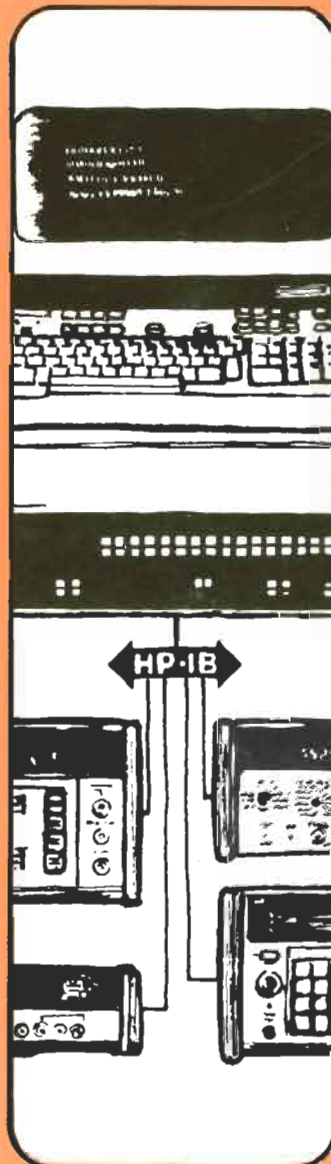
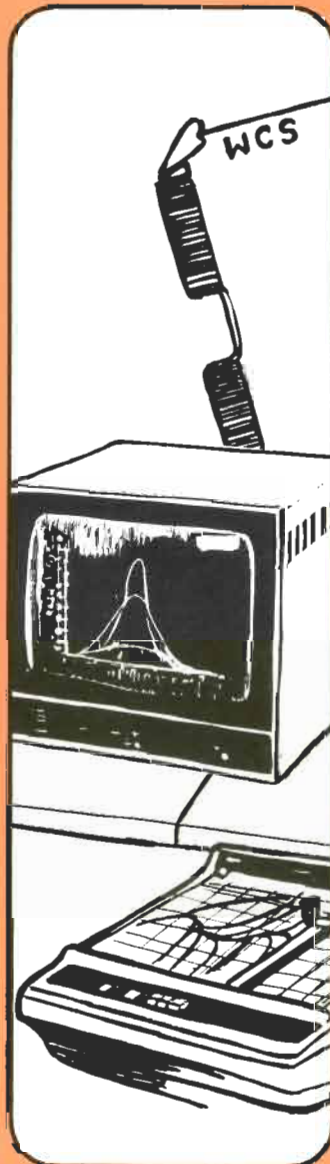


Computer Systems

COMMUNICATOR

```
YBNFI  
J=J+1  
340 CONTI  
DO 30  
YBNFI  
J=J+1  
CONTI  
TERP=  
CALL  
IFC IS  
GO TO  
TERP=  
CALL  
IFC IS  
WRITE  
FORMA  
GO TO  
E  
C  
WRITE  
FORMA  
END
```



HP Computer Museum
www.hpmuseum.net

For research and education purposes only.



ANNOUNCING THE FIRST HP-21 CALCULATOR WINNERS!

The following people have been awarded an HP-21 Scientific calculator for submitting the best feature-length article to the HP-1000 Communicator.

Customer: John A. Danos of Rohm & Haas, Bristol, Pennsylvania, for "Data Acquisition via HP 2313 Subsystem at Low Sampling Rates".

HP Internal: Lyle Weiman of HP Data Systems for "Advanced Debugging Techniques". Lyle is in the Software Engineering Lab.

Both articles are included in this issue of the Communicator.

An HP-21 will be awarded to a customer and to an HP person each issue for the best feature-length article which falls in the categories:

- Operating Systems
- Instrumentation
- Operations Management
- Computation

To be "feature-length", an article must have at least 1600 words, exclusive of listings and illustrations. This rule was relaxed for John A. Danos because the distribution schedule of the Communicator did not permit the rules to be published in time for this issue.

The eligibility rules (repeated from last issue) are:

1. No individual will be awarded more than one calculator per calendar year.
2. In the case of multiple authors, the calculator will be awarded to the first listed author of the winning article.
3. An article which is part of a series will compete on its own merits with other articles in this issue. The total of all articles in the series will not compete against the total of all articles in another series.
4. Employees of Technical Marketing in the HP Data Systems Division (Division 22) are not eligible.

All entries will be judged by a team of at least 3 people in Technical Marketing.

The deadlines for articles for the remainder of 1978 are:

Issue # 5 - August 15th
Issue # 6 - October 15th

All winners are announced in the HP-1000 Communicator in which their winning article appears.

ANNOUNCING OEM CORNER!

This issue marks the start of a major new section of the Communicator - OEM Corner. This section is for HP customers who market software of their own development for use on HP-1000 systems. The software may be a part of a system package which the OEM delivers as a "turnkey" package or a stand-alone software package. HP has many quality OEMs whose products often address markets which are specialized or aimed at a specific application area. Therefore, these products complement the systems offered by HP itself.

In this issue, we have "A Modern Language for On-Line Systems" by David C. Hamilton of Theta Computer Systems in Van Nuys, California.

EDITOR'S DESK

To qualify for inclusion in OEM Corner, an article should be of general interest to our readers and have educational value. That is, it should describe a technique or method of doing something. The article should contain numerous examples and be application-oriented rather than theoretical. We encourage the OEM to describe as many of the features of his product as he wishes but, in all cases, we are looking for general inter-educational value. A reprint of a press release or a marketing brochure is not sufficiently technical to qualify.

We encourage the OEM to place, at the very end of the article, up to 150 words of purely commercial information. This may include prices of the product and ordering information.

The article should be a minimum of 4 typed, double-spaced pages. A typical maximum length might be 10 pages but may exceed this.

Deadlines for specific issues of the Communicator are the same as those above for the calculator. Address all communications to:

Editor HP-1000 Communicator
HP Data Systems Division
11000 Wolfe Road
Cupertino, California 95014
Building 42U

All communications should include the author's address and phone number.

If possible, include the text of the article in machine-readable form, i.e. a file on magnetic tape, mini-cartridge or paper tape.

CONTENTS

User's Queue	1	OEM Corner	
Operating Systems		• A Modern Language for On-Line Systems Development	44
• Know Your DS-1000, Part 2	2	Bit Bucket	
• Advanced Debugging Techniques	15	• Software Samantha	51
• Generating a Minimum RTE II System	23	Bulletins	
• Spooling is Easy with High-Level Interface ...	24	• RTE II/III to RTE IV Upgrade Course Available	52
• Save Time and Effort in Generating Your First RTE-M Basic System	29	• New Training Program for HP-1000 Computer Systems	53
Computation		• Training Schedule	55
• Microcoded Fast Fourier Transform	32	• User Training Services	62
• Plotting on the 9871A Printer Through a 26XX Terminal	33		
Instrumentation			
• Data Acquisition Via HP 2313 Subsystem at Low Sampling Rates	37		

USER'S QUEUE

The User's Queue includes announcements of new programs added to the Contributed Library (LOCUS) and tips, techniques or methods suggested by our readers.

We have not received any new programs for LOCUS since the last issue of the Communicator.

The following letter comes to us on the capabilities of the Formatter.

"I am enclosing a routine I wrote to test the capabilities of the FTN4 Formatter for writing real or double precision numbers using an integer. I have used this capability with other compilers and was quite pleased to see that it worked with the new Formatter.

I have found it quite useful for printing line counters which exceed integer magnitude or to print large floating point numbers without the decimal point.

I hope this will be of use to others who dislike using code to remove the trailing decimal point.

Yours truly,
John Conner
Digicon Geophysical Corporation
3701 Kirby Drive, Suite 112
Houston, Texas 77098"

```
FTN4,L
      PROGRAM FMTST(3,89)
C
C SHOW CAPABILITY TO WRITE REAL OR DOUBLE PRECISION NUMBERS
C IN INTEGER FORMAT.
C
      DOUBLE PRECISION J
      REAL I
      INTEGER LU(5)
      CALL RMPAR (LU)
      IF (LU.EQ.0) LU = 1
100  WRITE (LU,1000)
1000 FORMAT (" ENTER 2 FLOATING POINT VALUES:_" )
      READ (LU,*) I,J
      IF (I.EQ.0.) GO TO 999
C
C WRITE THEM OUT AS FLOATING POINT AND INTEGER
C
      WRITE (LU,1010) I,I,J,J
1010 FORMAT (" #1 AS REAL = "F25.12,/,
&          " #2 AS INTEGER = "I12,/,
&          " #3 AS DOUBLE = "F25.12,/,
&          " #4 AS INTEGER = "I12)
      GO TO 100
999  END
      END$
```

Many thanks to John Conner. I am sure that many of you will also be interested in this capability.

KNOW YOUR DS/1000, PART 2

Al Liu / DSD

This is the second of a series of articles on the internals of the DS/1000 software. To comprehend these articles requires a basic understanding and knowledge of DS/1000 which is described in the DS/1000 Programmer's Reference Manual and the DS/1000 Network Manager's Manual.

The intention of this article is to provide:

- A. a configuration chart of software modules to generate an optimized system at a node,
- B. a functional description of each DS/1000 software module,
- C. the lists and the nodal entry points in the resident subsystem global area (SSGA) module called RES.

NOTE !!!

DS/1000 software is supported by HP only at the user interface level documented in the DS/1000 Programmer's Reference Manual. Subroutines not documented which are described or mentioned herein or elsewhere in the series, must never be called directly from a user program. They are not supported by HP when used in such manner. They are described or mentioned solely for the purpose of illustration. Their functions and calling sequences are subject to change without notice to any user.

The third article in this series will describe:

- a. a node's initialization by LSTEN,
 - b. the error detection and reporting scheme,
 - c. the error recovery scheme.
- A. A Configuration Chart of Software Modules to Generate an Optimized Node

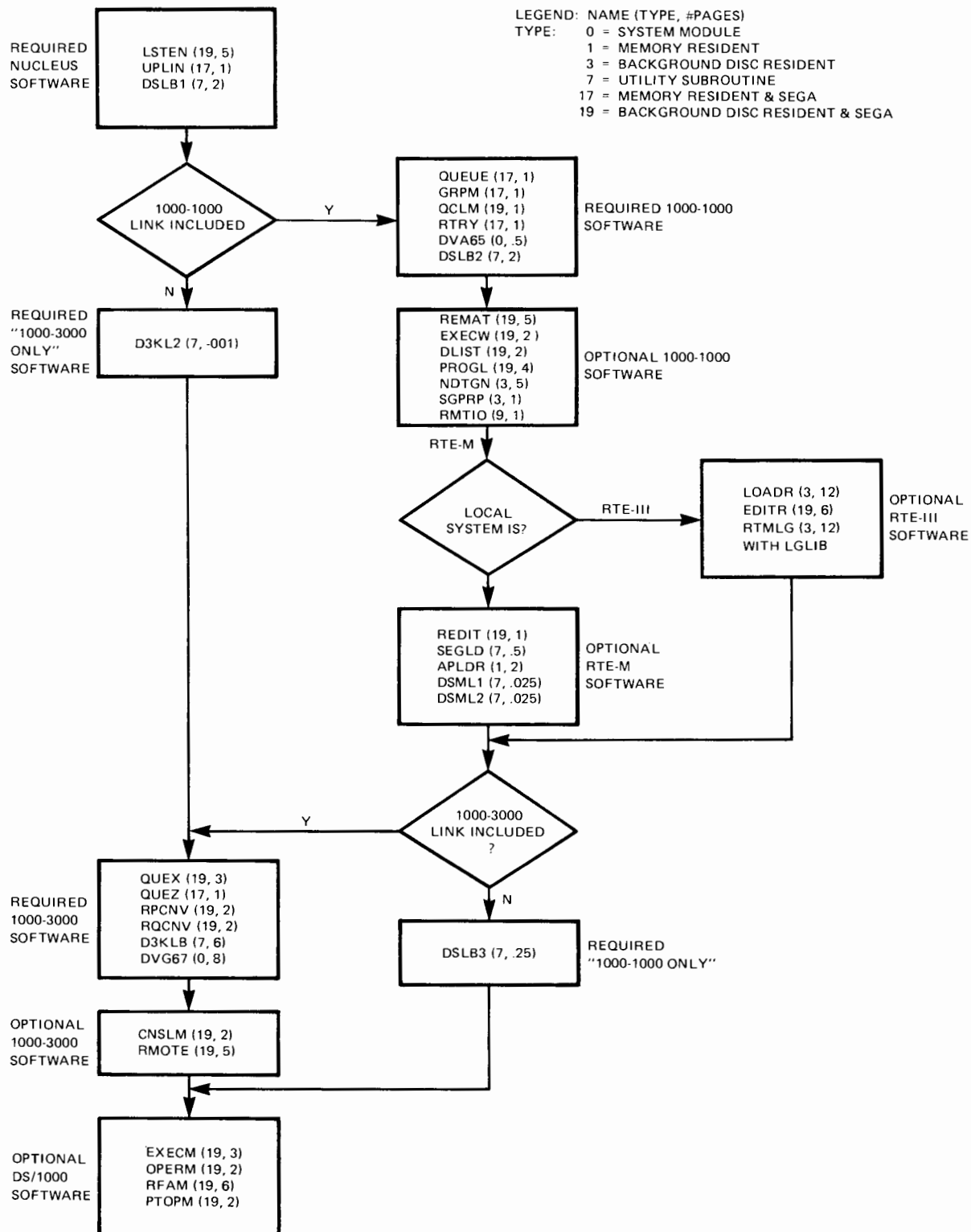
The DS/1000 software numbering catalogue shows a list of 33 relocatable modules (excluding two diagnostic absolute modules). It is unnecessary to include all of them in a node's system generation. The selection of these modules is based upon:

- a. the types of communication link (whether to another DS/1000 or to a DS/3000) are included at this node.
- b. the type of system for the node, i.e. whether a RTE-III, a RTE-MII or a RTE-MIII.
- c. the options to be included at the node depending on the user's applications. For example, if a node is used for program-to-program communication only, then PTOPM module should be included. RFAM (remote-file-access) module and EXECM/EXECW modules (remote-EXEC-calls) need not be included in the generation of the node.

