unit number to save it.

Both utilities are completely documented in the updated RTE Utility Reference Manual (#92060-90017) dated March '77 including examples and worksheets.

For information and ideas concerning applications of soft keys, refer to issue #12 of the Communicator (if you do not already receive the Communicator, now you know what you've been missing!).

I'm sure you'll agree, that with a little imagination and these two GREAT utilities, the possible applications and benefits of the soft key feature are endless.

So good luck, and KEEP pressing those Soft Keys!!!

Software Samantha

In this issue I will share with you a letter and program I received, as well as present information on some capabilities which are currently available under RTE but, as of yet are not documented. I would like to thank ***Donald L. Clapp*** for correcting the information given in this column in the July issue concerning system entry points. He has also included his own program LIBLS which correctly lists the system entries.

Did you realize you can retrieve the transmission log and status word from a formatted read, or parse a file manager "namr" just as FMGR does? Also, now there is a method of determining programmatically the operating system in control, RTE II, RTE III, or RTE-M.

Dear Sam,

Enclosed is a program that lists all RTE Library entries. For disc resident subroutines the type, size, amount of common used, disc location and extended name record are listed.

The information published in the July **COMMUNICATOR** was close, but No Cigar. The entry types contained in the right byte of word 3 are:

TYPE	= 0	Memory resident
	= 1	Disc Resident
	= 4	Replacement

I was not able to confirm the proper type for an absolute entry.

Sincerely,
ELI LILLY AND COMPANY
Donald L. Clapp
Lab Automation and
Process Control Services

Dear Donald,

Your findings are correct. An absolute entry is type three, type two is not currently being implemented. At present there is only one absolute library entry, $PDSK. It may be used to protect peripheral discs in the same manner as FMP files on logical unit two or three. If this option is desired, at generation $PDSK should be specified absolute and given a nonzero value. As the system is booted up the value of $PDSK is checked and, if nonzero, certain values are patched to permit checking for illegal writes on peripheral discs. Please note ***Donald's*** program LIBLS printed at the end of the column.

Sincerely,
Samantha

ISTAT AND ITLOG - STATUS WORD AND TRANSMISSION LOG

There exist two entry points in the FMTIO module of the formatter which are accessible as user-callable functions or subroutines. ISTAT and ITLOG return the transmission log and status word after formatted input. These routines will be documented in the next revision of the DOS/RTE Relocatable Library Reference manual. The calling sequence for the two routines is given below:

```
ICHRS = ITLOG(IXXX)
```

where

ICHRS = The number of characters read or written by the formatter by its last input/output request to the system. The value of ICHRS will be in the range zero to 134 (120 of binary) regardless of the specified buffer size in the read or write statement.

IXXX = The same as ICHRS

```
ISTUS = ISTAT(IXXX)
```

where

ISTUS = The status word returned from the exec in the last input/output call which used the formatter.

IXXX = The same as ISTUS

NAMR - PARSE FMGR "NAMR"

Another routine which is mentioned briefly in the Relocatable Subroutines manual under the name "NAMR" will parse a "namr" in FORTRAN or Assembly language in the same manner as the RTE II file manager FMGR. The routine reads an input buffer of any length and produces a parameter buffer ten words long. The format of the output buffer is as follows:

```
word n n= 1,10
word 1 = 0 if type is 0,
         16 bit two's complement number if type is 1,
         or characters 1 and 2 if type is 3
word 2 = 0 if type is 0 or 1, characters 3 and 4 if type is 3
word 3 = 0 if type is 0 or 1, characters 5 and 6 if type is 3
word 4 = parameter type of all seven parameters in two bit pairs
         0 = null parameter
         1 = integer numeric parameter
         2 = not implemented
         3 = left justified six ASCII character parameter
         bits for ,FNAME : P1 : P2 : P3 : P4 : P5 : P6 ,
         are        0,1   2,3  4,5  6,7  8,9 10,11 12,13
word 5 = characteristics of 1st sub-parameter defined as in word 1
word 6 = characteristics of 2nd sub-parameter defined as in word 5
word 7 = "                  " 3rd sub-parameter "                  "
word 8 = "                  " 4th sub-parameter "                  "
word 9 = "                  " 5th sub-parameter "                  "
word 10= "                  " 6th sub-parameter "                  "
```

The calling sequence for routine NAMR is seen below:

```
IF ( NAMR ( IPBUF,INBUF,LENTH,ISTRC )) 10,20
```

where the parameters are defined as follows:

IPBUF = The ten word destination parameter buffer.

INBUF = The starting address of input buffer containing "namr"to be parsed.

LENTH = The positive character length of "INBUF".

ISTRC = The starting character number in "INBUF". This parameter will be updated for possible next call to "NAMR" as the new starting character in "INBUF". Please note ISTRC is modified by this routine, thus it must be passed as a variable in the call statement and not as a constant.

The flow of control following the call is dependent upon the value returned in the A-register:

10 branch = NAMR was not passed a buffer of greater than zero length to parse, that is if ISTRC > LENTH.

20 branch = NAMR routine was passed a buffer of at least one character in length.

Consider the following example,

+12345, DOUG:DB:−12B:,,GEORGE: A,
&PARSE:JB::4:−1:1775:123456B
and the results returned from NAMR.

NAMR #	W1	W2	W3	W4	W5	W6	W7	W8	W9	W10
1	12345	0	0	00001B	0	0	0	0	0	
2	DO	UG		00037B	DB	−10	0	0	0	0
3	0	0	0	00000B	0	0	0	0	0	0
4	GE	OR	GE	00017B	A	0	0	0	0	0
5	&P	AR	SE	12517B	JB	0	4	−1	1775	−22738

$OPSY - OPERATING SYSTEM TYPE

An entry point has been added to RTE II, RTE III, and RTE-M operating systems which allows the user to determine programmatically the system in control. The entry point $OPSY has been added to the input/output control module RTIOC. In the definition of $OPSY bits 15, 0, 1, and 2 are set for specific characteristics of the system:

BIT	MEANING
15,	set to 1 for RTE, 0 for DOS.
0,	set to 1 for disc sector size of 64 words, 0 for sector size of 128 words.
1,	set to 1 for DMS.
2,	set to 1 for disc-based RTE (i.e., RTE II and III), set to 0 for memory based (i.e., RTE-M).

The value of $OPSY is as follows:

SYSTEM	$OPSY
RTE II	−3
RTE III	−1
RTE-MI	−7
RTE-MII	−15
RTE-MIII	−5
DOS-M	1

```
PROGRAM LIBLS - LIST SYSTEM ENTRY POINTS
FTN4,L
C     RTE SUBROUTINE LIBRARY LISTINGS
C     LISTS ALL ENTRY POINTS IN SYSTEM
C     VERSION #1 DLC
C
      PROGRAM LIBLS
      DIMENSION IGET(1),ID(4,16),NAM(64)
C
      IGET(0)=1
      IDSKAD=IGET(1761B)
      ITRAK=IAND(IDSKAD,77600B)/128
      ISEC=IAND(IDSKAD,177B)
      NUM=IGET(1762B)
C
10    CALL EXEC(1,2,ID,64,ITRAK,ISEC)
      DO 50 I=1,16
      ITYPE=IAND(ID(3,I),377B)+1
      GO TO (60,70,80,80,90),ITYPE
60    WRITE(6,1060) ID(1,I),ID(2,I),ID(3,I),ID(4,I)
1060  FORMAT(X,2A2,A1," MEM. RES. ",K8)
      GO TO 49
C
70    KSEC=IAND(ID(4,I),177B)
      KTRK=IAND(ID(4,I),77600B)/128
      CALL EXEC(1,2,NAM,64,KTRK,KSEC)
      JLEN=NAM(1)/256
      KTYPE=NAM(10)
      KMAIN=IAND(NAM(7),77777B)
      KCOM=NAM(9)
      IF (JLEN.LE.17) GO TO 75
      WRITE(6,1070) ID(1,I),ID(2,I),ID(3,I),KTYPE
     X       ,KMAIN,KCOM,KTRK ,KSEC,(NAM(K),K=18,JLEN)
1070  FORMAT(X,2A2,A1," DISC RES. ",5I5,4X,20A2)
      GO TO 49
C
75    WRITE(6,1070) (ID(J,I),J=1,3),KTYPE,KMAIN,KCOM,KTRK,KSEC
      GO TO 49
C
80    WRITE(6,1070) ID(1,I),ID(2,I),ID(3,I),ID(4,I)
      GO TO 49
1080  FORMAT(X,2A2,A1," ABS. ",K8)
```

```
C
90    WRITE(6,1090) ID(1,I),ID(2,I),ID(3,I),ID(4,I)
1090  FORMAT(X,2A2,A1," REPLACE ",K8)
C
49    NUM=NUM-1
      IF(NUM.LE.0) STOP
50    CONTINUE
C
      ISEC=ISEC+1
      IF(ISEC.LT.IGET(1757B)) GO TO 10
      ISEC=0
      ITRAK=ITRAK+1
      GO TO 10
      END
      END$
```

If you have any questions, suggestions, or comments about your 1000(9600) system, please address them to:

SOFTWARE SAMANTHA
c/o Communicator 1000 (9600) Group
HP Data Systems Division
11000 Wolfe Road
Cupertino, CA. 95014

NEW HIGH PERFORMANCE MEMORY MEANS INCREASED SYSTEM THROUGHPUT

Bill Elmore/DSD

Hewlett-Packard has introduced a High Performance Memory System for its 21MX E-Series Computers. Since memory reference instructions make up a large part of those executed in the computer, the new memory system is expected to increase system performance by as much as 30%.

Hewlett-Packard's E-Series Computer was designed with an eye on the future; its Variable Microinstruction Timing architecture allows higher speed memories to be used as they become available. The new memory has a cycle time of 350 nanoseconds, compared to the 560 nsec cycle time of HP's Standard Performance Memory. HP adopted the use of 16-pin high speed RAM's for use in the High Performance Memory System only after the stringent qualifying procedures applied to earlier memory products were met. All RAM's are subjected to an extensive conditioning and checkout process that we feel is the single most contributing factor to the reliability of Hewlett-Packard's memory products.

The 12741A High Performance Memory Module is a 16K word array and must be used with the 2102E Memory Controller. Where High Performance Memory is being installed in an existing E-Series Computer, a change of the memory backplane may also be necessary.