The following chart is intended to help you in selecting the proper modules for an optimized system at a node. In the chart, each module is identified with its "program type by default" (implying the recommended type) and its size in number of pages. The purpose is that as you go along the chart selecting the modules, you will be aware of which modules are memory resident. Totalling their sizes will give you an estimate of the size of the memory resident portion of DS/1000 in the node, which is generally a concern in a system generation especially for RTE-M. Likewise, you also will know the number of disc/partition resident DS/1000 programs in the node and their sizes. This knowledge will enable you to assign partitions optimally in the system generation for the node.

OPERATING SYSTEMS

Following the chart are functional descriptions for these modules. The sequence of modules described follows the order in which they are listed in the chart, top to bottom and left to right.



OPERATING SYSTEMS

B. Functional Descriptions of the DS/1000 Software Modules

Legend : software modules (type, # pages) = description

Required Nucleus Software

- LSTEN (19, 5) = initializes the local node by setting up list pointers in RES, allocating class numbers and resource numbers for inter-module communications and synchronization control, and enabling the communication lines. It is also used to quiesce the node and to change the timeout parameters for each communication link.
- UPLIN (17, 1) = a time-scheduled program (run once every 5 seconds, initiated by LSTEN) to perform the following functions:
- 1) checks and updates timeout values in all transactions (i.e. DS/1000 as well as DS/3000),
 - 2) releases the class buffer and the TCB of a timed out transaction if required,
 - 3) re-schedules GRPM, RTRY, QCLM, QUEX, QUEZ or any Network Interface Monitors (NIMs) if they are dormant,
 - 4) re-enables any downed link.
- DSLB1 (7, 2) = contains library subroutines:
- GET = contains subroutines - GET, ACCEPT, REJECT, FINIS used by the slave program in a program-to-program communication.
- D65GT = does CLASS GET on specified class #, then moves data in the "gotten" class buffer to the caller's buffer when CLASS GET is completed.
- #REQU = re-queues a class buffer to another class or to an EQT.
- D65SV = sends a reply (and data if any) back to the originating node (as a "slave" replying to a "master").
- DRTEQ = given a LU #, it returns the LU's DRT content and its EQT address.
- PGMAD = given a program name, it returns the ID segment address, program status and a flag for long/short ID.
- RES = contains :
- 1) the TCB (Transaction Control Block) list headers and their maintenance subroutine #RSAX.
 - 2) the Transaction Status Table (TST) header for DS/3000 link only.
 - 3) the Nodal Routing Vector Table (NRV) header.
 - 4) the entry points that are global to the node's software modules.
- (The actual TCBs, TSTs and NRVs reside in a large block of System Available Memory allocated by LSTEN when it is initializing the node.)

Required "1000-3000 only" Software

- D3KL2 (7,.001) = contains a dummy entry point for D65MS for a system not linked to any DS/1000 node.

Required 1000-1000 Software

- QUEUE (17, 1) = scheduled by the DS/1000 driver DVA65 when a new transaction is to be read from the interrupting line.
- It performs a CLASS READ of the pending input into either GRPM's class, or PROGL's class in case the transaction is for a down-load.

OPERATING SYSTEMS



- GRPM (17, 1) = General-Request/Reply-Processor-Module for 1000-1000 link is the heart of 1000-1000 communications. All transactions either incoming from or outgoing to a DS/1000 node must go through this module. This includes the store-and-forward operation. The functions of this module have been described in "Know Your DS/1000, Part 1". Its counterpart in 1000-3000 link is QUEX.
- QCLM (19, 1) = prints out error message passed on from GRPM.
- RTRY (17, 1) = error retry for GRPM by holding back the rejected request for a delay time period before giving it back to GRPM.
- DVA65 (0, 5) = the driver for 1000-1000 link. It requires the firmware ROMs.
- DSL2 (7, 2) = contains RFA subroutines and utilities, remote EXEC call subroutines, etc. They are all type 7, i.e. utility.
- 1) DMESG = sends "TELLOP" message to the system console at another node.
 - 2) DMESS = sends operator commands to a DS/1000 node to be executed. Similar to MESSS in HP-1000.
 - 3) FCOPY = transfers a file from any disc to any disc in the network.
 - 4) FLOAD = forces a download of an absolute program into a RTE-M node. It requires a working RTE-M system at the destination node with the DS/1000 APLDR. This is not for cold-boot-loading a remote node!
 - 5) GNODE = returns the local node #, specified in the global constant #NODE in RES, to the caller.
 - 6) RFMST = contains subroutines that make remote file access calls (in other words, FMP calls on a file at a remote DS/1000 node). The following lists these subroutines and their counterparts in FMP.

RFA	FMP
DAPOS	APOSN
DCLOS	CLOSE
DCONT	FCONT
DCRET	CREAT
DLOCF	LOCF
DNAME	NAMF
DOPEN	OPEN
DPOSN	POSNT
DPURG	PURGE
DREAD	READF
DSTAT	FSTAT
DWIND	RWINDF
DWRIT	WRITF

They all call the common subroutine \$PREP in RFMST to build front end of the request buffer and to move the 4-word pseudo-DCB to the request buffer. They all call D65MS to send the request buffer to the DS/1000 node.

OPERATING SYSTEMS

- 7) DEXEC = similar to EXEC calls except DEXEC calls can send requests to another DS/1000 node. DEXEC calls work exactly the same as EXEC calls when applied to the local node. When applied to a remote node, DEXEC only accepts codes 1, 2, 3, 6, 9, 10, 11, 12, 13, 23, 24, 25 and 99. (Code 99, which EXEC does not have, is special to DEXEC only.)

When codes 1, 2, 3, 6, 10, 11, 12, 13, 25, 99 are sent to the remote node, they will be processed by EXECM (i.e. immediate execute) at the remote node.

Codes 9, 23 and 24 will be processed by EXECW (i.e. execute with wait).

- 8) SEGLD = a remote version of RTE-M segment loader. It uses RFA calls (DOPEN, DREAD, DCLOS) instead of FMP calls (OPEN, READF, CLOSE).
- 9) D65MS = counterpart of D3KMS in 3000 link. It is called when a request is to be sent to another DS/1000 node.
- i) allocates a class # for the request (which will become a "master class #").
 - ii) a LOCK/WAIT on the RES table access resource number, #TBRN.
 - iii) calls #RSAX to add a TCB to the master-request list.
 - iv) sends the request (and data if any) to the remote DS/1000 node by a CLASS WRITE-READ to GRPM's class (#GRPM).
 - v) waits for reply from the remote DS/1000 node by doing a CLASS GET (via subroutine D65GT) on the "master class #". (GRPM will write the reply into this class when it receives the reply via DVA65.)
 - vi) de-allocates the "master class #" and removes the associated TCB from the master-request list.
- 10) D65AB = called by FLOAD, DEXEC and D65MS to output abort messages for the program that is using these subroutines if an error occurs. The message format is:

```
DSXX: [program's name] YYYYYY
* [program's name] ABORTED ! *
```

where XX = error code (App. B in Prog. Ref. Manual)

YYYYYY = address within the aborting program where the error is detected.

It then terminates the program by CALL EXEC (6).

Note : D65AB is called only from master-level routines such as DEXEC and only when the user does not want to process errors himself. It is never called from those master routines that always require the user to process errors himself such as the RFA subroutine calls.

Optional 1000-1000 Software

REMAT (19, 5) = operator interface for remote RTE/FMGR commands

EXECW (19, 2) = a Network Interface Monitor (NIM) that processes DEXEC CALLS 9, 23 and 24 (schedule with wait and queue with or without wait) from another node.

OPERATING SYSTEMS

- DLIST (19, 2) = a Network Interface Monitor (NIM) that services the DL (Directory List) request from a remote node's REMAT program by sending the directory list output to the requesting node.
- PROGL (19, 4) = a Network Interface Module (NIM) that services the program down-load request from a remote node.
- NDTGN (3, 5) = a utility program that generates a Network Description Table which is composed of Nodal Routing Vectors (NRVs), one for each node in the network.
- SGPRP (3, 1) = a utility program that prepares a RTE-M segmented program which has been relocated by RTMLG, for loading into a remote RTE-M node by that node's APLDR. It can be run in either a RTE-M or a RTE system.
- RMTIO (7, 1) = a remote/local version of FMTIO (RTE FORTRAN format I/O) which allows the FORTRAN READ and WRITE statements to be executed to a remote node. Refer to p. 2-14 in the Network Manager's Manual for details. This module should not be generated into the system. Instead it should be relocated on-line using LOADR or RTMLG to append it to the calling program.

Optional RTE-III Software

- LOADR (3,12) = the relocating LOADR similar to RTE-III's LOADR , except that it relocates directly from binary relocatable disc files instead of from LOAD-AND-GO tracks. Page size is variable depending on the size of the symbol table required by the program being relocated.
- EDITR (19, 6) = this version of the EDITR provides editing of a disc file on a remote disc-based node. Page size is a variable.
- RTMLG (3,12) = the RTE-M LOADR/GENERATOR which is run on a RTE-III node to generate a RTE-M system or to relocate a RTE-M program for a remote node. Page size is a variable.

Optional RTE-M Software

- REDIT (19, 1) = a utility program to be used only on a RTE-M node. It does not perform an editing function itself, but uses DEXEC 9 to schedule EDITR at the remote node to do the editing. Refer to p. 2-16 in the Network Manager's Manual for details.
- SEGLD (7,.5) = the remote version of the RTE-M subroutine by the same name, which can load a program's segment from a disc file in a remote node. It should be used only in a RTE-M node.
- APLDR (1, 2) = the remote version of the RTE-M program by the same name, which can load RTE-M programs from a remote node.
- DSL1 (7,.025) = contains the subroutine MSTAT , which is the RTE-M version of RTE-FMP's FSTAT subroutine.
- DSL2 (7,.025) = contains dummy entry points for :
APOSN, CLOSE, CREAT, FCONT, LOCF, NAMF, OPEN, POSNT, POST, PURGE, READF, RWNDF, WRITF
This module is required only if the RTE-M node does not include the disc filing system.

Required 1000-3000 Software

- QUEX (19, 3) = reads and sends data from/to 3000. Calls DVG67.
***** The heart of 1000-3000 communication *****
***** The counterpart of GRPM in 1000-1000 *****

OPERATING SYSTEMS

- QUEZ (17, 1) = wakes up QUEX from its CLASS GET suspend when impending input from 3000 is detected by DVG67.
- RPCNV (19, 2) = converts DS/1000 replies, going out to a DS/3000 node, to the DS/3000 format and passes it to QUEX to be sent.
- RQCNV (19, 2) = converts DS/3000 requests, coming into the DS/1000 node, to the DS/1000 format and passes it to one of the Network Interface Monitors (NIMs):
RFAM, EXECM, PTOPM, CNSLM, OPERM.
- D3KLB (7, 6) = contains the following type 7 routines (i.e. utility)

- 1) the file handling routines for a DS/3000 file. The general functions they go through are:

- Step # 1 = build front end of request buffer
Step # 2 = move caller's parameters to request buffer
Step # 3 = send the request buffer to 3000 by calling subroutine D3KMS, which writes the request buffer to QUEX's class.
Step # 4 = wait for reply from 3000 (i.e. via the chain of DVG67→QUEZ→QUEX→itself).
Step # 5 = pass the returned parameters/data in the reply back to the caller.

- a) FOPEN = to open a file on DS/3000
- b) FREAD = to read a record from a DS/3000 file. Additionally in step # 5, the received data are de-blocked into DS/1000 records.
- c) FWRIT = to write a record to a DS/3000 file. Additionally in step # 2, data from the caller are blocked into the request buffer.
- d) FCHEK = to check the status of a DS/3000 file.
- e) FINFO = to get access info (such as record size, block size, record pointer, etc.) and status of a DS/3000 file.
- f) FCLOS = this routine contains the following subroutines and namesake functions:

CLOSE, READSEEK, READLABEL, WRITELABEL, SPACE, POINT, CONTROL, SETMODE, RENAME, RELATE, LOCK and UNLOCK.

- 2) the routines for sign-on and sign-off on DS/3000.

- a) HELLO = must be called by a user program before initiating any communications with DS/3000. It sets up communications to DS/3000 and tries to create a remote session on HP-3000. DS/3000 will reply with the "Session Process #" for the session created. HELLO then sets it into a TCB which is added to the "process #" list.
- b) BYE = to terminate communication with DS/3000, causing DS/3000 to release the session. After sending a "BYE" and getting an "OK" reply from DS/3000, it removes the TCB with the process # from the "process #" list.

- 3) other routines :

- a) D\$EQT = extension of EQT for DVG67 (to bypass the system allocation, i.e. not using the "X" field in EQT definition). It contains long term statistics and an event trace table.

OPERATING SYSTEMS

- b) POPEN= contains the PTOP master subroutines for DS/3000 as well as for PTOP to DS/1000. These subroutines should be used by the master program in order to initiate each PTOP transaction with the slave program on the other node.
- c) D3KMS= counterpart of the DS/1000 D65MS subroutine. It is called in order to send a DS/3000 request to QUEX. Its functions which depend on the bits set in the control word of the calling sequence , are a combination of :
 - i) get a class # for the request (which becomes a "master class #" within DS/1000).
 - ii) LOCK/WAIT on the RES table access RN #TBRN.
 - iii) add a TCB to the "master-request list" as a new entry.
 - iv) store the "master class #" into the request buffer as a "FROM PROCESS #".
 - v) send the request to DS/3000 by writing it to QUEX's CLASS (#QXCL in RES).
 - vi) wait & get reply from DS/3000 by doing a CLASS GET on the "master class #". (QUEX will write the reply into this class when it receives the reply from DVG67).
 - vii) de-allocate the "master class #" and remove the associated TCB from the "master-request list" in RES.
- d) HSLC= the logical driver part of DVG67, which resides in Sub-System Global Area (SSGA). It defines the bisync protocol subset used for DS/3000 communications.

DVG67 (0 ,8) = The physical driver for 3000 link. Interrupts from the DS/3000 link are processed here. It is called from QUEX in the form of EXEC I/O calls.

Optional 1000-3000 Software

CNSLM (19, 2) = handles unanticipated standard list request (\$STDLIST) or standard input request (\$STDIN) from 3000 , such as warning messages from the 3000 operator.

RMOTE (19, 5) = provides a "virtual terminal" to the HP-3000 system.

Required "1000-1000 only" Software

DSL3 (7,25) = needed only if there is no 3000 link. Contains a dummy entry point , D\$EQT , to satisfy UNDEFs (which are referenced by LSTEN) and a shortened POPEN module which does not contain the PTOP master subroutines for DS/3000.

Optional DS Software

EXECM (19, 3) = a Network Interface Monitor (NIM) that is required only if its remote nodes are to send DEXEC calls to it.

OPERM (19, 2) = a Network Interface Monitor (NIM) that is required only if its remote nodes are to send RTE operator commands (via the REMAT program and the DMESS call).

OPERATING SYSTEMS

- RFAM (19, 6) = a Network Interface Monitor (NIM) that is required only if its remote nodes are to make RFA (Remote-File-Access) calls to it. Page size is a factor of the number of files to be opened concurrently (i.e. the number of active DCBs) at the node.
- PTOPM (19, 2) = a Network Interface Monitor (NIM) that is required only if its remote nodes are to make PTOp (Program-To-Program) calls to it. The PTOp type of communication provides the most basic and efficient communication among any nodes in a network. A user can design the PTOp programs for each node so that they can handle the RFA and the DEXEC type of calls without requiring those NIMs at the nodes.

C. Description of "RES"

"RES" is a module that resides in the subsystem global area (SSGA) in the RTE system. All DS/1000 modules (except a few such as RTMLG and LOADR) must access this module. Therefore, they are either type 17 (memory resident and accessing SSGA) or type 19 (disc/partition resident and accessing SSGA) modules.

The module "RES" contains the following:

- a) #RSAX = a privileged library routine which controls the access to and maintains the node's TCBs (i.e., transaction-control-block) for all current requests and replies.
- b) A table of TCB list-headers (i.e. each item in this table is the starting address of a list of TCBs). The format of each list is shown in App. C in the DS/1000 Programmer Reference Manual.

All TCBs actually reside in the System Available Memory. Each TCB in every list is 5-words long and is single-directionally linked to the next TCB in the list. First word in a TCB contains the address of the next TCB. First word in the last TCB in a list contains a zero to mark the end of that list.

There are four major types of TCB lists.

- 1) #PNLH = the DS/3000 Process Number List Header

Each TCB represents a current LOGON (i.e. HELLO) to DS/3000. When a LOGON is established at the 3000 system, it gets a process number assigned to it, which is kept in Word 3 of the TCB. Within this list, master-requests and slave-replies are intermixed (i.e. not kept in separate lists as in the DS/1000 link).

Refer to section C-4.1 in the Network Manager's Manual for format description.

- 2) #MRTH = the DS/1000 Master-Request List Header

Each TCB represents a master-request currently "unreplied" (i.e. it has not yet received the associated slave-reply from the other node).

Refer to section C-4.2 in Network Manager's Manual for format description.

- 3) #STxx = the DS/1000 Slave-Reply List Headers

"xx" stands for the type of slave-replies that are linked to this list. The types (i.e., stream types) are assigned as follows:

- | | | |
|----|-----|--|
| 01 | for | DLIST |
| 02 | for | CNSLM (operator requests for DS/3000 link) |
| 03 | for | EXECW (for DEXEC calls with WAIT) |
| 04 | for | PTOPM |
| 05 | for | EXECM (for DEXEC calls without WAIT) |

OPERATING SYSTEMS

06 for RFAM
07 for Operator requests (for DS/1000 link)
09 for PROGL

At present, stream type 00, 08 and 10 are unused.

Each TCB in these lists represents a slave-reply that is to be sent to another node. A TCB will be removed from its "slave-reply" list when the reply has been "sent" by either a CLASS WRITE to GRPM's class (#GRPM) for DS/1000, or to RPCNV's class (#RPCV) for DS/3000 reply.

Refer to section C-4.3 in Network Manager's Manual for format description.

4) #NULL = the Null List of Available TCBs

The 5-word TCBs are linked together to form a pool of unused TCBs. This list is first allocated and initialized by LSTEN when it is initializing the node. The other lists are empty, i.e. the list-headers #PNLH, #MRTH and the #STxxs all contain zero.

As requests and replies are needed, TCBs are allocated from the front of the null list to the appropriate lists. As requests and replies are successfully completed, their TCBs are returned to the end of the null list.

When the null list is exhausted (i.e. when #NULL contains a zero) at a node, it results in a local-busy condition at that node for any new incoming request.

c) A table of nodal entry points that are initialized by LSTEN (with parameters from the operator) when LSTEN is run for the first time at the node, or set by LSTEN at a later time as modifications.

1) #FWAM = the starting address of the block of SAM that LSTEN requests from the system for the storage of TCBs, TST and NRVs.

2) #SAVM = the number of words in the SAM block.

3) #TBRN = the resource number to access the TCB lists.

Prior to calling #RSAX, a DS module must call RNRQ on this resource number with wait and LOCK, which may cause it to be placed in state 3 (suspended) when there is no TCB in the #NULL list at the node.

4) #QRN = the resource number for quiescing the node.

It is set by LSTEN when the operator enters the "/Q" command.

5) #GRPM = the class number that GRPM always does a CLASS GET on.

Any transaction to be sent to a DS/1000 node is passed to GRPM by doing a CLASS WRITE to #GRPM such as by D65MS and by RTRY.

6) #QCLM = QCLM's class number. Any communication error detected by GRPM and which it does not "RTRY" is sent to QCLM to be printed on the system console. This avoids having GRPM to be I/O bound for error processing.

7) #BUSY = the number of active TCBs at the node, i.e. the number of TCBs that has been taken out of the #NULL list and been placed into other lists.

OPERATING SYSTEMS

This counter is managed by #RSAX routine. It is checked by QUEUE just before QUEUE terminates. If there are still some active TCBs, QUEUE terminates at once. If there is no active TCB, the node might be "quiesced" and QUEUE performs an "insurance check". It calls RNRQ to globally lock and clear the quiescent resource number #QRN. If #QRN is not locked to anybody, this is a harmless call and QUEUE will terminate at once. However, if #QRN is locked, the node is quiesced. QUEUE will be suspended at state 3, waiting for #QRN to become unlocked. During that state, DVA65 receiving transmission from other nodes will not be able to schedule QUEUE (QUEUE not dormant) and therefore rejects those transmissions with the "REMOTE-BUSY" error to the originating nodes.

- 8) #MSTO = the Master-Request Timeout Value.

It is applied to all master-requests to be sent to a node that does not have a "master-request timeout override" specified in its NRV entry.

Its default value (45 secs) is "hard coded" into LSTEN. It can be changed by the "/T" command in LSTEN and a new value is entered for "MASTER T/O = ".

(The master-request timeout override is specified by the answer to "CPU#,LU,TIMEOUT ?" question in LSTEN's initialization of the node or NDTGN's building of the Network Description Table. It is stored in the second word of the NRV entry. Refer to section C-4 in Network Manager's Manual for format description.)

- 9) #SVTO = the Slave-Reply Timeout Value.

Its default value (30 secs) is "hard coded" into LSTEN. It can be changed by the "/T" command in LSTEN when a new value is entered for "SLAVE T/O = ".

- 10) #RTRY = the class number on which RTRY always does a CLASS GET.

When a transmission (request or reply) has been rejected by the "REMOTE-BUSY" error at the remote DS/1000 node, and provided #BREJ is not zero, GRPM passes this transmission buffer to RTRY by re-queuing it to #RTRY. RTRY brought out from its suspend state 3, will attempt the re-transmission after a delay time period.

- 11) #WAIT = the "REMOTE-QUIET WAIT" time in seconds.

It is specified by the "/T" command in LSTEN. When a "REMOTE-BUSY" error persists at a DS/1000 node and the "Remote-busy-retries. #BREJ" has been exhausted, the calling program at the local node can be optionally placed on the time list with a delay specified by #WAIT is zero, the calling program is not re-scheduled and "DS08" error is determined at once.

- 12) #SWRD = the node's security code.

It is input by the operator at LSTEN's initialization of the node. LSTEN uses it to check operator's input for the "/Q, /T and /R" commands. It is also used by REMAT when REMAT processes the "SW" command.

- 13) #BREJ = the retry count for "REMOTE-BUSY" reject.

Its default value (3) is set by LSTEN. It can be changed with the "/T" command in LSTEN when a new value is entered for the "REMOTE-BUSY RETRIES [1 TO 10] ?" question.

The count is actually stored into bits 8 to 11 in #BREJ. When it is used, it is set into the first word of a request/reply buffer by either GRPM, D65MS or EXECM when they are setting up this buffer.

OPERATING SYSTEMS



It is counted down by GRPM on the receiving node which has the "REMOTE-BUSY" condition and stored back into the transmitted buffer, which is sent back to the originating node. When it reaches zero and the originating node detects it, "DS08" error is returned to the user.

- 14) #RPCV = the class number upon which RPCNV always does a CLASS GET.

The Network Interface Monitors (NIMs) send replies to both DS/1000 and DS/3000 nodes. When the reply is intended for a DS/3000 node, the NIM re-queues the reply buffer from its class to #RPCV in order for RPCNV to GET the buffer and convert it to DS/3000 format.