And what does this all mean to you? High Performance Memory will increase system performance and still give you the reliability you expect from HP memories.

"FRIENDLY" DOCUMENTATION FOR RTE-M

Dick Walker/DSD

Introduction of RTE-M to the HP 1000 series of operating systems was accompanied by a coordinated effort to make it's documentation easy to use by newer HP customers. Changes and enhancements incorporated into the RTE-M manuals include:

- A chapter of program development examples in the *RTE-M Programmer's Reference Manual* to aid users through some of their initial programming projects.
- Expanded glossaries at the front of the *RTE-M Programmer's Reference Manual* and *System Generation Manual* to provide quick summary definitions of terms that may still be unfamiliar.
- More "cookbook" procedures when a topic lends itself to a step-by-step format.
- Improved indexing scheme for all RTE-M manuals. Faster access to main information sources is provided by printing a principal page reference to a topic in *boldface* type, to distinguish it from the page numbers of secondary references to the same topic.
- Simplified generation worksheet forms in the *RTE-M System Generation Reference Manual*. Forms are heavily annotated with comments to guide the user in filling in the correct data.

By the way, look for this "friendliness" in all future software documentation. We'll be trying hard to make it equally friendly.

A potent resource, available to all HP 1000 customers, for getting improved documentation, is the Reader Comment form at the back of every manual. Reader Comment forms mailed to DSD are always carefully studied and evaluated, and become a permanent part of a manual's Master File until a revision or update package is produced.

Your feedback as a reader is important. You are listened to. Constructive criticisms are as welcome as expressions of reader satisfaction.

NEW CONTRIBUTED LIBRARY CATALOG READY

A new revision of the Data Systems Division Contributed Library Program Catalog has just been released. It contains over 600 program descriptions, prices and ordering information. This invaluable volume represents literally thousands of man hours and reflects the professional efforts of HP customers in all fields of interest.

Programs listed are selected from customer contributions that have been documented according to directions supplied in the catalog and deemed to be of most value to the broadest spectrum of users. The LOCUS catalog as it is currently called (part number 22000-90099) is available for $15.00 from Hewlett-Packard Company, Contributed Software, P.O. Box 61809, Sunnyvale, California 94088, please use the special order form supplied in back of this issue for ordering the LOCUS catalog. Foreign customers please contact your local HP sales office.

NEW RELEASES FOR DATA SYSTEMS LOCUS:

22681-18953 HP 59310B INTERFACE BUS DRIVER

This package consists of a DOS driver designed to provide control of the HP Interface Bus Card (59310A) and a set of eleven Algo-callable interface bus utility subroutines which provides a user interface for the control of the bus card. Both the driver and the subroutines, enables communication with devices attached to the bus. One version of the driver allows data transfer through the Direct Memory Access (DMA) channel. The utility subroutines require the use of the HP 59310A Interface Bus driver. They also assume the inclusion of .SIN., .ERR., .DIO, .IOR., and .PRAM. subroutines in the disc library.

22681-18953	PT	$30.00

22682-18945 UNEXT

This program searches each mounted disc for files with extents, Type 3 or greater, and restores them without extents. It will also, optionally reclaim from Type 4 files only, any unused blocks from the file.

22682-18945	PT	$10.00
22682-13345	Cass	$35.00

22682-18946 21MX MICROCODED FFT WITH AUTOMATIC SCALING ON OVERFLOW

This subroutine performs a microcode enhanced FFT (Fast Fourier Transform) with automatic scaling on overflow. Data to be transformed must be presented as a single integer array. The data may be complex or real. If complex, the odd subscripted elements are the "reals", and the even subscripted elements are the "imaginaries". If real, the "reals" are contiguous. A complex transform of size N requires 2*N memory locations. A real transform of size N requires N memory locations. Execution time for 1024 complex points is about 300 ms (milliseconds). Execution time for 1024 complex (all imaginaries = 0) points or 1024 real points is about 160 ms. Each overflow increases execution time by 6 ms for complex transforms; each overflow increases execution time by 3 ms for complex (all imaginaries = 0) or real transforms. The assembly language portion of the subroutine requires 269 memory locations. The microcoded portion requires 512 words of control store.

22682-18946	PT	$40.00
22682-13346	Cass	$70.00

22682-18947 (STRAK) TEMPORARY, EXPANDABLE FIXED LENGTH RECORD DISC FILE

STRAK provides the RTE user with a temporary file which is automatically expandable. The file is written on the system tracks and operates like a random access type 2 file manager file. It supports fixed length records with the length specified in a setup call. The program is useful for temporary storage of data when the quantity of data cannot be determined beforehand.

22682-18947	PT	$10.00
22682-13347	Cass	$35.00

22682-18948 FORMATTED READ/WRITE TO DISC FILES USING FORTRAN READ/WRITE STATEMENTS

This program is similar to the 'CODE' program in the FORTRAN Library. When a CALL DISK (FNAME,ISTAT) is made before the formatted read or write statement, the date is transferred to/from the specified file. This gives the ability to format data used on the disc. (ASCII source files may be either written or read). The CALL DISK and the READ/WRITE following are treated as if they were one statement. If the READ/WRITE is executed without executing the CALL DISK, then it functions normally. When executing the CALL DISK, the unit number is ignored. The transfer is directed towards the file named in FNAME. 'DISK' is intended for use with only ASCII records.

22682-18948	PT	$20.00

22682-18949 CLASS - INTERROGATE CLASS I/O SYSTEM

This program allows the RTE-II or RTE-III user to completely interrogate the class I/O system. It is capable of displaying the entire class table status, listing the contents of the class table, and clearing out pending class buffers on a particular class number. The utility is completely conversational with error messages.

22682-18949	PT	$10.00
22682-13349	Cass	$35.00

22682-18950 TRACK

This program gives a complete breakdown in tabular form of the RTE system and auxiliary discs as to ownership. The program lists swap tracks for programs being swapped and the system library and entry point table.

22682-18950	PT	$10.00
22682-13350	Cass	$35.00

22682-18951 FMP-UTIL

These three programs allow the user to restore, save or retrieve File Manager Files on 7970 or 3030 mag tape transports and a 7900 or 7905 disc. All utilities are completely conversational and self configuring with elaborate error checking. Checks are made to ensure File Manager directory capability.

22682-18951	PT	$10.00
22682-13351	Cass	$35.00

22682-18952 MINV — INVERSE ASSEMBLER FOR 2100 MICROCODE

"MINV", is an inverse assembler for 2100 microcode. It will produce a listing from a binary tape generated by the Micro Assembler. 'MINV', will also produce source code that can be directly incorporated in assembly language programs. It generates 'OCT' statements with the proper micro instruction codes and puts the corresponding inverse assembly in the comments field. This can be put into a program so that the program can write it to WCS, making the program more self contained.

22682-18952	PT	$10.00
22682-13352	Cass	$35.00

22682-18953 RTDMO

This program illustrates how to utilize the File Manager Call and program the 2640 series terminals. Since the nature of the program lends itself toward a general ASCII message processor (page oriented), it can be used to implement and display any desired data case with limits tailored to the 2640 series screen for 23 lines with 132 characters per line. The cursor is reserved for line 24.

22682-18953	PT	$10.00
22682-13353	Cass	$35.00

22682-18954 LISTF — SELECTIVE FMGR DIRECTORY LISTER

'LISTF', produces a partial directory list of FMGR directory(s) based on user specified criteria. Any combination of the following criteria may be specified by the user:

1. Filename, or more generally, the presence of given characters in any of the 6 filename character positions.
2. Filetype.
3. Cartridge reference number
4. Extent number

A file must satisfy all the criteria given by the user in order to be included in the output list. Omitted criteria default to 'ALL'. Output maybe directed to a device of the user's choosing. Alternatively, output maybe directed to a file (named listXX) in procedure file format for subsequent processing.

22682-18954	PT	$10.00
22682-13354	Cass	$35.00

22682-18955 EQTXX

'EQTXX', is used to access EQT entries. Specified words of an EQT entry can be read or written to dynamically change aspects of a device under program control. EQT entries can be accessed by EQT number or LU number.

22682-18955	PT	$10.00
22682-13355	Cass	$35.00

22682-18956 21MX MICROCODED .PACK AND .FLUN

These two routines are direct replacements for the standard library subroutines. A total of 17 words of microcode replaces more than 100 words of Assembly Language. Hardware requirements include a 21MX CPU and User Control Store.

22682-18956	PT	$10.00
22682-13356	Cass	$35.00

22682-18957 OVER

This program allows the RTE-II/III user to input absolute records from any device into corresponding memory. The program is completely conversational and is primarily designed for on-line driver development. The program requires 5K of memory.

22682-18957	PT	$10.00
22682-13357	Cass	$35.00

22682-18958 DFINE — REDEFINE PARTITIONS ON-LINE

This program allows users to redefine partitions in RTE-III while on-line. Changes may be memory only, disc only, or memory and disc. The system may be active while partitions are being redefined. There are minimal cautions (input errors which can not be detected): these are explained in the documentation. Pages with parity errors may be omitted from the redefined partitions. Partition definition follows the same procedure as the generator itself (except the generator does not exclude pages with parity errors). It starts by printing the page requirements of real-time and background programs and any partition assignments. Memory size is requested: the response may be a memory size less than or greater than specified at generation. The ease of redefining partitions permits the user to experiment to find the optimum for his application and to make changes to accomodate occasional unusual needs. For example, a partition equal to the maximum addressable (say 14 pages) may be needed for a program that is run perhaps once a month. The rest of the time, a 14 page partition would be wasteful of memory. Some of the questions a user might have are:

1. How many partitions do I need?
2. How many should be real-time and how many should be background?
3. Should some of my partitions be reserved for special programs? Which partitions and which programs?

DFINE makes it easy to experiment with these parameters and change the partitions without regeneration. This places less of a burden upon the user to thoroughly analyze his needs prior to doing the generation.

22682-18958	PT	$ 50.00
22682-13358	Cass	$105.00

22682-18959 DOS-M DUMP — DISC DUMP FILE UTILITY

'DUMP' is an interactive program which will allow the DOS user to conveniently examine the contents of any user disc files. The user specifies the name of the file to de dumped, the sector with which to begin the dump, the number of consecutive sectors to be dumped, and the output format. Currently output is available in any combination of the following formats:

ASCII, OCTAL HALF-WORD, INTEGER, FLOATING POINT

The user is prompted for all inputs, and receives immediate help to correct input errors. Output is in the form of a page numbered, labeled listing of the selected file in the selected formats.

22682-18959	PT	$10.00

22682-18960 NAMR

'NAMR' is an RTE-II or RTE-III subroutine that interfaces to system subroutine $PARS to break out the individual fields of a standard RTE file name. The passed buffer initially contains NAME:SC:CRN and NAMR returns the name as an ASCII string and the security code and cartridge number as binary integers. NAMR can be passed a buffer or it will input the buffer from a specified logical unit, using system subroutine REIO.

22682-18960	PT	$10.00
22682-13360	Cass	$35.00

22682-18961 AUTOMATIC TEXT REPORT FORMATTER

This program allows the RTE-II/III or M, user to produce a formatted listing of an FMP file with automatic page numbering and titles. The program accepts a file name (NAMR) and prompts the user for print control information to produce the listing. For editing purposes, there is an option to produce line numbers for each line of text. The program has extensive error checking, can be run under batch or spooling, and text input can be from external devices through FMP type 0 files.

22682-18961	PT	$10.00
22682-13361	Cass	$35.00

22682-10962 INTEL 8080 ABSOLUTE PROGRAM ASSEMBLER

M8080 assembles absolute programs written in the Assembly described in the "INTEL 8080 Assembly Language Programming Manual", with the following exceptions:

1. Macros are not supported, because more powerful macro facilities are already available.
2. Only the operands + and – are allowed within expressions.
3. Statements labels are limited to 5 characters.

22682-10962	800 BPI MT	$40.00
22682-11962	1600 BPI MT	$40.00
22686-13362	Cass	$70.00

SOFTWARE UPDATES

Following are cross-reference lists of the available 92001B, 92060B, 92062A, and 92064A (options 20 & 40) software modules, the media on which the software modules are distributed, and the date code or revision of each module up to, and including level 1710. Software modules updated since the last issue are indicated for each reference.

NOTE:

For each module, interdependencies with other modules may exist (i.e., any updated module may require other updated modules to function properly).

SOFTWARE MODULE NUMBERS: 92001B LEVEL 1710 (RTE II)

The following modules are also available on a 7900 RTE Master Software Disc (#92001-13001), or a 7905 RTE Master Software Disc (#92001-13101).

Module (Paper Tape) Number	Module File Name	Module Descriptor	Part Number Mini-Cartridge	Date Code or Revision
02607-16004	!S4L07	24K SIO LINE PRINTER DRIVER	92001-13305	1538
09601-16021	%DVR15	RTE 7261A DRIVER	92062-13301	A
12732-16001	%DVR33	FLEXIBLE DISC DRIVER	92062-13304	1650
12970-16004	!S4MT1	24K SIO MAG. TAPE DRIVER	92001-13305	1550
20747-60001	%DVR30	RTE FIXED HEAD DISC DRIVER	92062-13301	C
20808-60001	%CAL10	CAL. PLOTTER DRIVER	92062-13301	B
20810-60001	%CALIB	CAL. PLOTTER LIBRARY	92062-13301	C
20875-60001	%1FTN	FORTRAN MAIN CONTROL	92060-13308	E
20875-60002	%2FTN	FORTRAN PASS 1	92060-13308	E
20875-60003	%3FTN	FORTRAN PASS 2	92060-13308	E
20875-60004	%4FTN	FORTRAN PASS 3	92060-13308	E
20875-60005	%5FTN	FORTRAN PASS 4	92060-13308	E
24129-60001	%ALGOL	RTE/DOS ALGOL PART 1	92060-13305	1643
24129-60002	%ALGL1	RTE/DOS ALGOL PART 2	92060-13305	C
24153-60001	%FF.N	RTE/DOS FORMATTER	92060-13303	C
24170-60001	%1FTN4	RTE/DOS FORTRAN IV PART 1	92060-13306	C
24170-60002	%2FTN4	RTE/DOS FORTRAN IV PART 2	92060-13306	C
24170-60003	%3FTN4	RTE/DOS FORTRAN IV PART 3	92060-13306	C
24177-60001	%1FFT4	RTE/DOS FAST FORTRAN IV PART 1	92060-13307	1442
24177-60002	%2FFT4	RTE/DOS FAST FORTRAN IV PART 2	92060-13307	1442
24306-60001	%DECAR	DEC. STRING ARITH LIBR.	92060-13303	A
24998-16001	%RLIB1	RTE/DOS LIBRARY PART 1	92060-13302	1624
24998-16001	%RLIB2	RTE/DOS LIBRARY PART 2	92060-13302	1624
24998-16002	%FF4.N	FORTRAN IV FORMATTER	92060-13303	1624
25117-60499	%DVR24	RTE 7970 7T MAG. TAPE DRIVER	92062-13301	D
29013-60001	%DVR31	RTE 7900A DISC DRIVER	92062-13301	1710
29028-60002	%DVR12	RTE 2767A DRIVER	92062-13301	A
29029-60001	%DVR00	RTE TTY/PUNCH/PHOTO READER	92062-13301	1642
29030-60001	%DVR11	RTE 2892A CARD READER DRIVER	92062-13301	1710
29100-60017	!S4LP	24K SIO LINE PRINTER	92001-13305	A
29100-60018	!S4SYD	24K SIO SYSTEM DUMP	92001-13305	A
29100-60019	!S4PHR	24K SIO PHOTO READER	92001-13305	A
29100-60020	!S4PUN	24K SIO TAPE PUNCH	92001-13305	A
29100-60022	!S4L67	24K SIO 2767 LINE PRINTER	92001-13305	A
29100-60023	!S4MT2	24K SIO 7970 MAG. TAPE	92001-13305	A
29100-60049	!S4MT3	24K SIO MAG. TAPE	92001-13305	A
29100-60050	!S4TER	24K SIO TERMINAL PRINTER	92001-13305	A
59310-16002	%1DV37	RTE HP-IB WITHOUT SRQ	92062-13301	1710
59310-16003	%2DV37	RTE HP-IB WITH SRQ	92062-13301	1710

Module (Paper Tape) Number	Module File Name	Module Descriptor	Part Number Mini-Cartridge	Date Code or Revision
59310-16004	%HPIB	HP-IB DEVICE SUBROUTINE	92062-13301	1710
59310-16005	%SRQ.P	SRQ TRAP UTILITY	92062-13304	1710
72008-60001	%1DV10	COMP. 7210A PLOTTER DRIVER	92062-13301	A
72009-60001	%2DV10	MIN. 7210A PLOTTER DRIVER	92062-13301	A
91200-16001	%DVA13	91200A DRIVER	92062-13301	1648
91200-16002	%TVLIB	91200A VIDEO MONITOR LIBRARY	92062-13301	1648
91200-16004	%TVVER	91200A TV INTERFACE VERIFIER	92062-13301	1648
92001-16002	%LDR2	RTE LOADER	92001-13301	1640
92001-16003	%MTM	MULT. TERMINAL MONITOR	92001-13301	B
92001-16004	%2DP43	POWER FAILURE DRIVER	92001-13301	1633
92001-16005	%SYLIB	RTE SYSTEM LIBRARY	92001-13301	1710
92001-16012	%CR2SY	CORE RESIDENT OPERATING SYS.	92001-13301	1710
92001-16013	!2GN00	RTE-II 7900 OFF-LINE GEN.	92001-13303	1631
92001-16014	%AUTOR	AUTO RESTART PROGRAM	92001-13302	1631
92001-16018	!2GNFH	RTE-II FIXED HEAD DISC GEN.	92001-13306	1631
92001-16020	%DVA12	2607/10/13/14/17/18 DRIVER	92062-13301	1534
92001-16026	!2GN05	RTE-II 7905 OFF-LINE GEN.	92001-13303	1631
92001-16027	%4DV05	RTE 2640A/2644A DRIVER	92062-13301	1650
92001-16028	%0DV05	RTE 2640A DRIVER	92062-13301	1650
92001-16029	%$CMD2	RTE-II COMMAND PROGRAM	92001-13301	1710
92001-16030	%WHZT2	RTE-II WHZAT PROGRAM	92001-13302	1631
92001-16031	%RT2G1	RTE-II ON-LINE GENERATOR PART 1	92001-13304	1704
92001-16031	%RT2G2	RTE-II ON-LINE GENERATOR PART 2	92001-13304	1704
92001-18014	&AUTOR	AUTOR RESTART SOURCE	92001-13302	1631
92002-12001	%BMPG1	BATCH MONITOR PROGRAM PART 1	92002-13301	1631
92002-12001	%BMPG2	BATCH MONITOR PROGRAM PART 2	92002-13301	1631
92002-12001	%BMPG3	BATCH MONITOR PROGRAM PART 3	92002-13301	1631
92002-12002	%2SPO1	RTE-II SPOOL MONITOR PART 1	92002-13303	1631
92002-12002	%2SPO2	RTE-II SPOOL MONITOR PART 2	92002-13303	1631
92002-16006	%BMLIB	BATCH LIBRARY	92002-13302	1631
92002-16010	%EDITR	RTE EDITOR	92002-13302	C
92060-12004	%ASMB	RTE ASSEMBLER	92060-13304	1639
92060-16028	%XREF	CROSS REFERENCE	92060-13304	A
92060-16031	%DVR32	RTE 7905A DISC DRIVER	92062-13301	A
92060-16038	%SWTCH	RTE-II SWITCH PROGRAM	92001-13304	1710
92060-16039	%SAVE	SAVE PROGRAM	92060-13309	1704
92060-16040	%RESTR	RESTORE PROGRAM	92060-13309	1704
92060-16041	%VERFY	DISC VERIFY PROGRAM	92060-13309	1704
92060-16042	%COPY	DISC COPY PROGRAM	92060-13309	1704
92060-16043	%DBKLB	DISK BACK UP LIBRARY	92060-13309	1704
92060-16044	!DSKUP	OFF-LINE DISK BACK UP	92060-13309	1704
92060-16045	%RDNAM	READ NAMR PROGRAM	92001-13302	1631
92060-16052	%KEYS	SOFT KEY UTILITY	92001-13302	1707
92060-16053	%KYDMP	SOFT KEY DUMP UTILITY	92001-13302	1707
92060-18046	&UPDAT	UPDATE TRANSFER FILE	92001-13302	1631
92060-18047	&PKDIS	PACK DISC TRANSFER FILE	92001-13302	1631
92202-16001	%DVR23	RTE 7970 9T MAG. TAPE DRIVER	92062-13301	A
92900-16002	%2DV47	RTE 92900A DRIVER W/OUT DMS	92062-13302	1643

SOFTWARE MODULE NUMBERS: 92060B LEVEL 1710 (RTE III)

The following modules are also available on a 7900 RTE Master Software Disc (#92060-13001), or a 7905 RTE Master Software Disc (#92060-13101).