- 15) #RQCV = the class number upon which RQCNV always does a CLASS GET.

QUEX and D3KMS both do CLASS WRITES to #RQCV to pass the request buffers received from DS/3000 in order for RQCNV to convert them into DS/1000 format and pass them to the proper NIM to be serviced.

- 16) #LU3K = the logical unit number for the DS/3000 link.

It is set by LSTEN in its initialization of the node when "LU OF HP 3000?" is answered. It is used in all EXEC calls for I/O to the DS/3000.

- 17) #QZRN = the resource number by which DVG67 wakes up QUEZ.

It is initially locked globally by QUEZ and passed to DVG67 as one of the parameters in an EXEC call for "special read" by QUEZ. QUEZ then terminates, but is scheduled by QUEX later and repeats the LOCK/WAIT on #QZRN. At that time, QUEZ will be suspended in state 3 because #QZRN is still being LOCKED.

When DVG67 receives an end-of-transmission from DS/3000, it unlocks #QZRN, which takes QUEZ out of suspension. QUEZ then does a CLASS WRITE of zero length record into #QXCL (QUEX's class) to bring QUEX out of its CLASS GET suspension. QUEX can then process the received transmission from DS/3000.

QUEZ LOCKs #QZRN to itself again and then terminates. When QUEX completes its processing, it schedules QUEZ to put QUEZ back into suspend state 3 before going back to its own CLASS GET again.

This process is repeated for every DS/3000 transaction through the use of #QZRN and #QXCL.

- 18) #QXCL = the class number on which QUEX always does a CLASS GET.

A transaction (request or reply) which is to be sent to DS/3000, is passed to QUEX by a CLASS WRITE to #QXCL. This transfers the request/reply buffer and the data buffer, if any, to QUEX.

- 19) #TST = the Header for the DS/3000 Transaction Status Table.

This is a two-word header. The first word contains the starting address of the Transaction Status Table and the second word contains the number of words in the Transaction Status Table.

- 20) #RFSZ = the number of "OPEN" RFA files at the node.

It is set by the answer to "INPUT # OF FILES:" in LSTEN's initialization. It is used only by the multiple DCB version of RFAM when it initializes by building up the DCB directory list "RFAMD" (each entry being 9-words long) in the remaining portion of RFAM's memory partition. If the number specified by the operator exceeds the available memory, a message will be output to the system console and #RFSZ will be set to the maximum value allowed for the memory available.

OPERATING SYSTEMS

21) #CNOD = the Current User Node number.

It is applicable only when a DEXEC call from another node is being processed by EXECW (i.e. stream type 3).

It contains the node number from which the DEXEC call originates. (The source node is in word 3 of the master-request buffer received.) Since the master-request is to be executed with wait, the information is saved in #CNOD.

When the master-request is successfully executed, #CNOD is reset to -1 by EXECW, indicating no more active remote execute-with-wait request at this node.

22) #LNOD = the Down-load node number.

This is special only to the DEXEC call for remote-schedule of APLDR for down-loading. It contains the node number from which an RTE-M program is to be down-loaded.

23) #NODE = the node number of the local node.

It is set by the answer to "LOCAL CPU # ?" in LSTEN's initialization. It is set into word 3 of every request and reply buffer that is to be sent to another DS/1000 node. It is used in the following modules:

EXECW, OPERM, DLIST, REDIT, PTOPM, RFAM

24) #NCNT = the negative number of NRVs (Nodal Routing Vectors).

It is set by the answer to "NUMBER OF NODES ?" in LSTEN's initialization. Each NRV is 2-words in length. It is used in GRPM, RTRY, D65MS and EXECM as a loop counter for searching through the NRV table for the logical unit that corresponds to a specified node number.

25) #NRV = the starting address of the NRV Table.

The NRV table is in SAM and is allocated as part of the large SAM block that LSTEN has requested during its initialization. The rest of this SAM block is used for the TCBs and the TST (which is for DS/3000 link only).

ADVANCED DEBUGGING TECHNIQUES

by Lyle Weiman

In a previous article, the use of DBUGR was described for debugging programs. In addition, it was mentioned that interactive debugging can introduce delays which may change the characteristics of the problem, or it may even vanish entirely. Bugs of this nature are extremely hard to find, but a few techniques have been developed to aid in finding them. This article describes some of them.

The Aborting Program Freezer

Often, the 'bug' you're looking for manifests itself by causing a program to commit some mortal sin and be aborted by RTE, or can be forced to do so in some clever way. In this case, RTE washes its hands of the offender in very short order. Except for a cryptic message on the operator console describing the error, no trace is left of the program, and therefore it is very difficult to determine what really caused the problem in the first place. There is a program in the HP Contributed Library (LOCUS) which can help you here. It's called 'ABDMP' (for the Aborting Program freezer), and gets its name because it "freezes" a program in its partition on any abortive error, rather than let RTE terminate it. You can perform a "post mortem" examination on it interactively, using the features of DBUGR to trace lists, interpret instructions and data in the most useful fashion. This program has been tested only on RTE-IV, but could probably be modified to work in RTE-II/III fairly easily.

'ABDMP' allows you to establish a list of programs which are to be "frozen" should they be aborted for any reason, at any time. It doesn't matter how infrequently the error occurs: any abortive error causes the "freezing" (normal terminations are not trapped). Should any other program abort which is not in the list, RTE is allowed to take its own action. Alternatively, you may specify that all programs are to be added to the list. This is useful when the error seems to cause various programs to abort at random, or programs are run at different times using different ID segments (or different names).

It is assumed that the reader is familiar with the DBUGR Utility of RTE-IV, described briefly in a previous Communicator article. You may wish to refer to that article to refresh your memory.

The main program is called 'ABDMP'. You run it, and specify the name of one or more programs which are to be "frozen" for your examination, should they be aborted by RTE. 'ABDMP' inserts a "patch" into the RTE System Message Processor, \$ERMG, so that some "special" code can be executed first. \$ERMG is entered whenever RTE wants to abort a program abnormally (Memory Protect, Dynamic Mapping, IOxx errors, etc.). If you are unfamiliar with this routine, review the Communicator article, "Know Your RTE" describing program aborting. The "special" code is contained in the module 'SPATC', supplied with 'ABDMP', which compares the ID segment address of the currently-executing program (which is the one being aborted, usually) with the IDs of all programs in its list. The exception is when an operator aborts a program.

In RTE-IV, the instruction following the entry point to \$ERMG is a NOP.

'ABDMP' allows you to add or delete programs from this list, or print all programs in the list. Any program which appears in the list will be "frozen" if it aborts; that is, it will be locked in its partition, and you will be given a chance to examine it. You will have all the power of the DBUGR utility to examine the program, but you can't execute or patch any part of it. Furthermore, you will be running in one partition, and examining another, one page at a time. If you wish to examine a different page, you must exit DBUGR and use 'ABDMP' to set up the map properly and copy another page, then re- enter DBUGR. You exit DBUGR by typing the two keys 'Escape' and 'P'.

After examination, the aborting program is "unfrozen", and RTE completes the process of aborting it (releasing its resources, etc.)

Alternatively, you can specify that any program is to be "frozen" when it aborts (use -1 instead of program name for the Add Program and Delete Program commands).

You may also dump the program to a lineprinter, mag tape, etc., for off-line analysis.

OPERATING SYSTEMS

One of the modules in this package, 'SPATC', must be generated into the system. When you run the main program, 'ABDMP', a call to this routine is made to be sure that the patch in \$ERMG has been inserted. This routine compares the ID segment address of the currently-executing program with those in its list. If no match is found, \$ERMG is re-entered, after simulating the overlaid instruction (if any), and the normal RTE program-abort path is taken. If a match is found, then \$LIST is called to place the program in State 3 (General Wait). The ID segment is "doctored" so that no event can take it out of this state (it is placed in its own "wait" queue), the "memory-lock" bit is set, and other information is stored in the remaining three locations in the ID which specifies the reason the program was aborted (for this reason, you cannot examine the 'temporary' words of the ID segment to determine the EXEC call which the aborted program last executed). #LIST is called to schedule 'ABDMP'. \$ERMG is by-passed, and return is made to its caller (i.e., RTE does not abort the program).

When 'ABDMP' is scheduled, it asks for operator input. The lists may be updated, listed, etc.

SCHEDULING 'ABDMP'

***ON, ABDMP, <lutty>, <lulist>**

where <lutty> = lu of your terminal

<lulist> = list device for program dumping
(default=6).

The program prints its name, and asks you for commands.

ABORTING PROGRAM DUMPER

Commands are:

```
AP,PROG = ADD PROGRAM "PROG" TO LIST OF PROGRAMS
DP,PROG = DELETE PROGRAM "PROG" FROM LIST OF PROGRAMS
PROG    = PROGRAM NAME, OR -1 TO FORCE DUMP ON ALL ABORTS
LI      = PRINT LIST OF PROGRAMS
EX      = EXIT SETUP PHASE, WAIT FOR PROG TO BE ABORTED
DU,PROG = DUMP PROGRAM [MUST BE IN MEMORY]
NOTE: "LIST OF PROGRAMS" REFERS TO LIST OF PROGRAMS
      TO BE DUMPED, IN CASE THEY ABORT.
```

Any illegal command will yield the message ILLEGAL COMMAND.

The **DU,PROG** form can be used to dump or analyze a program currently in any partition which has not been aborted. It is preferable to place the program in a suspended state before doing so, however. Exiting 'ABDMP' after examining a program using the DU,PROG command will not abort the program.

When you terminate (EX command), the program scans ID segments for programs whose ID segments have been doctored as described above. If none are found, the program waits for ten seconds, then repeats the search. This is repeated forever, thus guaranteeing that no aborted program will ever be missed, no matter how many are aborted, in which order, or what 'ABDMP' was doing at the time. To re-schedule the main program (to add, delete or list programs), simply set its "break" bit, and wait for it to be re-scheduled by RTE. Alternatively, you may abort it and re-schedule it, using "NOW":

```
*OF, ABDMP, 1      (remove program from time & scheduled lists)
*ON, ABDMP, NOW, 1
```

OPERATING SYSTEMS

When a "frozen" program is detected, 'ABDMP' prints the following message:

```
PROGRAM aaaaa ABORTED, xxnn  oooooo  PRIORITY ddddd, PRIMARY ENTRY POINT oooooo TYPE d
A=oooooo, B=oooooo, E/O=oooooo
LOW MAIN =oooooo, HIGH MAIN + 1=oooooo
LOW BP =oooooo, HIGH BP + 1 =oooooo
PARTITION # dd, SIZE =dddd
```

where **aaaaa** = program name, **xxnn** = 4-character reason (**MP** ,**I007**, etc.) **oooooo** = octal number **dddd** = decimal number

PARTITION # dd is the partition number in which the program was running when it was aborted. Size is the size of the program in pages.

'ABDMP' enters the "examine program" phase, reminding you with the prompt

**EXAMINE ABORTED PROGRAM
COMMAND?**

Entering ?? will print the following menu:

```
DISPLAY COMMANDS ARE:
DI,LU = DISPLAY CONTENTS OF ABORTING PROGRAM'S PARTITION
AD,ADDR=SET UP MAP SO THAT <ADDR>'S PAGE
        MAY BE EXAMINED WITH DBUG
DB      =CALL DBUG (USUAL DBUG COMMANDS CAN BE USED
EX      =EXIT. ABORTING PROGRAM IS TERMINATED & THIS
        PROGRAM WAITS FOR ANOTHER PROGRAM TO ABORT
```

The **DI** command is used to print the program on an output device for later analysis. The printout LU is optional. If not supplied, the printout LU given as the second parameter of the schedule request will be used.

The **AD** command is used to set up for use with DBUGR. **ADDR** is the address you wish to examine. The program sets up one of the map registers it doesn't use to contain the physical page number corresponding to the aborting program's logical address given by the address specified. 'ABDMP' reports the first and last logical address of the aborting program which can be examined, and the locations at which they may be viewed ('ABDMP' copies one page from the aborting program's partition, so that DBUGR may be called. This allows only one page at a time to be viewed). You should then enter DBUGR (**DB** command), define a symbol to be the address used to access the actual address you want, and reference all data relative to this point. To access data on a different page, exit DBUGR (escape **P**), enter the "examine aborted program" command phase, set up a new address with another **AD** command, and re-enter DBUGR. All symbols you defined will still be defined.