Module (Paper Tape) Number	Module File Name	Module Descriptor	Part Number Mini-Cartridge	Date Code or Revision
02607-16004	!S4L07	24K SIO LINE PRINTER DRIVER	92001-13305	1538
09601-16021	%DVR15	RTE 7261A DRIVER	92062-13301	A
12732-16001	%DVR33	FLEXIBLE DISC DRIVER	92062-13304	1650
12970-16004	!S4MT1	24K SIO MAG. TAPE DRIVER	92001-13305	1550
20747-60001	%DVR30	RTE FIXED HEAD DISC DRIVER	92062-13301	C
20808-60001	%CAL10	CAL. PLOTTER DRIVER	92062-13301	B
20810-60001	%CALIB	CAL. PLOTTER LIBRARY	92062-13301	C
20875-60001	%1FTN	FORTRAN MAIN CONTROL	92060-13308	E
20875-60002	%2FTN	FORTRAN PASS 1	92060-13308	E
20875-60003	%3FTN	FORTRAN PASS 2	92060-13308	E
20875-60004	%4FTN	FORTRAN PASS 3	92060-13308	E
20875-60005	%5FTN	FORTRAN PASS 4	92060-13308	E
24129-60001	%ALGOL	RTE/DOS ALGOL PART 1	92060-13305	1643
24129-60002	%ALGL1	RTE/DOS ALGOL PART 2	92060-13305	C
24153-60001	%FF.N	RTE/DOS FORMATTER	92060-13303	C
24170-60001	%1FTN4	RTE/DOS FORTRAN IV PART 1	92060-13306	C
24170-60002	%2FTN4	RTE/DOS FORTRAN IV PART 2	92060-13306	C
24170-60003	%3FTN4	RTE/DOS FORTRAN IV PART 3	92060-13306	C
24177-60001	%1FFT4	RTE/DOS FAST FORTRAN IV PART 1	92060-13307	1442
24177-60002	%2FFT4	RTE/DOS FAST FORTRAN IV PART 2	92060-13307	1442
24306-60001	%DECAR	DEC. STRING ARITH LIBR.	92060-13303	A
24998-16001	%RLIB1	RTE/DOS LIBRARY PART 1	92060-13302	1624
24998-16001	%RLIB2	RTE/DOS LIBRARY PART 2	92060-13302	1624
24998-16002	%FF4.N	FORTRAN IV FORMATTER	92060-13303	1624
25117-60499	%DVR24	RTE 7970 7T MAG. TAPE DRIVER	92062-13301	D
29013-60001	%DVR31	RTE 7900A DISC DRIVER	92062-13301	1710
29028-60002	%DVR12	RTE 2767A DRIVER	92062-13301	A
29029-60001	%DVR00	RTE TTY/PUNCH/PHOTO READER	92062-13301	1642
29030-60001	%DVR11	RTE 2892A CARD READER DRIVER	92062-13301	1710
29100-60017	!S4LP	24K SIO LINE PRINTER	92001-13305	A
29100-60018	!S4SYD	24K SIO SYSTEM DUMP	92001-13305	A
29100-60019	!S4PHR	24K SIO PHOTO READER	92001-13305	A
29100-60020	!S4PUN	24K SIO TAPE PUNCH	92001-13305	A
29100-60022	!S4L67	24K SIO 2767 LINE PRINTER	92001-13305	A
29100-60023	!S4MT2	24K SIO 7970 MAG. TAPE	92001-13305	A
29100-60049	!S4MT3	24K SIO MAG. TAPE	92001-13305	A
29100-60050	!S4TER	24K SIO TERMINAL PRINTER	92001-13305	A
59310-16002	%1DV37	RTE HP-IB WITHOUT SRQ	92062-13301	1710
59310-16003	%2DV37	RTE HP-IB WITH SRQ	92062-13301	1710
59310-16004	%HPIB	HP-IB DEVICE SUBROUTINE	92062-13301	1710
59310-16005	%SRQ.P	SRQ TRAP UTILITY	92062-13304	1710
72008-60001	%1DV10	COMP. 7210A PLOTTER DRIVER	92062-13301	A
72009-60001	%2DV10	MIN. 7210A PLOTTER DRIVER	92062-13301	A
91200-16001	%DVA13	91200A DRIVER	92062-13301	1648
91200-16002	%TVLIB	91200A VIDEO MONITOR LIBRARY	92062-13301	1648
91200-16004	%TVVER	91200A TV INTERFACE VERIFIER	92062-13301	1648
92001-16003	%MTM	MULT. TERMINAL MONITOR	92060-13301	B
92001-16005	%SYLIB	RTE SYSTEM LIBRARY	92060-13301	1710
92001-16014	%AUTOR	AUTO RESTART PROGRAM	92060-13310	1631
92001-16020	%DVA12	2607/10/13/14/17/18 DRIVER	92062-13301	1534
92001-16027	%4DV05	RTE 2640A/2644A DRIVER	92062-13301	1650

Module (Paper Tape) Number	Module File Name	Module Descriptor	Part Number Mini-Cartridge	Date Code or Revision
92001-16028	%0DV05	RTE 2640A DRIVER	92062-13301	1650
92001-18014	&AUTOR	AUTO RESTART PROGRAM SOURCE	92060-13310	1631
92002-12001	%BMPG1	BATCH MONITOR PROGRAM PART 1	92002-13301	1631
92002-12001	%BMPG2	BATCH MONITOR PROGRAM PART 2	92002-13301	1631
92002-12001	%BMPG3	BATCH MONITOR PROGRAM PART 3	92002-13301	1631
92002-16006	%BMLIB	BATCH LIBRARY	92002-13302	1631
92002-16010	%EDITR	RTE EDITOR	92002-13302	C
92060-12001	%3SP01	RTE-III SPOOL MONITOR PART 1	92060-13313	1631
92060-12001	%3SPO2	RTE-III SPOOL MONITOR PART 2	92060-13313	1631
92060-12003	%CR3SY	MEMORY RESIDENT SYSTEM	92060-13301	1710
92060-12004	%ASMB	RTE ASSEMBLER	92060-13304	1639
92060-16001	%3DP43	POWER FAILURE DRIVER	92060-13301	1633
92060-16004	%LDR3	RTE-iii LOADER	92060-13301	1640
92060-16006	%WHZT3	RTE-III WHZAT PROGRAM	92060-13310	1631
92060-16028	%XREF	CROSS REFERENCE	92060-13304	A
92060-16029	!3GN00	7900 RTE-III GENERATOR	92060-13311	1631
92060-16031	%DVR32	RTE 7905A DISC DRIVER	92062-13301	A
92060-16032	!3GN05	7905 RTE-III GENERATOR	92060-13311	1631
92060-16035	%$PVMP	$PVMP	92060-13301	A
92060-16036	%$CMD3	RTE-III COMMAND PROGRAM	92060-13301	1710
92060-16037	%RT3G1	RTE-III ON-LINE GENERATOR PART 1	92060-13312	1704
92060-16037	%RT3G2	RTE-III ON-LINE GENERATOR PART 2	92060-13312	1704
92060-16038	%SWTCH	RTE-III SWITCH PROGRAM	92060-13312	1710
92060-16039	%SAVE	SAVE PROGRAM	92060-13309	1704
92060-16040	%RESTR	RESTORE PROGRAM	92060-13309	1704
92060-16041	%VERFY	DISC VERIFY PROGRAM	92060-13309	1704
92060-16042	%COPY	DISC COPY PROGRAM	92060-13309	1704
92060-16043	%DBKLB	DISK BACK UP LIBRARY	92060-13309	1704
92060-16044	!DSKUP	OFF LINE DISK BACK UP	92060-13309	1704
92060-16045	%RDNAM	READ NAMR PROGRAM	92060-13310	1631
92060-16052	%KEYS	SOFT KEY UTILITY	92060-13310	1707
92060-16053	%KYDMP	SOFT KEY DUMP UTILITY	92060-13310	1707
92060-18046	&UPDAT	UPDATE TRANSFER FILE	92060-13310	1631
92060-18047	&PKDIS	PACK DISK TRANSFER FILE	92060-13310	1631
92202-16001	%DVR23	RTE 7470 9T MAG. TAPE DRIVER	92062-13301	A
92900-16003	%3DV47	RTE 92900A DRIVER WITH DMS	92062-13302	1643

SOFTWARE MODULE NUMBERS: 92062A LEVEL 1710 (RTE DRIVERS)

Module (Paper Tape) Number	Module File Name	Module Descriptor	Part Number Mini-Cartridge	Date Code or Revision
09601-16021	%DVR15	RTE 7261A DRIVER	92062-13301	A
12732-16001	%DVR33	FLEXIBLE DISC DRIVER	92062-13304	1650
20747-60001	%DVR30	RTE FIXED HEAD DISC DRIVER	92062-13301	C
20808-60001	%CAL10	CAL. PLOTTER DRIVER	92062-13301	B
20810-60001	%CALIB	CAL. PLOTTER LIBRARY	92062-13301	C
25117-60499	%DVR24	RTE 7970 7T MAG. TAPE DRIVER	92062-13301	D
29013-60001	%DVR31	RTE 7900A DISC DRIVER	92062-13301	1710
29028-60002	%DVR12	RTE 2767A DRIVER	92062-13301	A
29029-60001	%DVR00	RTE TTY/PUNCH/PHOTO READER	92062-13301	1642
29030-60001	%DVR11	RTE 2892A CARD READER DRIVER	92062-13301	1710
59310-16002	%1DV37	RTE HP-IB WITHOUT SRQ	92062-13301	1710
59310-16003	%2DV37	RTE HP-IB WITH SRQ	92062-13301	1710
59310-16004	%HPIB	HP-IB DEVICE SUBROUTINE	92062-13301	1710
59310-16005	%SRQ.P	SRQ TRAP UTILITY	92062-13304	1710
72008-60001	%1DV10	COMP. 7210A PLOTTER DRIVER	92062-13301	A
72009-60001	%2DV10	MIN. 7210A PLOTTER DRIVER	92062-13301	A
91200-16001	%DVA13	91200A DRIVER	92062-13301	1648
91200-16002	%TVLIB	91200A VIDEO MONITOR LIBRARY	92062-13301	1648
91200-16004	%TVVER	91200A TV INTERFACE VERIFIER	92062-13301	1648
92001-16020	%DVA12	2607/10/13/14/17/18 DRIVER	92062-13301	1534
92001-16027	%4DV05	RTE 2640A/2644A DRIVER	92062-13301	1650
92001-16028	%0DV05	RTE 2640A DRIVER	92062-13301	1650
92060-16031	%DVR32	RTE 7905A DISC DRIVER	92062-13301	1704
92202-16001	%DVR23	RTE 7470 9T MAG. TAPE DRIVER	92062-13301	A
92900-16002	%2DV47	RTE 92900A DRIVER W/OUT DMS	92062-13302	1643
92900-16003	%3DV47	RTE 92900A DRIVER WITH DMS	92062-13302	1643

SOFTWARE MODULE NUMBERS: 92064A OPTIONS 20 & 40 LEVEL 1711 (RTE-M)

The following modules are unique in that they are available on Flexible disc as well as Paper Tape and Mini-Cartridge.