The PARSE routine (described in the RTE Reference Manual) is used for decoding all input, except for those commands entered directly to DBUGR. Its rules determine syntax. Note that octal addresses must be suffixed by "**B**".

The program requires at least one extra page of memory address space to be available when it runs. It uses this extra page of address space for accessing memory in other partitions. That is, if the program requires 9 pages, it will run in a 9-page partition, but it cannot be run if loaded at any address higher than 60002(8).

The 'SPATC' module must be generated into the system in Table Area I. All other modules are part of the main program, 'ABDMP'. Priority is not critical, but the program type must be 3.

OPERATING SYSTEMS

***** PRINTOUT FORMAT *****

If you elect instead to print the program for later analysis, the format is as follows:

BASE PAGE

```
<addr> 00000 00000 00000 00000 00000 00000 00000 00000 aaaaaaaaaaaaaaaaaa
<addr> 00000 00000 00000 00000 00000 00000 00000 00000 aaaaaaaaaaaaaaaaaa
<addr> 00000 00000 00000 00000 00000 00000 00000 00000 aaaaaaaaaaaaaaaaaa
<addr> 00000 00000 00000 00000 00000 00000 00000 00000 aaaaaaaaaaaaaaaaaa
.
.
.
.
etc.
```

INDEX REGISTERS(X,Y)

```
00000 00000 PROGRAM

<addr> 00000 00000 00000 00000 00000 00000 00000 00000 aaaaaaaaaaaaaaaaaa
<addr> 00000 00000 00000 00000 00000 00000 00000 00000 aaaaaaaaaaaaaaaaaa
<addr> 00000 00000 00000 00000 00000 00000 00000 00000 aaaaaaaaaaaaaaaaaa
<addr> 00000 00000 00000 00000 00000 00000 00000 00000 aaaaaaaaaaaaaaaaaa
<addr> 00000 00000 00000 00000 00000 00000 00000 00000 aaaaaaaaaaaaaaaaaa
.
.
.
.
etc.
```

SAMPLE EXECUTION

Schedule program:

```
:RU,ABDMP
```

Program signals you:

ABORTING PROGRAM DUMPER

Program asks for directions:

```
COMMAND?(??=HELP)
```

```
LI TEST
COMMAND?(??=HELP)
EX
```

Operator asks for list of programs "armed" for freezing.
Sample program "TEST"
is only program in list. Program asks for new command.
Operator terminates.

OPERATING SYSTEMS

Note: 'ABDMP' now continues to scan ID segments at 10-second intervals, searching for a "frozen" program. You may abort it at this time, since there are no "frozen" programs as yet. When RTE RTE does abort a program, 'SPATC' will be sure 'ABDMP' is scheduled. Thereafter, it may not always be advisable to abort 'ABDMP', particularly if programs are being aborted frequently. However, the worst that can happen is that a program will be aborted and 'ABDMP' will be aborted before it detects the program. The very next program to be aborted will cause 'ABDMP' to be re-scheduled, and it will find both "frozen" programs (assuming there remains a third partition in which to run 'ABDMP' — two will be tied up in this case).

Operator causes sample program TEST to generate a memory-protect violation (see TEST's listing)

07>BR,TEST

'ABDMP' detects program "TEST" has been "frozen".

```
PROGRAM TEST ABORTED, MP      30006  PRIORITY  5000,
PRIMARY ENTRY POINT 30002,TYPE  3
A=177777, B=  10000, E/O= 77776
LOW MAIN= 30000 HIGH MAIN + 1= 31122
LOW BP =      2, HIGH BP + 1 =   13
PARTITION #      3, 1ST PG#=  81, SIZE=   22
EXAMINE ABORTED PROGRAM      Operator is asked to set up map, etc.
COMMAND?
AD,30000B
```

Operator sets up map to examine first program page.
Note that address is specified in octal, with "B"
suffix.
'ABDMP' responds:

```
PHYS.PG#  82  LOGICAL ADDRESS 30000 THROUGH  31777 CAN BE ACCESSED
STARTING AT LOCN 33346 THRU  35345
EXAMINE ABORTED PROGRAM
COMMAND?
```

```
DB
START DBUGR
```

Operator enters DBUGR.
DBUGR responds.

Note that DBUGR has its own syntax rules, different from PARSE and everything else in RTE. Operator sets symbol "P" to start of page for examining aborting program. First location is examined, and found to contain a JSB instruction.

```
33346<P:  P/  JSB 10,I
```

Next locations are examined using Control-J (linefeed).

```
P+1/  IOR 5   =30005  P+2/  IOR 35   =30035
P+3/  SSA   =2020
P+4/  HLT 77   =102077
```

(Note that the memory-protect violation instruction appears to be 2 locations ahead of the address reported by RTE. This will be true of all addresses in the first program page, because RTE begins loading the program two locations higher than the page boundary, in order to save space to save the X and Y-register. 'ABDMP' does not correct for this.

```
P+5/  LDX
P+6/  IOR 1070 =31070
P+7/  LDY  \P
END DBUGR
```

Operator exits DBUGR by hitting Escape P.
DBUGR responds.

OPERATING SYSTEMS

EXAMINE ABORTED PROGRAM
COMMAND?
EX

'ABDMP' asks for new set-up commands.

Operator is finished.

TEST ABORTED

'ABDMP' finishes up the "abort" process on the progr by sending an "OF,<program name>,1" message to the system message processor. Its partition and all other resources are released

Operator examines status of active programs.

07>ON,WHZAT

8:42:24:510

```
*****  
PT SZ PRGRM,T ,PRIOR*DRMT*SCHD*I/O *WAIT*MEMY*DISC*OPER * NEXT TIME *  
*****  
0 ** R$PN$*1 *00010 ***** 1  
0 ** WHZAT*1 *00001 ***** 1  
1 7 FMG07*3 *00090 ***** 3,ABDMP  
2 8 ABDMP*3 *00030 0 ***** 8:42:31:450  
*****  
DOWN LU'S  
*****  
DOWN EQT'S  
*****  
8:42:24:580
```

Note that 'ABDMP' is in the time list, but TEST has been aborted.

WARNING WARNING WARNING WARNING WARNING WARNING WARNING

WARNING WARNING WARNING WARNING WARNING WARNING WARNING

This routine contains no protection against "phantom ID segments", that is, ID segments which are created by FMGR when a type 6 (saved program) file is run. These are blanked after the program completes. When the program is run again, it may appear in another ID segment, not in the list kept by 'SPATC'.

When a program which is in 'ABDMP's list is aborted, it is frozen in its partition. It is therefore advisable to examine the program quickly, then allow it to be aborted. Dumping the program's partition contents to a high-speed device may be the only realistic way to avoid tying up a valuable partition for interactive analysis, but this hampers you in that instructions are not decoded for you.

If a number of programs are aborted at once, the fact that they are all frozen in different partitions may prevent 'ABDMP' from running. In this event, it may be advisable to modify the source so that a copy of the program is "canned": that is, stored in a disc file immediately. A separate program may be used to analyze the program using DBUGR. This second program can be cannibalized from parts of 'ABDMP'.

OPERATING SYSTEMS



The code is now in the Contributed Library, under the name 'ABDMP'.

LIST OF PROGRAM MODULES

&ABDMP (FORTRAN) MAIN PROGRAM
&CMDIN (FORTRAN) COMMAND INPUT INTERPRETER (SUBROUTINE)
&ABRSV (ASSEMBLY) PROGRAM LISTER, ENTRY & DELETE ROUTINES
&DMPRN (ASSEMBLY) SUBROUTINE FOR PHYSICAL MEMORY PRINTING
&SPATC ASSEMBLY CODE FOR SYSTEM PATCH(LOADED WITH SYSTEM AT GEN TIME)
&MSGFR (ASSEMBLY) OUTPUTS CANNED MESSAGES FOR FORTRAN (FRMTR ELIMINATOR)
&&TEST (ASSEMBLY) TEST PROGRAM

You may not ever reach the breakpoint. Although exasperating, this, too, is helpful information. Move it somewhere else. Try again. Just be sure that you know in advance what you'll be looking for when you get there, or you may reach a breakpoint, then realize this tells you absolutely nothing.

There are utilities for dumping an entire system at the point of a crash, and printing the results after you boot the system up again (these almost always require a mag tape, by the way). The prospect of examining 32K or more of octal printout may not appeal to everyone, but you can usually focus pretty quickly upon what is happening simply by going through the system tables. If you know what they're supposed to look like, you can tell which one(s) have been trampled upon. One such dump will leave too many possibilities, so don't expect too much. You get your most valuable information from repeated dumps, just like moving pictures, one for each crash, perhaps even one or two while the system is working for comparison. You will then be able to reduce the possibilities considerably, although not necessarily to any kind of definitive answer yet. For example, you may find that the problem occurs whether or not certain other programs are scheduled, whether or not certain drivers are active, whether or not SAM is relatively empty or nearly fully-allocated, and so on. All of this is useful information because it helps to eliminate possibilities.

Another technique which is sometimes useful is to "patch" a frequently-entered RTE system routine, such as \$IRT or \$XEQ or \$CIC (you'll need to be VERY careful on this one) so that you halt if the condition you're looking for has occurred. You can then list all programs in the active lists, determine their states, and so on, by going through the system lists. Since the system map will be enabled in this case, you don't have to worry about it. This technique has limited use because the state of the system may have changed too much to know what it was upon entry to \$IRT, and it certainly requires a thorough understanding of RTE.

A very useful, very quick way to locate mapping problems is to look at the maps (system, user and the two DCPC maps). These should have the same values for pages 1 through about 3 or so, depending upon the system. There should be a discontinuity in the user map at the page corresponding to the user program's load address. Page 0 should contain the physical page number immediately preceding this point. In RTE-IV, there will be another discontinuity corresponding to the driver partition. You can determine the particular driver which was being executed from the EQT pointers on the base page.

For some problems, particularly the kind which don't seem to make sense, aren't repeatable, or the failure shows up in a number of different ways, it can be helpful to define the boundaries of the problem, that is, the various circumstances under which it is known to occur. Ask yourself questions like, "What is the same in all cases? Does the system need to be heavily loaded, or at least some particular resources such as SAM, DCPC, etc., or does it occur for various loadings? What programs are always present? Did the problem suddenly show up? Was the system working under these conditions before some change was introduced (never mind that you KNOW that this change couldn't possibly affect the problem. Very often, it turns out that there is an obscure dependence)? Try to isolate it into the simplest possible set of conditions, and then ask yourself what could cause such a failure under these circumstances?" Do what you can do to eliminate as many of these possibilities as possible.

For some problems, particularly applications where a transaction enters the system and undergoes a series of processing stages, it can be helpful to add an "audit trail," that is, every transaction is written out to disc or mag tape, complete with time-of-day, the name of the program or subroutine and the state of the transaction. You then trace backwards to find the one(s) which weren't processed properly, and then generate the same transaction types to determine why they went wrong, narrowing in on the problem and printing more helpful information with each pass. It will not be helpful to print everything about every transaction, you'll be unable to see the problem for all the printout. This type of debugging tool may prove useful many times, even well after the system has been "commissioned," so it pays to keep it around and do a good job of documenting it.

OPERATING SYSTEMS

You may find it useful to enlist the aid of hardware devices to help you. If communications links are involved, you may want to use a communications analyzer (the usefulness of these devices is limited to communications links which use a standard protocol, so determine this first). A hardware device which checks the memory addresses and halts the computer when a certain address is accessed is called a "hardware breakpoint", which you may be able to wire up quickly if you understand the computer well enough.

There are no generalities here. This is, for the most part, unblazed trail. Many people have passed this way, but they've all gone in different directions without marking the paths. It's important to realize why this is so, because it's a clue to the state of mind of those who've gone before, and likely yours when it happens to you. You're really tuned into the problem, often not consciously aware of why you take each step, but looking back later you see the progression was strikingly rational. This is by way of saying that there seem to be no "cookbook" procedures yet. The most successful practitioners are artists, not scientists, and they work on a level which is a curious blend of instinct and scientific training. Therefore, don't be worried if you feel you don't know how to proceed when something like this happens. Just use your head, dig in and get started. You won't find the answer any other way. You're only really defeated when you start believing in magic.

GENERATING A MINIMAL RTE II SYSTEM

*by John Blommers/Gerald Green
Defence Research Establishment Pacific
Victoria, British Columbia*

From time to time, the RTE II user has been faced with the LOADR error L03 - memory overflow. This forced the user to reduce the size of this program or to restructure it using overlays. Neither alternative is particularly satisfying. Another choice is to generate a system that takes less memory, leaving more space for programs. Of course, you will sacrifice some features when you do this.