STRUCTURE

The RTE-M operating system is divided into three groups. Refer to the RTE-M Programmer's Reference Manual (part no. 92064-90002) for a description of the operating systems.

Within this list the modules that correspond with each operating system are described as MI, MII, or MIII.

CARTRIDGE TAPES

There are three cartridge tapes that contain the three operating systems. The part numbers of these cartridge tapes and the corresponding operating systems follow:

92064-13301	RTE-MI
92064-13302	RTE-MII
92064-13303	RTE-MIII

Modules that correspond with two or all three operating systems and are contained on more than one cartridge tape contain (MI), (MII), or (MIII) in their description.

Modules that do not directly relate to the operating systems are contained on the other cartridge tapes.

FLEXIBLE DISCS

There are two flexible discs referred to as GEN DISC and APP DISC. The GEN DISC (92064-13401) contains all the software that can be loaded at generation. The APP DISC (92064-13402) contains all the application software that can be loaded on-line. As with the cartridge tapes, some of the modules can be found on both flexible discs.

The Generation disc contains the following:

- Off-line generator
- All operating system software
- I/O drivers
- Certain HP user programs

The Applications disc contains the following:

- HP applications programs — Assembler
  FORTRAN compiler
  Editor
  Cross reference
  program
- Certain relocatable system software
- Certain user programs

Modules that appear on both flexible discs contain (GEN DISC) or (APP DISC) in their description.

Module (Paper Tape)	Module File Name	Module Descriptor	Mini-Cartridge	Flexible Disc	Date Code or Revision
09601-16021	%DVR15	RTE 7261A CARD READER DRIVER	92062-13304	92064-13401	A
12732-16001	%DVR33	FLEXIBLE DISC DRIVER	92062-13304	92064-13401	1650
20808-60001	%CAL10	RTE PLOTTER DRIVER	92062-13302	92064-13401	B
20810-60001	%CALIB	CAL. PLOTTER LIBRARY	92062-13302	92064-13401	C
24153-60001	%FF.N	RTE/DOS FORTRAN FORMATTER	92060-13303	92064-13402	C
24153-60001	%FF.N	RTE/DOS FORTRAN FORMATTER	92060-13303	92064-13401	C
24306-60001	%DECAR	DOSM STRING ARITH PK	92060-13303	92064-13401	A
24998-16001	%RLIB1	RTE/DOS LIBRARY	92060-13302	92064-13401	1624
24998-16001	%RLIB1	RTE/DOS LIBRARY	92060-13302	92064-13402	1624
24998-16001	%RLIB2	RTE/DOS LIBRARY	92060-13302	92064-13401	1624
24998-16001	%RLIB2	RTE/DOS LIBRARY	92060-13302	92064-13402	1624
24998-16002	%FF4.N	FORTRAN IV FORMATTER	92060-13303	92064-13402	1624
24998-16002	%FF4.N	FORTRAN IV FORMATTER	92060-13303	91064-13401	1624
29028-60002	%DVR12	RTE 2767A DRIVER	92062-13303	92064-13401	A
29029-60001	%DVR00	RTE TTY/PUNCH/PHOTO READER	92062-13302	92064-13401	1642
29030-60001	%DVR11	RTE 2892A CARD READER DRIVER	92062-13303	92064-13401	1710
59310-16002	%1DV37	HP-IB WITHOUT SYSTEM REQUEST	92062-13304	92064-13401	1710
59310-16003	%2DV37	HP-IB WITH SYSTEM REQUEST	92062-13304	92064-13401	1710
59310-16004	%HPIB	HP-IB RTE UTILITY	92062-13304	92064-13401	1710
59310-16005	%SRQ.P	SRQ.P TRAP UTILITY	92062-13304	92064-13401	1710
72008-60001	%1DV10	COMP. 7210A PLOTTER DRIVER	92062-13302	92064-13401	A
72009-60001	%2DV10	MIN. COMP. 7210A PLOTTER DRIVE	92062-13302	92064-13401	A
91200-16001	%DVA13	91200 TV INTERFACE DRIVER	92062-13303	92064-13401	1648
91200-16002	%TVLIB	VIDEO MONITOR LIBRARY	92062-13303	92064-13401	1648
91200-16004	%TVVER	TV INFT VERIF	92062-13303	92064-13401	1648
92001-16020	%DVA12	2607/10/13/14/17/18 DRIVER	92062-13303	92064-13401	1534
92001-16027	%4DV05	RTE 2644/45 DRIVER	92062-13302	92064-13401	1650
92001-16028	%0DV05	RTE 2640A DRIVER	92062-13302	92064-13401	1650
92060-16052	%KEYS	SOFT KEY UTILITY	92064-13304	92064-13402	1707
92060-16053	%KYDMP	SOFT KEY DUMP UTILITY	92064-13304	92064-13402	1707
92064-12005	%FMPC	CARTRIDGE FMP/FMPCR (LIB)	92064-13306	92064-13401	1709
92064-12006	%FMPF	FLEX DISC FMGR LIB (GEN DISC)		92064-13401	1709
92064-12006	%FMPF	FLEX DISC FMGR LIB (APP DISC)		92064-13402	1709
92064-16001	%MSY1	MI OPERATING SYSTEM	92064-13301	92064-13401	1709
92064-16002	%MSY2	MII OPERATING SYSTEM	92064-13302	92064-13401	1709
92064-16003	%MSY3	MIII OPERATING SYSTEM	92064-13303	92064-13401	1709
92064-16005	%MBU	MI BUFFERING	92064-13301	92064-13401	1650
92064-16006	%MMP	MI SCHEDULING OPTION	92064-13301	92064-13401	1650
92064-16008	%MTI	TIMER OPTION (MIII)	92064-13303	92064-13401	1650
92064-16008	%MTI	TIMER OPTION (MII)	92064-13302	92064-13401	1650
92064-16008	%MTI	TIMER OPTION (MI)	92064-13301	92064-13401	1650
92064-16009	%MTS	TIME SCHEDULING OPTION (MIII)	92064-13303	92064-13401	1650
92064-16009	%MTS	TIME SCHEDULING OPTION (MII)	92064-13302	92064-13401	1650
92064-16009	%MTS	TIME SCHEDULING OPTION (MI)	92064-13301	92064-13401	1650
92064-16010	%MOP	OPERATOR COMMAND OPTION (MI)	92064-13301	92064-13401	1650
92064-16010	%MOP	OPERATOR COMMAND OPTION (MII)	92064-13302	92064-13401	1650
92064-16010	%MOP	OPERATOR COMMAND OPTION (MIII)	92064-13303	92064-13401	1650
92064-16011	%MCL	CLASS I/O OPTION (MII)	92064-13302	92064-13401	1650
92064-16012	%MAP	MI/II ABSOLUTE PROGRAM LOADER	92064-13305	92064-13401	1650
92064-16013	%MDMLB	DUMMY LIBRARY (MI)	92064-13301	92064-13401	1650
92064-16013	%MDMLB	DUMMY LIBRARY (MII)	92064-13302	92064-13401	1650
92064-16013	%MDMLB	DUMMY LIBRARY (MIII)	92064-13303	92064-13401	1650

Module (Paper Tape) Number	Module File Name	Module Descriptor	Mini-Cartridge	Flexible Disc	Date Code or Revision
92064-16015	%MCL3	CLASS I/O OPTION (MIII)	92064-13303	92064-13401	1650
92064-16016	%MAP3	MIII ABSOLUTE PROGRAM LOADER	92064-13305	92064-13401	1650
92064-16017	%FMGC0	CARTRIDGE FILE MANAGER	92064-13305	92064-13401	1709
92064-16018	%DRC	CARTRIDGE DIR HAN PROGRAM	92064-13304	92064-13401	1650
92064-16019	%TBLCR	CARTRIDGE DIRECTORY TABLES	92064-13304	92064-13401	1650
92064-16021	%DRC1	MI CARTRIDGE DIRECTORY SUBR	92064-13306	92064-13401	1650
92064-16022	%RTMGN	SYSTEM GENERATOR	92064-13305	92064-13401	1709
92064-16023	%RTMLD	RELOCATING LOADER (APP DISC)	92064-13305	92064-13402	1709
92064-16023	%RTMLD	RELOCATING LOADER (GEN DISC)	92064-13305	92064-13401	1709
92064-16024	%RTMSC	LOADER SUB CONTROL (APP DISC)	92064-13305	92064-13402	1709
92064-16024	%RTMSC	LOADER SUB CONTROL (GEN DISC)	92064-13305	92064-13401	1709
92064-16025	%MEDIT	EDITOR		92064-13402	1703
92064-16026	%MASM6	CROSS REFERENCE SEGMENT		92064-13402	1650
92064-16027	%MPF	MI/MII POWER FAIL	92064-13304	92064-13401	1650
92064-16029	%MPF3	MIII POWER FAIL	92064-13304	92064-13401	1650
92064-16030	%MAUTO	AUTOR REL	92064-13304	92064-13401	1650
92064-16031	%MRN	RESOURCE NUMBER MANAGER (MII)	92064-13302	92064-13401	1650
92064-16031	%MRN	RESOURCE NUMBER MNGR (MIII)	92064-13303	92064-13401	1650
92064-16032	%ONMTM	MULTI TERMINAL MONITOR (GEN D)	92064-13305	92064-13401	1650
92064-16032	%ONMTM	MULTI TERMINAL MONITOR (APP D)	92064-13305	92064-13402	1650
92064-16033	!MCGEN	ABSOLUTE CARTRIDGE GENERATOR	92064-13307		1709
92064-16034	%SGPRP	SEGMENT PROGRAM PREP		92064-13402	1650
92064-16035	%MPRMP	PROMPT (MTM)	92064-13305	92064-13401	1650
92064-16036	%MRSPN	RESPONSE (MTM)	92064-13305	92064-13401	1650
92064-16040	%MASM0	ASSEMBLER MAIN CONTROL		92064-13402	1650
92064-16041	%MASM1	ASSEMBLER SEGMENT 1		92064-13402	1650
92064-16042	%MASM2	ASSEMBLER SEGMENT 2		92064-13402	1650
92064-16043	%MASM3	ASSEMBLER SEGMENT 3		92064-13402	1650
92064-16044	%MASM4	ASSEMBLER SEGMENT 4		92064-13402	1650
92064-16045	%MFTN0	FORTRAN MAIN CONTROL		92064-13402	1650
92064-16046	%MFTN1	FORTRAN SEGMENT 1		92064-13402	1650
92064-16047	%MFNT2	FORTRAN SEGMENT 2		92064-13402	1650
92064-16050	%MASM5	ASSEMBLER SEGMENT D		92064-13402	1650
92064-16051	%MXRF0	CROSS REFERENCE MAIN		92064-13402	1650
92064-16054	%DIRD	CARTRIDGE DIRECTORY READ	92064-13304	92064-13401	1650
92064-16055	%FMGF0	FLEX DISC FILE MNGR (APP DISC)		92064-13402	1709
92064-16055	%FMGF0	FLEX DISC FILE MNGR (GEN DISC)		92064-13401	1709
92064-16056	%DRF	F DISC DIRECT PROG (GEN DISC)		92064-13401	1650
92064-16056	%DRF	F DISC DIRECT PROG (APP DISC)		92064-13402	1650
92064-16057	%TBLFP	FLEXIBLE DISC DIRECT TABLES		92064-13401	1709
92064-16060	%DRF1	F DISC DIRECTORY SUB (GEN D)		92064-13401	1650
92064-16060	%DRF1	F DISC DIRECTORY SUB (APP D)		92064-13402	1650
92064-16075	!MFGEN	ABSOLUTE FLEXIBLE DISC SYSTEM		92064-13401	1709
92064-16080	%STRTM	RTE-M SYSTEM START-UP	92064-13304	92064-13401	1709
92064-16081	%MSYLB	RTE-M SYSTEM LIBRARY (APP DISC)	92064-13306	92064-13402	1709
92064-16081	%MSYLB	RTE-M SYSTEM LIBRARY (GEN DISC)	92064-13306	92064-13401	1709
92064-18059	&TBLCR	CARTRIDGE DIRECTORY TBLS SOURC	92064-13304	92064-13402	1650
92064-18126	&MHELP	EDITOR HELP FILE SOURCE		92064-13402	1650
92064-18141	&MAUTO	AUTOR SOURCE	92064-13304	92064-13402	1650
92064-18171	&TBLFP	FLEXIBLE DISC DIRECTORY SOURCE		92064-13402	1709
92202-16001	%DVR23	RTE 7970 9T. MAG. TAPE DRIVER	92062-13304	92064-13401	A
92900-16002	%2DV47	RTE 92900A DRIVER WITHOUT DMS	92062-13302	92064-13401	1643
92900-160[illegible]3	%3DV47	RTE 92900A DRIVER WITH DMS	92062-13302	92064-13401	1643

DOCUMENTATION

The following tables list currently available customer manuals for Data Systems Division products. This list supersedes the list in the last issue of the **COMMUNICATOR.**

The most recent changes to the tables are indicated for easy reference. Prices are subject to change without notice.

Copies of manuals and updates can be obtained from your local Sales and Service office. The address and telephone number of the office nearest to you are listed in the back of all customer manuals.

Update packages are free of charge. If you require an update package only, send your request to:

Software/Publications Distribution
11000 Wolfe Road
Cupertino, Ca. 95014

Customers in the U.S. may also order directly by mail. Simply list the name and part number of the manual(s) you need on the Corporate Parts Center form supplied at the back of the **COMMUNICATOR** 1000.

A few words about documentation terms:

New — A new manual refers only to the first printing of a manual. When first printed, a manual is assigned a part number.

Revised — A revised manual is a printing of an existing manual which incorporates new and/or changed information in its contents. For example, a manual is revised when an update package is incorporated into the manual: the manual gets a new print date and the update package disappears. Note that a revision to a manual effectively obsoletes the previous version of the manual.

Update — An update package is a supplement to an existing manual which contains new and/or changed information. Updates are issued when information must get to customers, yet it is inappropriate to issue a revised manual. An update has no part number; it is automatically included when you order the manual with which it is associated.

1000 SYSTEM MANUALS

PART NUMBER	MANUAL TITLE	PRICE	DATE	UPDATE
02170-90006	HP 1000 Computer System Installation and Service	$ 2.50	3/77	
02172-90005	Getting Started with Your HP 1000	4.00	9/76	
91780-93001	RJE/1000 Programming Manual	9.50	11/76	

RTE SYSTEMS MANUALS

PART NUMBER	MANUAL TITLE	PRICE	DATE	UPDATE
02313-93002	RTE 2313B Analog-Digital Interface Subsystem Operating and Service Manual	30.00	8/76	
02320-93002	RTE System Driver DVR76 for HP 2320A Low Speed Data Acquisition Subsystem Programming and Operating Manual	1.00	8/74	
02321-93001	RTE System Driver DVR74 for HP 2321A Low Speed Data Acquisition Subsystem Programming and Operating Manual	1.00	8/74	
09600-93010	RTE System DVR11 for HP 2892A Card Reader Programming and Operating Manual	1.00	8/74	
09600-93015	91200B TV Interface Kit; Programming and Operating Manual	4.50	7/75	1/76
09601-93007	RTE Device Subroutine for HP 5327A/B-H48 Counter	2.50	12/74	
09601-93009	RTE Device Subroutine for HP 5326A-H18 Counter	2.50	12/74	
09601-93015	RTE for 40-bit Output Register #12556B	1.00	10/74	
09603-93001	9603A/9604A Control System and Scientific Measurement Operating and Service Manual	7.50	5/76	
09610-93003	ISA FORTRAN Extension Package Reference Manual	4.50	7/76	
09611-90009	9611A Operating 406 Industrial Measurement and Control System	.25	4/75	
09611-90010	HP 6940A/B Multiprogrammer Verification Manual	4.50	8/75	

PART NUMBER	MANUAL TITLE	PRICE	DATE	UPDATE
12604-93002	RTE DVR40 for 12604B Data Source Interface	1.00	8/74	
12665-93001	RTE System Driver DVR65 for HP 12771A Computer Serial Interface Kit	1.00	8/74	
12732-90001	RTE Driver DVR33 Programming Manual	2.00	2/77*N	
13197-90001	RTE Driver DVR36 Programming and Operating Manual	3.00	9/76	
24998-90001	DOS/RTE Relocatable Library Reference Manual	10.00	3/76	
25117-93003	RTE System Driver DVR24 for HP 7970 Series Digital Magnetic Tape Unit	1.00	8/74	
29003-93001	RTE System Driver DVR66 for HP 12772A Coupler Modem Interface Kit Programming and Operating Manual	1.00	8/74	
29003-93003	RTE System Driver DVR66 for HP 12770A Coupler Serial Interface Kit Programming and Operating Manual	1.00	8/74	
29009-93001	RTE System Driver DVR62 for HP 2313B Subsystem	2.50	8/74	
29028-95001	RTE HP 2610A/2614A Line Printer Driver	1.50	8/73	
29029-95001	Real-Time Executive System Driver DVR00 for Multiple Device System Control Small Programs Manual	1.50	11/75	
29100-93001	RTE System Driver DVR40 (29100-60041) for HP 12604B Data Source Interface Programming and Operating Manual	1.00	8/76	
29101-93001	RTE Core-Based Software System Users Manual	10.00	1/76	
29102-93001	RTE BASIC Software System Programming and Operating Manual	10.00	3/74	8/75
29103-93001	RTE System Cross Loader; Programming and Operating Manual	2.50	12/76	
91060-93005	RTE Driver for X-Y Display Storage Subsystem (HP Model 1331C-016) Programming and Operating Manual	1.00	8/74	
91062-93003	Real-Time Executive System Driver for DVM/Scanner Subsystem	9.00	8/74	
91700-93001	Distributed System CCE Operating Manual	20.00	12/76	
91705-93001	Distributed System SCE/5 Operating Manual	15.00	12/76	
92001-90015	RTE DVR05 for 264X Terminals	2.00	9/76	
92001-93001	RTE-II Software System Programming and Operating Manual	10.00	3/77*R	
92060-90004	RTE-III Software System Programming and Operating Manual	12.00	7/76	3/77
92060-90005	RTE Assembler Reference Manual	7.00	12/76	
92060-90009	RTE-III General Information Manual	4.00	2/76	
92060-90010	RTE Batch/Spool Monitor and Operating System Pocket Guide	3.00	4/77*R	
92060-90012	RTE: A Guide for New Users	6.50	7/76	
92060-90013	Batch-Spool Monitor Reference Manual	9.50	3/77*R	
92060-90014	RTE Interactive Editor Reference Manual	6.00	3/76	
92060-90017	RTE Utility Programs	3.00	3/77*R	
92060-90020	RTE On-Line Generator	15.00	7/76	3/77
92064-90002	RTE-M Programmer's Reference Manual	14.00	3/77*N	
92064-90003	RTE-M System Generation Reference Manual	7.50	3/77*N	
92064-90004	RTE-M Editor Reference Manual	6.00	1/77	3/77
92200-93001	RTE System Driver DVR12 for HP 2607A Line Printer Programming and Operating Manual	1.00	8/74	
92200-93005	Real-Time Executive Operating System Drivers and Device Subroutine Manual	5.00	7/76	3/77
92202-93001	RTE System Driver DVR23 for HP 7970 Series Digital Mag Tape Units Programming and Operating Manual	1.00	8/74	
92400-93001	92400A Utility Library Subroutine for Sensor-Based Diagnostics	7.50	11/76	
93005-93005	Thermal Line Printer Subsystem for Driver DVR00 (RTE)	2.50	12/74	