To produce a minimal RTE II system, perform some or all of the following steps, depending upon requirements:

1. Remove all drivers not required, but keep DVR00 (or DVR05) and DVR31 (or DVR32, according to disc type).
2. Delete all EQT entries not needed, but keep one for the console and one for the disc.
3. Remove the power-fail software DVP43 and AUTOR. Note that AUTOR can be stripped down so that it merely reports a simple "POWER FAILED" message using an EXEC call instead of a FORTRAN WRITE using the formatter (as it currently does).
4. Minimize the number of ID segments (short and long), number of I/O classes, number of resource numbers and number of LU Mappings..
5. Keep the number of entries in the device reference table as small as possible (saves a few words).
6. Move all I/O devices to be used to high priority slots, freeing more base page. Be sure no devices corresponding to undeclared trap cells can interrupt.
7. Delete PRMPT and R\$PN\$, thus eliminating the Multi-Terminal Monitor capability.
8. Delete the spooling modules: SMP, SPOUT, JOB, GASP, EXTND and DVS43.
9. Make D.RTR background disc resident. Thus, there is no foreground partition at all. Only the background partition remains. Eliminate \$\$CMD.
10. While generating the minimal RTE II, specify LINKS IN BASE while relocating the operating system, and LINKS IN CURRENT for the background programs. It turns out that (in the minimal system) the lack of base page is not a problem. By specifying base page links for the system, main memory is maximized. Each program, of course, reuses the same area of base page.
11. Load only LOADR, FMGR and D.RTR at generation time and leave one ID segment available for user programs. If the user suddenly needs an ID segment, he may SP the program holding the ID segment and OF the program to free it.
12. Some System Available Memory (SAM) must be allowed (say 500 words) between the system and the background partition. Respond with a zero to all generator questions regarding moving boundaries except for the last such question.
13. Eliminate all buffered I/O and don't use CLASS I/O, since these require SAM. Do this at generation time because without \$\$CMD, the LU and EQ commands will not be recognized.

By using all of the techniques described above, it is possible to obtain a background boundary of 24000B with base page links starting at 543B and a SAM of 1K. This is approximately 22 pages of main memory in a 32K computer, more than twice the usual size in a typical system.

OPERATING SYSTEMS

SPOOLING IS EASY WITH HIGH-LEVEL INTERFACE

by Jim Bridges

This article was suggested by a memo sent to us by T.D. Chase and P.K. Stock of Bell Labs in Holmdel, New Jersey. The concepts expressed have since been incorporated into a much more complex piece of software in use at Bell.

The spool subsystem of the HP-1000 disc-based systems provides a means to associate a disc file with a logical unit (LU) which can be treated as if it were, for example, a magnetic tape drive. The disc file can then be handled as if it were a file on magnetic tape. All the usual magnetic tape commands may be used: rewind, backspace, etc. Once the association (or link) between the LU and the file is set up, the somewhat complex procedure of disc file handling is traded for the simpler orientation of reading and writing from/to magnetic tape.

HP provides two methods for establishing the link between a file and a spool LU:

1. The program may be run under batch mode, in which case the use of spooling is transparent. That is, the association between the file and the spool LU is made outside the program.
2. The program may deliberately make calls to access the spool subsystem. The association between the file and the spool LU is made deliberately by the program in order to simplify programming.

This article makes a contribution in both cases. It extends the capability offered in batch mode to any copy of FMGR operating in conversational mode and it provides a simpler calling sequence to the Spool Monitor Program (SMP) provided by HP.

In order for the procedures described below to work properly, the system manager must have initialized the spool subsystem with GASP.

Transparent Operation

Let's take, for example, the use of the assembler in an RTE II system. We would like to input source from a file and output the relocatable to a file without using the LS (logical source) or the LG (load and go) areas of the disc. Therefore, we need two spool LUs. To automate the procedure we will use a transfer file as follows:

```
:RU,SGET,&PRDG::254
:CA,8,1P
:IF,1P,LE,0,5
:RU,SGET%IPRDG::254
:CA,9,1P
:IF,1P,LE,0,2
:RU,ASMB,8G,0G,9G
:IF,,EQ,,1
:DP,SPDDL FAILURE
:RU,KS,8G
:RU,KS,9G
::
```

The program SGET associates the file with a spool LU (sets up the linkage) and program KS releases the spool LU. If the spool set-up is successful, the number passed back in global 1P is the spool LU. To release the spool LU, the same number is passed to program KS. In between the two program calls, the associated LU may be used to access the file as if it were on a magnetic tape.

In a conversational mode, for example,

```
:RU,SGET,&PRDG::254
:DP,1P
```

<spool LU number displayed here>

OPERATING SYSTEMS

Spooling from FORTRAN Programs

The assembly language subroutine shown at the end of this article has two entry points which may be called from FORTRAN programs. To associate a file with a spool LU,

CALL FILE (ILU,INAM,ISC,ICR,IDISP)

ILU = Spool LU returned by subroutine
INAM = File name to be associated with spool LU
ISC = Optional file security code
ICR = Optional cartridge reference number
IDISP = Optional disposition flag (default = 23B)

To release a spool LU,

CALL FDONE (ILU)

ILU = Spool LU assigned by SGET

The default setting of the bits in IDISP is exactly the same as used by the program SGET.

For example, suppose we want to write a program which uses file "INFO" for output and references it using formatted WRITE statements in FORTRAN. The outline of the program would be:

```
FTN4,L      .....  
            DIMENSION INAME(3)  
            DATA INAME/2H"I,2HNF,2HD /  
            .....  
100         CALL FILE (LU,INAME)  
            .....  
200         WRITE (LU,1000) A,B,C  
1000        FORMAT ("A,B,C="3F6.2)  
            .....  
300         CALL FDONE (LU)  
            .....  
            END  
            END$
```

Conclusion

The programs SGET, KS and the FORTRAN-callable subroutines FILE and FDONE provide a high-level interface to the HP-1000 spool subsystem. They provide an additional level of simplicity which should encourage all programmers to make use of spooling, since little knowledge is required.

```
FTN4,L  
PROGRAM SGET(3,75),SET-UP SPOOL LU TO A FILE 4-28-78  
INTEGER IB(16),IT(5),IBUF(20),IPBUF(10)  
DATA IB/7*0,23B,23B,1,6*0/  
DATA IPBUF/10*0/  
DATA IT/-1,4*0/  
DATA ISTRC/1/
```

OPERATING SYSTEMS

```
C
C SETS UP A SPOOL LU TO A FILE. FILE IS BOTH READ & WRITE
C FOR DVR23. ON,SGET,<NAMR> PASSES BACK SPOOL LU IN GLOBAL 10G
C
C :RU,SGET,<NAMR>
C :DP,1P          (WILL SHOW SPOOL LU ASSIGNED, IF ANY)
C
```

```
CALL GETST (IBUF,-40,ICNT)
IF (NAMR(IPBUF,IBUF,ICNT,ISTRC)) 99,10
10  ITYPE = IAND (IPBUF(4),3)
    IF (ITYPE.NE.3) GO TO 89
    IB (3) = IPBUF (1)
    IB (4) = IPBUF (2)
    IB (5) = IPBUF (3)
    IB (6) = IPBUF (5)
    IB (7) = IPBUF (6)
```

```
C
C CHECK TO SEE IF USING FORM :RU,SGET,NAME,SC,CR
C
    IF ((IB(6).NE.0).OR.(IB(7).NE.0)) GO TO 80
    IF (NAMR(IPBUF,IBUF,ICNT,ISTRC)) 80,30
```

```
C
C IF NO MORE TO PARSE, ASSUME NAMR
C
30  IB (6) = IPBUF (1)
    IF (NAMR(IPBUF,IBUF,ICNT,ISTRC)) 80,40
40  IB (7) = IPBUF (1)
80  CALL SPOPN (IB,IT)
    IT (4) = IB (6)
    IT (5) = IB (7)
99  CALL PRTN (IT)
100 CALL EXEC (6)
89  IT = IT - 1
    GO TO 99
    END
    END$
```

```
FTN4,L
PROGRAM KS(3,75),RELEASE SPOOL LU ASSIGNED BY SGET 4-28-78
C
C :RU,KS,<LU>          (TO RELEASE SPOOL LU ASSIGNED BY SGET)
C
    INTEGER IT(5),IS(3)
    DATA IS/2HSM,2HP ,2H /
    CALL RMPAR (IT)
    CALL EXEC (23,IS,4,IT)
    END
    END$
```

OPERATING SYSTEMS

ASMB,R,L,B

NAM FPACK,7 PROGRAM INTERFACE TO SPOOLING 4-28-78

* THESE ROUTINES PROVIDE A PROGRAM INTERFACE TO THE SPOOL SYSTEM
* THERE ARE TWO ENTRY POINTS:

*
* 1. CALL FILE (ILU,INAM,ISC,ICR,([IDISP])
* ILU = SPOOL LU RETURNED BY SUBROUTINE
* INAM = FILE NAME TO BE ASSOCIATED WITH SPOOL LU
* ISC = OPTIONAL FILE SECURITY CODE
* ICR = OPTIONAL CARTRIDGE REFERENCE NUMBER
* IDISP = OPTIONAL DISPOSTION FLAG (DEFAULT = 23B)

*
* BIT MEANING 0/1
* 0 PURGE FILE/SAVE FILE
* 1 NO HOLD/HOLD
* 2 ALWAYS ZERO
* 3 USER FILE/SPOOL POOL FILE
* 4 HEADERS/STANDARD FILE FORMAT

*
* BITS 8 & 7
* 0 0 READ & WRITE
* 0 1 READ ONLY
* 1 0 WRITE ONLY

*
* 2. CALL FDONE (ILU)
* ILU = SPOOL LU ASSIGNED

*
* THUS, FILE ASSIGNS THE SPOOL LU AND FDONE RELEASES IT.
*

EXT SPOPN,.ENTR,EXEC
ENT FILE,FDONE
SUP

A EQU 0 A REGISTER
B EQU 1 B REGISTER
BUFFR OCT 0,0
BNAM ASC 3,
BSC DEC 0
BCR DEC 0
OCT 23
BDIS OCT 23
DEC 1,0,0,0,0,0,0

*
* SET UP FILE
*

LU NOP SPOOL LU TO BE PASSED BASK
NAME NOP FILE NAME
SC NOP SECURITY CODE
CR NOP CARTRIDGE REFERENCE
DISP NOP DISPOSITION WORD
*

OPERATING SYSTEMS

```
FILE  NOP
      JSB .ENTR
      DEF LU
      DLD NAME,I
      DST BNAM
      ISZ NAME
      DLD NAME,I
      STB BNAM+2

*
*  OPTIONAL PARAMETERS
*
      CLB
      LDA SC
      STB SC          CLEAR FOR NEXT TIME

*
*  WATCH OUT FOR THIS CODE! THE A CONTAINS THE ADDRESS IF THE
*  PARAMETER WAS PASSED OR IT CONTAINS A ZERO. IN EITHER CASE,
*  THE CODE IS VALID.
*
      LDA A,I
      STA BSC          SET SC OR DEFAULT 0

*
      LDA CR
      LDA A,I
      STA BCR          SET CR OR DEFAULT 0

*
      LDA DISP
      STB DISP
      SZA,RSS
      JMP .10          DEFAULT TO 23B

*
      LDA A,I
      STA BDIS
      JMP .20

*
.10   LDA =B23
      STA BDIS

.20   JSB SPOPN        CALL SPOOL OPEN ROUTINE
      DEF **3
      DEF BUFR
      DEF LU,I
      JMP FILE,I

*
JLU   NOP
FDONE NOP
      JSB .ENTR
      DEF JLU
      JSB EXEC          RELEASE SPOOL LU
      DEF **5
      DEF D23
      DEF SMP
      DEF D4
      DEF JLU,I
      JMP FDONE,I

*
D23  DEC 23
SMP  ASC 3,SMP
D4   DEC 4
      END
      END*
```


OPERATING SYSTEMS

SAVE TIME & EFFORT IN GENERATING YOUR FIRST RTE-M BASIC SYSTEM

by Todd Field

In some circles, generating an RTE-M system with BASIC is considered a good example of a Catch-22 situation. Before you can generate BASIC into the system, you need to run RTMTG to generate a user Branch and Mnemonic (B&M) Table to satisfy the BASIC interpreter's externals. Before you can run RTMTG to create the B&M tables, you need a system. Hence, the catch. Although the procedure of loading and running RTMTG is not particularly difficult nor time consuming, if you have an RTE-II, RTE-III or RTE-IV system with the assembler available, you can save this step.