HARDWARE MANUALS

PART NUMBER	MANUAL TITLE	PRICE	DATE	UPDATE
02108-90002	HP 21MX Reference Manual	5.50	6/76	7/76
02108-90006	HP 21MX Installation and Service Manual	10.00	7/76	
02108-90004	HP 21MX Operators Manual	5.00	7/76	
02109-90001	HP 21MX E-Series Computer Operating and Reference Manual	8.00	9/76	
02109-90002	HP 21MX E-Series Installation and Service	15.00	8/76	10/76
02109-90006	HP 21MX M- and E-Series I/O Interfacing Guide	7.50	12/76	
12979-90007	HP 12979A I/O Extender Operating & Ref Manual	5.00	12/75	
12979-90006	HP 12979A I/O Extender Install & Service	15.00	6/75	12/76
12990-90003	HP 12990A Memory Extender Inst. & Serv. Manual	5.50	4/76	8/76

LANGUAGE MANUALS

PART NUMBER	MANUAL TITLE	PRICE	DATE	UPDATE
02100-90140	Decimal String Arithmetic Routines	$ 6.50	2/77*R	
02108-90032	HP 21MX M-Series Computer RTE Microprogramming Reference Manual	15.00	10/76*R	
02108-90034	HP 21MX M-Series Computer RTE Microprogramming Pocket Guide	2.75	1/77*N	
02109-90004	21MX E-Series RTE Microprogramming Reference Manual	20.00	10/76	3/77
02109-90008	21MX E-Series Computer RTE Microprogramming Pocket Guide	2.50	11/76	
02116-9014	HP Assembler Manual	6.50	8/75	
02116-9015	HP FORTRAN Manual	6.00	1/77	
02116-9016	Symbolic Editor	4.50	2/74	
02116-9072	ALGOL Reference Manual	10.00	11/76*R	
12907-90010	Implementing the HP 2100 Fast FORTRAN Processor	1.00	7/76	
24307-90014	DOS-III Assembler Reference Manual	8.00	7/74	11/75
92060-90005	RTE Assembler Reference Manual	7.00	12/76	
92060-90016	Multi-User Real-Time BASIC Reference Manual	12.00	2/77	4/77
92063-90001	IMAGE/1000 Data Base Management System Reference Manual	9.00	2/77*R	
92065-90001	RTE-M Real-Time BASIC Language Reference Manual	8.50	2/77	
5951-1321	HP FORTRAN IV Reference Manual	6.00	12/75	

TRAINING SCHEDULE

The schedule for customer training courses on Data Systems Division products has been expanded to include courses offered at our European training centers. Listed below are courses offered in the U.S. and in Europe during the period May 1977 through August 1977.

You can also obtain a copy of the training schedule from your local HP sales office. A European course schedule is available through the sales offices in Europe; a U.S. schedule through U.S. sales offices.

*Prices quoted are for courses at the two U.S. training centers only. For prices of courses at European training centers please consult your local HP Sales Office.

REGISTRATION

Requests for enrollment in any of the above courses should be made through your local HP representative. He will supply the Training Registrar at the appropriate location with the course number, dates, and requested motel reservations. Enrollments are acknowledged by a written confirmation indicating the Training Course, time of class, location and accommodations reserved.

ACCOMMODATIONS

Students provide their own transportation, meals and lodging. The Training Registrar will be pleased to assist in securing motel reservations at the time of registration.

CANCELLATIONS

In the event you are unable to attend a class for which you are registered please notify the Training Center Registrar immediately in order that we may offer your seat to another student.

TRAINING CENTER ADDRESSES

Cupertino

11000 Wolfe Road
Cupertino, California 95014
(408) 257-7000

Sunnyvale

974 East Arques
Sunnyvale, California

Rockville

4 Choke Cherry Road
Rockville, Maryland 20850
(301) 948-6370

Boise

P.O. Box 15
15 N. Phillippi Street
Boise, Idaho 83707
(208) 376-6000
TWX: 910-970-5784

Boblingen

Kundenschulung
Herrenbergerstrasse 110
D-7030 Boblingen, Wurttemberg
Tel: (07031) 667-1
Telex: 07265739
Cable: HEPAG

Winnersh

King Street Lane
GB-Winnersh, Wokingham
Berks RG11 5 AR
Tel: Wokingham 784774
Cable: Hewpie London
Telex: 847178 9

Grenoble

5, avenue Raymond-Chanas
38320 Eybens
Tel: (76) 25-81-41
Telex: 980124

Milan

Via Amerigo Vespucci, 2
1-20124 Milan
Tel: (2) 62 51
Cable: HEWPACKIT Milano
Telex: 32046

Madrid

Jerez No 3
E-Madrid 16
Tel: (1) 458 26 00
Telex: 23515 hpe

Stockholm

Enighetsvagen 1-3, Fack
S-161 20 Bromma 20
Tel: (08) 730 05 50
Cable: MEASUREMENTS
Stockholm
Telex: 10721

TITLE

TRAINING COURSE RATES AND CENTER LOCATION

Course Number	Length	Price	Cupertino	Sunnyvale	Rockville	Boise	Boblingen	Winnersh	Grenoble	Milan	Madrid	Stockholm	Amsterdam/ Brus.
01ETC	RTE II/III Driver Writing Course				Jul 18								
	30 days	300											
22940A	2100 Maint.			May 16 Jul 25									
	10 days	$1000											
22941A	21MX Maint.			May 2 Jun 6 Jul 11					May 9 Aug 8				
	5 days	500											
22942A	7900 Maint.			May 9 Jun 20 Aug 8					Jun 6 Sep 5				
	5 days	500											
22943A	7970B Maint.					Jul 18			Jun 20 Aug 29				
	5 days	600											
22944A	7970E Maint.					Jul 11							
	5 days	600											
22945A	7905 Maint.			Jun 13 Jul 18					Jun 13 Jul 25 Sep 12				
	5 days	500											
22950A	2100 Ser. Assm.		May 23 Jun 13 Jul 11 Aug 1 Aug 22		May 9 Jun 6 Jul 18 Aug 15		May 23 Jul 18 Aug 22	Jun 13 Aug 8	Jun 13 Oct 3	May 30 Sep 5	Jun 13	May 23 Aug 22	May 23 Sep 5
	5 days	500											
22952A	DOS III B		**				Aug 15						
	5 days	500											
22960A	21MX Mic. Prog.		May 9 Jul 18						Jul 18				
	5 days	500											
22965B	RTE-II/III		May 9 May 16 May 16 May 23 Jun 6 Jun 13 Jun 13 Jun 20 Jul 11 Jul 18 Jul 18 Jul 25 Jul 25 Aug 1 Aug 8 Aug 15 Aug 15 Aug 22 Aug 22 Aug 29		May 16 May 23 Jun 13 Jun 20 Jul 11 Jul 18 Jul 25 Aug 1 Aug 8 Aug 15 Aug 22 Aug 29		Jun 20 Jun 27 Aug 1 Aug 8 Aug 29 Sep 5 Sep 26 Oct 3	May 9 May 16 Jul 11 Jul 18 Sep 26 Oct 3	May 9 May 23 Jun 20 Jul 4 Sep 9 Sep 19	Jun 13 Jun 27 Sep 19 Oct 3	Jun 20 Jun 27	Jun 6 Jun 13 Aug 29 Sep 5	Jun 6 Jun 13 Sep 19 Oct 3
	10 days	1000											
	(Course includes RTE-II/III operating system, batch spool monitor and file manager.)												
22968A	Measurement & control						Sep 12		Sep 12				
	2 days	200											

TITLE

TRAINING COURSE RATES AND CENTER LOCATION

Course Number	Length	Price	Cupertino	Sunnyvale	Rockville	Boise	Boblingen	Winnersh	Grenoble	Milan	Madrid	Stockholm	Amsterdam/ Brus.
22969A	Distb. Sys. 5 days	500	May 16 Jul 25 Sep 29		Jun 6		Jul 4		Jun 6				May 9
22977A	Image/DBMS 1000 5 days	500	Jun 6 Jul 11 Aug 1		Jul 11		May 9	Jul 25			Jul 4		
22978	TCS 2 days	200	*										
22979A	Real/Time Multiterminal Basic 3 days	300	**				Sep 14		Sep 14				
22980A	HPIB Multicomputer Bus Basic 3 days	300	Jun 6 Aug 8						Jun 27				Aug 29
22981A	HPIB Programming Under RTE 2 days	200	Jun 9 Aug 11						Jun 30				Sep 1
22983A	21MX E-Micro-programming 5 days	500	Jun 20 Aug 29										
22985A	RTE-M 5 days	500	May 23 Jun 20 Aug 8		Aug 8								

*NOTE: Dates within brackets are starting dates for week 1 and week 2 of the RTE course. In some cases there is a break between the two weeks of the class. Course 22977A, IMAGE/DBMS 1000 replaces 22953A (2100 IMAGE); the new class adds additional material and extends the training from 3 to 5 days.

**On Sufficient Demand.

HEWLETT-PACKARD
COMMUNICATOR SUBSCRIPTION AND ORDER INFORMATION

The Computer Systems COMMUNICATORS are bi-monthly systems support publications available from Hewlett-Packard on an annual (6 issues) subscription.

The following instructions are for customers who do not have Software Service Contracts.

1. Complete name and address portion of order form.
2. For new direct subscriptions (see sample below):
   a. Indicate which COMMUNICATOR publication(s) you wish to receive.
   b. Enter number of copies per issue under Qty column.
   c. Extend dollars (quantity x list price) in Extended Dollars column.
   d. Enter discount dollars on line under Extended Dollars. (If quantity is greater than 1 you are entitled to a 40% discount.*)
   e. Enter Total Dollars (subtract discount dollars from Extended List Price dollars).

   **To qualify for discount all copies of publications must be mailed to same name and address and ordered at the same time.*

SAMPLE

☒ **DIRECT SUBSCRIPTION**

Part No.	Description	Qty	List Price	Extended Dollars	Total Dollars
5951-6111	COMMUNICATOR 1000	3	$48.00	$144.00	
	(if quantity is greater than 1 discount is 40%)			57.60	
	TOTAL DOLLARS for 5951-6111				$86.40

3. To order back issues (see sample below):
   a. Indicate which publication you are ordering.
   b. Indicate which issue number you want.
   c. Enter number of copies per issue.
   d. Extend dollars for each issue.
   e. Enter total dollars for back issues ordered.

   All orders for back issues of the COMMUNICATORS are cash only orders (U.S. dollars only) and are subject to availability.

SAMPLE

☒ **BACK ISSUE ORDER FORM (cash only in U.S. dollars)**
(subject to availability)

Part No.	Description	Issue No.	Qty	List Price	Extended Dollars	Total Dollars
5951-6111	COMMUNICATOR 1000	XX	1	$10.00	$10.00	
		XX	2	10.00	20.00	
				10.00		
	TOTAL DOLLARS					$30.00

4. Domestic Customers: Mail the order form with your U.S. Company Purchase Order or check (payable to Hewlett-Packard Co.) to:

   HEWLETT-PACKARD COMPANY
   Computer Systems COMMUNICATOR
   P.O. Box 61809
   Sunnyvale, CA 94088
   U.S.A.

5. International Customers: Order by part number through your local Hewlett-Packard Sales Office.

HEWLETT-PACKARD
COMPUTER SYSTEMS COMMUNICATOR ORDER FORM

Please Print:

Name _____ **Title** _____

Company _____

Street _____

City _____ **State** _____ **Zip Code** _____

Country _____

☐ **HP** Employee Account Number _____ Location Code _____

☐ **DIRECT SUBSCRIPTION**

Part No.	Description	Qty	List Price	Extended Dollars	Total Dollars
5951-6111	COMMUNICATOR 1000	_____	$48.00	_____	
	(if quantity is greater than 1 discount is 40%)			_____	
	TOTAL DOLLARS for 5951-6111				_____
5951-6112	COMMUNICATOR 2000	_____	25.00	_____	
	(if quantity is greater than 1 discount is 40%)			_____	
	TOTAL DOLLARS for 5951-6112				_____
5951-6113	COMMUNICATOR 3000	_____	48.00	_____	
	(if quantity is greater than 1 discount is 40%)			_____	
	TOTAL DOLLARS for 5951-6113				_____

☐ **BACK ISSUE ORDER FORM (cash only in U.S. dollars)**
(subject to availability)

Part No.	Description	Issue No.	Qty	List Price	Extended Dollars	Total Dollars
5951-6111	COMMUNICATOR 1000	_____	_____	$10.00	_____	
		_____	_____	10.00	_____	
		_____	_____	10.00	_____	
	TOTAL DOLLARS					_____
5951-6112	COMMUNICATOR 2000	_____	_____	$ 5.00	_____	
		_____	_____	5.00	_____	
		_____	_____	5.00	_____	
	TOTAL DOLLARS					_____
5951-6113	COMMUNICATOR 3000	_____	_____	$10.00	_____	
		_____	_____	10.00	_____	
		_____	_____	10.00	_____	
	TOTAL DOLLARS					_____
TOTAL ORDER DOLLAR AMOUNT						_____

☐ **SERVICE CONTRACT CUSTOMERS**

You will receive one copy of either COMMUNICATOR 1000, 2000, or 3000 as part of your contract. Indicate additional copies below and have your local office forward. Billing will be included in normal contract invoices.

Number of additional copies _____

FOR HP USE ONLY

CONTRACT KEY

5951-6111 Number of additional copies _____
5951-6112 Number of additional copies _____
5951-6113 Number of additional copies _____

Approved _____

Please photocopy this order form if you do not want to cut the page off. You will automatically receive a new order form with your order.

HEWLETT hp PACKARD

CONTRIBUTED SOFTWARE
Direct Mail Order Form

NOTE: No direct mail order can be shipped outside the United States.

Please Print:

Name _____ **Title** _____

Company _____

Street _____

City _____ **State** _____ **Zip Code** _____

Country _____

Item No.	Part No.	Qty.	Description	List Price Each		Extended Total	

Sub-total		
Your State & Local Sales Taxes*		
Handling Charge	1	50
TOTAL		

*Tax is verified by computer according to your ZIP CODE. If no sales tax is added, your state exemption number must be provided: # _____.
If not, your order may have to be returned.

Domestic Customers: Cash required on all orders less than $50.00. Mail the order form with your check or money order (payable to Hewlett-Packard Co.) or your U.S. Company Purchase Order to:

HEWLETT-PACKARD COMPANY
Contributed Software
P.O. Box 61809
Sunnyvale, CA 94088

International Customers: Order through your local Hewlett-Packard Sales office. No direct mail order can be shipped outside the United States.

All prices domestic U.S.A. only. Prices are subject to change without notice.

ORDERING INFORMATION

Programs are available individually in source language on either paper tape, magnetic tape, or cassettes as indicated in the abstracts.

To order a particular program, it is necessary to specify the program identification number, together with an option number which indicates the type of product required. The program identification number with the option number composes the ordering number.

For example:

22113A-K01

The different options are:

K01 — Source paper tape and documentation
K21 — Magnetic tapes and documentation

NOTE

Specify 800 BPI or 1600 BPI Magnetic tape.

B01 — Binary tape and documentation
D00 — Documentation
L00 — Listing

Not all options are available for all programs.

Ten-digit numbers do not require additional option numbers such as K01, K21, etc. The 10-digit number automatically indicates the option or media ordered.

For example:

22681-18901 — The digits 189 indicate source paper tape plus documentation.
22681-10901 — The digits 109 indicate source magnetic tape plus documentation (800 BPI magnetic tape)
22681-11901 — The digits 119 indicate source magnetic tape plus documentation (1600 BPI magnetic tape)
22681-13301 — The digits 133 indicate source cassettes plus documentation

Only those options listed in each abstract are available.

Refer to the Price List for prices and correct order numbers.

Hewlett-Packard offers no warranty, expressed or implied and assumes no responsibility in connection with the program material listed.

HEWLETT-PACKARD
LOCUS CONTRIBUTED SOFTWARE CATALOG
DIRECT MAIL ORDER FORM

Please Print:

Name _____ Title _____

Company _____

Street _____

City _____ State _____ Zip Code _____

Country _____

☐ HP Employee Account Number _____ Location Code _____

Part Number	Description	Qty.	List Price Each	Extended Total
22000-90099	Locus Contributed Software Catalog		$15.00	
*If no sales tax is added, your state exemption number must be provided: #_____		Your State & Local Sales Taxes*		
If not, your order may have to be returned.		Handling Charge		1.50
		TOTAL		

Domestic Customers: Mail the order form with your check or money order (payable to Hewlett-Packard Co.) to:

HEWLETT-PACKARD COMPANY
LOCUS CATALOG
P.O. Box 61809
Sunnyvale, CA 94088

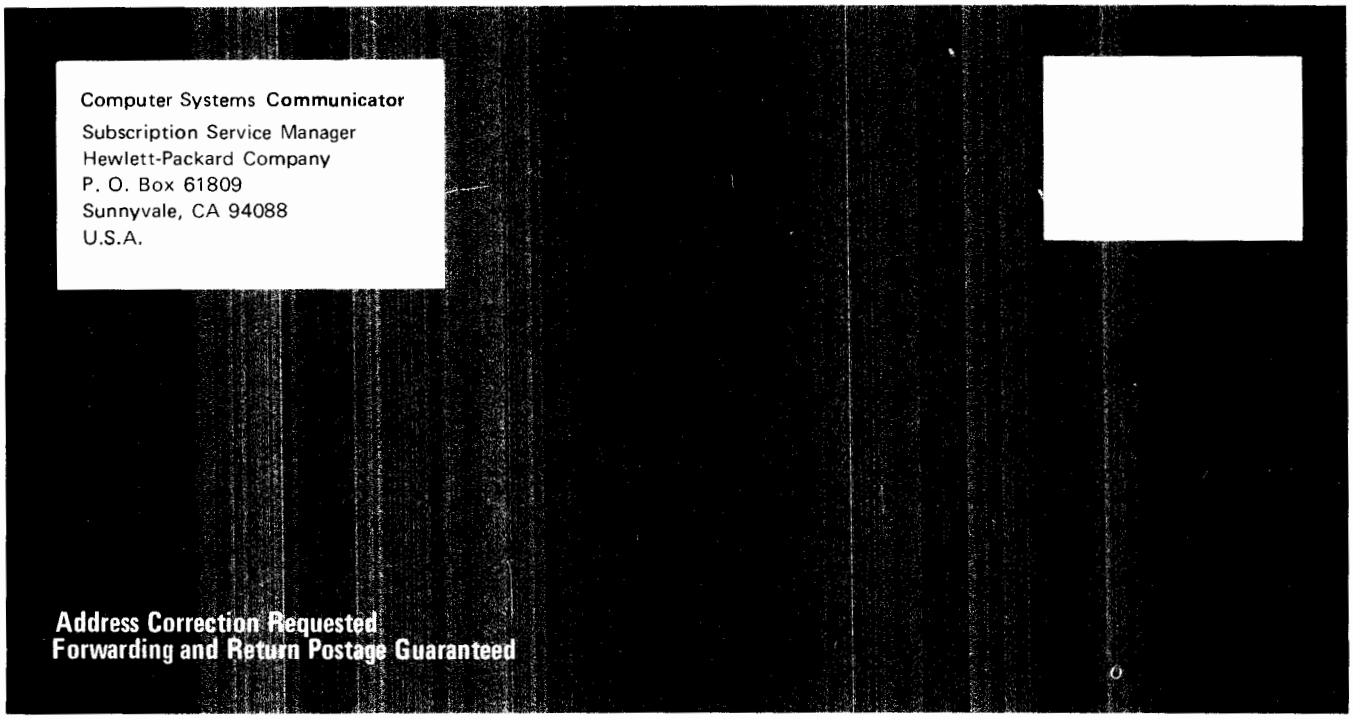
International Customers: Order by part number through your local Hewlett-Packard Sales Office.

NOTE: No direct mail order can be shipped outside the United States. All prices domestic U.S.A. only. Prices are subject to change without notice.

NOTES

Although every effort is made to insure the accuracy of the data presented in the **Communicator,** Hewlett-Packard cannot assume liability for the information contained herein.

Prices quoted apply only in U.S.A. If outside the U.S., contact your local sales and service office for prices in your country.



Printed in U.S.A. 6/77

Part No. 5951-6111