To see how the assembler can save you time, consider the purpose and format of the B&M tables. A program needs to be linked to all its subroutines before execution if the time consuming linking operation is to be avoided at run-time. The BASIC interpreter itself obviously cannot know what subroutines a user program will call, but the B&M table, as it is user defined, can. What the B&M tables are is a list of subroutine names as they are called for in the BASIC programs, with their starting addresses and linkage information. The format of this file is relocatable binary, the same as is produced by the assembler or any other compiler.

The description of RTMTG in the RTE-M generator manual and a quick glance at the source code for RTMTG give one all the information one needs to construct one's B&M tables without the use of RTMTG.

```
ASMB,L
*****
*
*      GENERAL FORM OF AN ASSEMBLER ROUTINE TO PRODUCE
*      UBMTBL, THE BASIC/1000M BRANCH AND MNEMONIC TABLES
*
*****
*
*      NAM BMTBL,7          USER GENERATED BRANCH & MNEMONIC TABLE
*
*      EXT subr1,subrn, ... ,subrn      *      ONE ENTRY PER SUBROUTINE
*
*****
*      BRANCH TABLE
*****
*
*      ENT  BRTBL
BRTBL DEF  **1
*
*      DEF  subr1          *      first subroutine
*      OCT  0n....n      *      n=0 SIMPLE VARIABLE  n=1 ARRAY
*      OCT  0k....k      *      k=0 TO SUBROUTINE    k=1 FROM
*      OCT  xm....m      *      x=0 REAL FUNCTION    x=1 INTEGER
*
*
*      *      m=1 INTEGER VARIABLE m=0 REAL
*
*      *      in the above 3 words, N,K and M are bits.
*
*      *      for n,k and m, bit 0 is var#1,
*
*      *      bit 2 is var#2,....,bit 14 is var#15.
*
*
*      . . .          *      repeat for every subroutine
*
*
*      DEF  subrn
*      OCT  0n....n
*      OCT  0k....k
*      OCT  xm....m
*
```

OPERATING SYSTEMS

```

*****
*   MNEMONIC TABLE   *
*****
*
*   ENT MNTBL
MNTBL DEC  -n          *   n = NUMBER OF SUBROUTINES
*
*   OCT  m0..0aaaaabbbb *   m=0 SUBROUTINE      m=1 FUNCTION
*                       *   aaaa = PARAMETER COUNT in bits 4-7
*                       *   bbbb = CHARACTER COUNT IN NAME in bits 0-3
*   ASC  3,subr1(      *   left parenthesis terminates subroutine name
*
*   . . .              *   repeat for every subroutine
*
*   OCT  m0..0aaaaabbbb
*   ASC  3,subrn(
*
*   END
*   END$

```

The listing above shows the general form of an assembler program to produce the B&M tables. The EXT's are all the subroutines to be included in the table. The two ENT's are called for by the BASIC interpreter. Words two, three and four of each branch table entry contain the necessary information to link the variables in the BASIC program to the variables in the subroutine. Word one of each branch table entry will contain, after loading, the address of the subroutine. Words two, three and four of each entry in the mnemonic table contain the name of the subroutine. Word one of each mnemonic table entry contains additional linking information. Note that the left parenthesis must immediately follow the subroutine name in the mnemonic table, and that the entry must be padded with blanks to 6 ASCII characters.

After the table has been assembled, the relocatable output can be transported to the RTE-M system on cassette, to be used in the generation process.

Below are B&M entries for the table generator RTMTG for sample subroutines SMART and STUPD. Immediately below is the same table produced according to the procedure described in this article.

```

OR(I,I),INTG,ENT=BIOR
HPIB(I,I,I)

0001          ASMB,L
0002*****
0003*
0004*   SAMPLE ASSEMBLER ROUTINE TO PRODUCE UBMTBL FOR TWO SUBROUTINES *
0005*
0006*****
0007*
0008 00000          NAM BMTBL,7  USER GENERATED BRANCH & MNEMONIC TABLE
0009*
0010          EXT BIOR,HPIB
0011*

```

OPERATING SYSTEMS

```
0012*****
0013*      BRANCH TABLE                                     *
0014*****
0015*
0016                      ENT BRTBL
0017 00000 000001R BRTBL DEF  **1
0018*
0019 00001 000001X      DEF BIOR      *   FIRST SUBROUTINE
0020 00002 000000      OCT 0          *   ALL SIMPLE VARIABLES
0021 00003 000000      OCT 0          *   NO VARIABLES RETURNED
0022 00004 100003      OCT 100003    *   INTEGER FUNCTION
0023*                      *   INTEGER VARIABLES
0024*
0025 00005 000002X      DEF HPIB      *   SECOND SUBROUTINE
0026 00006 000000      OCT 0          *   ALL SIMPLE VARIABLES
0027 00007 000000      OCT 0          *   NO VARIABLES RETURNED
0028 00010 000007      OCT 7          *   ALL INTEGER VARIABLES
0029*
0030*****
0031*      MNEMONIC TABLE                                   *
0032*****
0033*
0034                      ENT MNTBL
0035 00011 177776 MNTBL DEC  -2      *   2 SUBROUTINES
0036*
0037 00012 100043      OCT 100043    *   FUNCTION WITH 2 PARAMETERS
0038 00013 047522      ASC 3,DR(     *   LEFT PAREN TERMINATES NAME
      00014 024040
      00015 020040
0039*
0040 00016 000065      OCT 65        *   SUBROUTINE WITH 3 PARAMETERS
0041 00017 044120      ASC 3,HPIB(
      00020 044502
      00021 024040
0042*
0043                      END
```

COMPUTATIONS

MICROCODED FAST FOURIER TRANSFORM FOR E-SERIES COMPUTERS

by Glenn Talbot

If you occasionally need to use the tools of Digital Signal Analysis like System Transfer Function, Auto Power Spectrum, Cross Correlation, etc., you probably know that the key to making these calculations is the Fast Fourier Transform. You may have looked into a Dedicated FFT Computer (Fourier Analyzer) and decided that you don't need it often enough to justify that kind of money, but you are still sitting around twiddling your thumbs while your "Fast" Fourier Transform written in FORTRAN executes. Then the microcode-enhanced FFT that was just updated to run on your E-Series computer may be just what you are looking for!

This package, available from the Data Systems LOCUS Contributed Library, consists of the Microcode, an assembly language interface program, and a FORTRAN test/example program to verify the operation of your FFT and give you some examples to help you get started with your own Digital Signal Analysis programs.

To use your FFT package in your RTE Operating system on an E-Series Computer you will need a 1K WCS board (13197A) and DVR36 generated into your system. You also need the Microassembler which is part of the 92061A RTE Microprogramming Package.

When you have all of these things together in your system you need to Microassemble the two Microprograms and load them into WCS. They only use 512 words of your 1K WCS so there is plenty of room left over for further Microprograms. Then assemble the assembly language interface program, compile the FORTRAN test program, and load these two together and try out your FFT.

To transform your own data in an applications program you will first have to get your data into an INTEGER array. This is an INTEGER FFT with automatic rescaling on overflow. Then generate a SIN-COS look-up table; you may use the code from the test program for this. Then execute the FFT with simple calls to the interface program entry points.

Order your FFT package from your local HP Sales Office, or (if you are in the U.S.) you can use the Direct Mail Order form in the back of the COMMUNICATOR 1000. The part number is 22682-13395 for the minicartridge media option. For other options consult your local HP Sales Office.

If you want more information about the algorithms used, see the article in IEEE Transactions on Audio and Electroacoustics, Vol. AU-15, June 1967, pages 45-55 and "The Fast Fourier Transform Algorithm and its Applications," IBM Research Paper RC-1743, February 9, 1967, pages 30-33.

If you want to do Digital Signal Analysis but feel you need to know more about it before you can get started, read the HP Application Note 240-0, Digital Signal Analysis, Time and Frequency Domain Measurements. You may find that you could apply this at some point in your operation.



PLOTTING ON THE 9871A PRINTER THROUGH A 264X TERMINAL

by Larry Dwyer

The 9871A has the capability to do plotting as well as printing. The paper can be fed in a backward and forward motion. Plotting is a secondary function, however. For example, the resolution is different in the X and Y directions (1/120 inch in X, 1/96 inch in Y). Also, plotting is done via a series of unconnected dots rather than an unbroken line. Nevertheless, it would be nice to do "crude" plots on the 9871A - especially if no other plotter is available.

Some customers have inquired about the possibilities of using DVR05 (or DVA05) to control plotting on the 13349A subsystem (9871A connected to the terminal, accessed as subchannel 4). The data sheet for the RTE drivers package (5953-0861, page 4-5) specifically excludes support for this configuration. Nevertheless, the possibility has been investigated and there are ways to "get the job done".

The basic problem is that the driver always transmits a carriage return/line feed to a device on subchannel 4 (assumed to be a printer, not a plotter). In other words, "honesty mode" (transmit only what is in the buffer) does not apply to any subchannel other than zero (an interactive terminal). This is true whether you use the "honesty" bit in the control word or simply put a back arrow as the last character in the buffer.

A method of circumventing the problem is to go through a subroutine which repeats the last command ("un-doing" the carriage return/line feed) prior to doing the current command. Using this technique, the program listed below plots a simple sine wave on the 9871A.

The subroutines SCALE and PLOT are general purpose and can be included in any user program to plot any graph desired. The full area of the 9871A can be used as the plot area (11 inches by 13.5 inches). When plotting, the paper must be fed using the platen, not the external paper tractor.

The subroutines SPUT and SMOVE are part of the Decimal String Arithmetic package (described in manual 02100-90140) and are included on the Grandfather Discs under the file %DECAR. They are also included as part of the IMAGE/1000 product, 92063A.

```
FTN4,L
PROGRAM PLOTT(3,75),PLOT SINE WAVE ON 9871A PRINTER W/ DVR05
C
C :RU,PLOTT,,<PRINTER LU>
C
C     INTEGER LUS(5),PRNTR
C     EQUIVALENCE (LUS(2),PRNTR)
C     CALL RMPAR (LUS)
C     IF (PRNTR.EQ.0) STOP
C
C SET SCALING VALUES
C
C     CALL SCALE (0.,12.,-1.1,1.1)
C
C PRESET STARTING POSITION
C
C     CALL PLOT (0.0,0.0,0,PRNTR)
C
C PLOT AXIS AND TIC MARKS
C
C     DO 100 I=0,60
C     X = FLOAT (I)/10.
C     CALL PLOT (X,0.,0,PRNTR)
C     CALL PLOT (X,.025,1,PRNTR)
```

COMPUTATIONS

```
100 CALL PLOT (X,0.,1,PRNTR)
    CALL PLOT (0.,0.,1,PRNTR)
C
C PLOT A SINE WAVE
C
    DO 1000 I=0,600,5
    X = FLOAT (I)/100.
    Y = SIN (X)
1000 CALL PLOT (X,Y,1,PRNTR)
    END
C
    SUBROUTINE PLOT (X,Y,IZ,PRNTR)
    INTEGER IOBUF(6),PRNTR
C
C CALLING PARAMETERS:
C
C     X = DESTINATION X VALUE TO PLOT
C     Y = DESTINATION Y VALUE TO PLOT
C     IZ = PEN UP/DOWN DURING PLOT (0/1)
C
C THE OUTPUT BUFFER CONTAINS TWO GROUPS OF DATA. THE FIRST IS THE
C LAST PLOTTED POSITION AND THE SECOND IS THE CURRENT PLOTTED
C POSITION. THE FIRST IS REQUIRED TO MOVE THE "PEN" TO THE LAST
C POSITION BEFORE PLOTTING TO THE NEXT POSITION. THIS IS NECESSARY
C BECAUSE THE DRIVER ALWAYS OUTPUTS A CARRIAGE RETURN/LINE FEED
C TO THE PRINTER AT THE END OF A BUFFER.
C
C     A = ABSOLUTE PEN UP
C     LOWER CASE A = ABSOLUTE PEN DOWN
C     R = RELATIVE PEN UP
C     LOWER CASE R = RELATIVE PEN DOWN
C     @ = DATA BYTE, ZERO VALUE
C
C     15501B = ESC A
C     15541B = ESC LOWER CASE A
C
    DATA IOBUF/15501B,2H@,2H@,15541B,2H@,2H@/
    DATA IUP/65/,IDN/97/
C
C SCALE X AND Y BETWEEN 0 AND 1, RETURN IN X2, Y2
C
    CALL XSCAL (X,X2,Y,Y2,0)
    IF (X2.GT.1.) X2 = 1.
    IF (Y2.GT.1.) Y2 = 1.
    IF (X2.LT.0.) X2 = 0.
    IF (Y2.LT.0.) Y2 = 0.
C
C SCALE X2,Y2 TO MAX PLOT AREA. FIX TO INTEGER VALUES
C
    IX = 1587. * X2
    IY = 1065. * Y2
C
C GENERATE PLOT DATA BYTES
```

COMPUTATIONS

```
C
  IC1 = 64 + (IX/64)
  IF (IC1.GT.126) IC1 = 126
  IC2 = 64 + IAND (IX,63)
  IF (IC2.GT.126) IC2 = 126
  IC3 = 64 + (IY/64)
  IF (IC3.GT.126) IC3 = 126
  IC4 = 64 + IAND (IY,63)
  IF (IC4.GT.126) IC4 = 126

C
C MOVE DATA BYTES IC1 THRU IC4 TO OUTPUT BUFFER CHARACTER POSITIONS
C 9 THROUGH 12.
C
  CALL SPUT (IOBUF,9,IC1)
  CALL SPUT (IOBUF,10,IC2)
  CALL SPUT (IOBUF,11,IC3)
  CALL SPUT (IOBUF,12,IC4)

C
C SET PLOT MODE TO ABSOLUTE, PEN UP. IF IZ#0, PEN DOWN.
C
  CALL SPUT (IOBUF,8,IUP)
  IF (IZ.NE.0) CALL SPUT (IOBUF,8,IDN)

C
C OUTPUT THE PLOT DATA
C
  CALL EXEC (2,PRNTR,IOBUF,6)

C
C SAVE THE CURRENT DATA FOR THE NEXT PLOT
C
  CALL SMOVE (IOBUF,9,12,IOBUF,3)
  RETURN
  END

C
  SUBROUTINE SCALE (X1,X2,Y1,Y2)
C
C THIS SUBROUTINE IS THE USER ENTRY POINT TO XSCAL. IT'S PURPOSE
C IS TO MAKE THE IFLAG PARAMETER IN XSCAL TRANSPARENT TO THE USER.
C
  CALL XSCAL (X1,X2,Y1,Y2,1)
  RETURN
  END

C
  SUBROUTINE XSCAL (X1,X2,Y1,Y2,IFLAG)
C
C THIS SUBROUTINE WILL EITHER STORE THE SCALE DATA AWAY (IFLAG=1)
C OR WILL USE THE SCALING INFORMATION TO RESCALE THE CURRENT DATA
C (IFLAG=0).
C
C THE SCALING DATA IS ENTERED WITH THE PARAMETERS:
C
C      X1 = MIN X VALUE EXPECTED
C      X2 = MAX X VALUE EXPECTED
C      Y1 = MIN Y VALUE EXPECTED
C      Y2 = MAX Y VALUE EXPECTED
C
```

COMPUTATIONS

```
C      IFIRST = FIRST TIME FLAG
C
C      DATA IFIRST/0/
C
C STORE OR SCALE ?
C
C      IF (IFLAG.EQ.0) GO TO 100
C
C STORE SCALING DATA
C
C      SX1 = X1
C      DX = X2 - X1
C      SY1 = Y1
C      DY = Y2 - Y1
C      IFIRST = 1
C      RETURN
C
C SCALE CURRENT DATA X1, Y1 USING STORED SCALING DATA
C RETURN SCALED VALUES IN X2, Y2.
C
C IF THIS IS FIRST CALL, ERROR ... STOP 13
C
100  IF (IFIRST.EQ.0) STOP 13
C      X2 = (X1-SX1)/DX
C      Y2 = (Y1-SY1)/DY
C      RETURN
C      END
C      END$
```


DATA ACQUISITION VIA HP 2313 SUBSYSTEM AT LOW SAMPLING RATES

John A. Danos
Rohm & Haas Research Laboratories
Bristol, Pennsylvania

[Editors note: John A. Danos is the winner of an HP-21 Scientific calculator for submitting this article to the HP-1000 Communicator. For details on how you can win a calculator, see the Editor's Desk column in this issue.]

Using the HP 2313 subsystem in RTE II to collect data from several instruments at low data rates (.1 to 1 point per second) may present several problems. These include trade-offs between writing a number of memory resident programs, one for each instrument, or having disc resident programs swapping at high rates. A method was developed to have a single memory resident program scan a number of analog inputs and write the data to a dummy driver. This driver buffers analog data and sample time. It can be read from any disc resident program which selects the analog inputs of interest for processing. Since communication between the program collecting and storing the data and programs using the data is through a driver, all activity is controlled by the system itself, without the need for hardshakes, class I/O, etcetera.

The method uses three or more "programs". These are DVR77 (the dummy driver), AINLC and ANSAV. Listings for all three are included at the end of this article. Program ANSAV is the application program. There would normally be several programs of this type; one for each instrument on line.

DVR77

This driver stores data that is written to it from the memory resident program AINLC. It is configured to accept 16 analog inputs and the time the sample is taken. Analog inputs 0-7 are only buffered for 1 second; analog inputs 8-15 are buffered for 20 seconds. For each of the 20 buffered data points, the off-set time in seconds (resolution of 1/100) is also stored. The driver allows a WRITE call only from a memory resident program. It accepts two types of READ calls, slow and fast. For slow reads, the driver returns the time of the last sample and the 0-7 analog inputs. For fast reads, the driver returns the time of the last sample, all data for the 8-15 analog inputs of the last 20 seconds, the off-set times and the stack pointer.

AINLC

This is a memory resident program that makes calls to the 2313 subsystem to acquire data from 16 analog inputs. It checks the status of the 2313 and will set the device "up" if it is "down". In addition to analog data, it takes a time reading. This information is then written to DVR77 for storage. AINLC is scheduled once every second and should have a high priority.

ANSAV

This is a background application program that is scheduled every 10 seconds. It will collect data from inputs 8-15 at sampling intervals from 1 to 32,767 seconds. When it collects the required number of points, they are listed to the terminal from which the program was run. This program reads fast data from DVR77 rotating buffers. This program can serve as a basis for writing, filing and data reduction programs.

Results:

We have successfully used this technique in acquiring data from several gel permeation and liquid chromatographs. The application programs are scheduled by external events and monitor instrument status. Collected data are stored in files for later processing. System memory requirements are small with DVR77 and AINLC using about 360 words, of which half are data. AINLC executes once per second and requires a few milliseconds actual execution time. The 2313 will be busy about 20 milliseconds every second. Call to DVR77 require 2-3 milliseconds to complete and are of the immediate completion type.

INSTRUMENTATION

Conclusion

We feel that this technique fills a gap in using the 2313 subsystem in the automation of low data rate laboratory instruments. It allows for great flexibility in handling data from a large number of different types of instruments, all of which are not on-line at any given time. Programs are easily developed for the different instruments with no worry of interference or need for frequent system generations.

[Editor's note: The technique used by AINLC to "up" the device EQT for the 2313 driver assumes the EQT is the first in memory, as indicated by the contents of base page 1650B. Also, the technique will work only if a pending interrupt actually occurs. Simply changing the "down" bit on the EQT will not re-initiate the request. A simple method of processing a "downed" EQT is to call the system library routine 'MESSS' to give the "up" command to the device. This routine is described in Part 5 of the RTE manual.]

```
FTN4,L
PROGRAM ANSAV,3,50
C
C
C THIS PROGRAM MAKES CALLS TO DVR77 TO OBTAIN FAST(RELATIVE) DATA BEING
C COLLECTED BY AINLC. IT CAN SELECT 1-400 POINTS FROM 1 OF 8 ANALOG
C INPUTS AT INTERVALS OF 1-32767 SECONDS. THE POINTS AND TIMES
C ARE LISTED WHEN THE LAST POINT IS ACQUIRED.THE PROGRAM IS SCHEDULED
C EVERY 10 SECONDS. THE CALL TO DVR77 GETS THE LAST 20 SECONDS OF BUFFERED
C DATA FROM THE 2313 ANALOG SYSTEM. THE BUFFER IS SCANNED AND SELECTED
C DATA IS STORED IN IDATA AT THE GIVEN RATE. SUBROUTINE MAIN IS RESPONSIBLE
C FOR SORTING DATA FROM DVR77'S ROTATING BUFFER. THIS PROGRAM IN ADDITION
C TO LISTING INPUT DATA CAN SERVE AS A MODEL FOR FILING PROGRAMS.
C
C
DIMENSION ITJ(5),IY(1),IX(8,20),IP(5),DATA(400)
COMMON IDATA(400),ETIME(400),IZ(186),J,TLAST,IT1(5),T1,IRATE,IN1
EQUIVALENCE(IZ(27),IX(1,1))
CALL RMPAR(IP)
LU=IP(1)
WRITE(LU,4000)
4000 FORMAT("TYPE IN INPUT#, RATE, # OF POINTS")
C
C AINLC IS CONFIGURED SO THAT INPUTS 1-8 REFER TO ANALOG INPUTS.
C 8-15 OF SLOT 4. THE RATE IS THE INTERVAL BETWEEN POINT IN SECONDS
C THE MAXIMUM NO OF POINTS IS 400( DIMENSION OF IDATA,ETIME,ETC )
READ(LU,*)IN1,IN2,IN3
CALL INITL
C
C THIS CALL WILL "INITIATE" THE DATA COLLECTION BY GETTING THE 1ST
C POINT AND THE TIME.
C
C
IRATE=IN2
DO 10 I=1,5
ITJ(I)=IZ(I)
10 CONTINUE
GO TO 1000
C GET DATA AND STORE
200 CALL MAIN
C TEST IF DONE
IF(J.GT.IN3) GO TO 300
1000 CALL WAIT(10,2,IERR)
GO TO 200
C ALL DATA IN POST WRITE TO LIST DEVICE
300 WRITE(LU,4001)
```

INSTRUMENTATION

```
4001  FORMAT("TIME OF FIRST POINT")
      WRITE(LU,3000)ITJ
      WRITE(LU,4002)
4002  FORMAT("DATA POINT  TIME")
      WRITE(LU,4003)
4003  FORMAT(" ")
      N4=IN3/5
      DO 60 I=1,IN3
      DATA(I)=IDATA(I)*.8/2048.
60    CONTINUE
      K=0
      DO 40 I=1,N4
      WRITE(LU,3001)(DATA(K+J),ETIME(K+J),J=1,5)
3000  FORMAT(5(I6,5X))
3001  FORMAT(5(F7.4,2X,F7.2))
      K=K+5
40    CONTINUE
2000  END
      SUBROUTINE INITL
      DIMENSION IX(8,20)
      COMMON IDATA(400),ETIME(400),IZ(186),J,TLAST,IT1(5),T1,IRATE,IN1
      EQUIVALENCE (IZ(27),IX(1,1))
      CALL EXEC(1,26,IZ,186)
C SET INITIAL TIME
C CONVERT 1ST DATA TIME TO REAL VALUE
      CALL TABS(IZ,T1)
      ETIME(1)=0.
      ISTK=IZ(6)
      IF(ISTK.EQ.0)ISTK=20
      IDATA(1)=IX(IN1,ISTK)/16
      TLAST=0.
      J=2
      RETURN
      END
      SUBROUTINE MAIN
      DIMENSION IDOR(20),TORD(20),IX(8,20)
      COMMON IDATA(400),ETIME(400),IZ(186),J,TLAST,IT1(5),T1,IRATE,IN1
      EQUIVALENCE (IZ(27),IX(1,1))
C GET DATA FROM DVR77
      CALL EXEC(1,26,IZ,186)
C CONVERT TIME TO REAL VALUE
      CALL TABS(IZ,TNOW)
      TNOW=TNOW-T1
      ISTK=IZ(6)
C REORDER TIME AND DATA FROM DVR77 BUFFER
      TTEST=IZ(1)/100.+IZ(2)
      DO 20 I=1,20
      IF(ISTK.EQ.0)ISTK=20
      IDOR(21-I)=IX(IN1,ISTK)
      TDIFF=TTEST-IZ(ISTK+6)/100.
      IF(TDIFF.LT.0.)TDIFF=TDIFF+60
      TORD(21-I)=TNOW-TDIFF
      ISTK=ISTK-1
20    CONTINUE
C STORE DATA AND TIME VALUES AT NTH INTERVALS
      I=0
31    I=I+1
      IF(I.GT.20)GO TO 34
      IF(TORD(I).GE.TLAST)GO TO 30
      GO TO 31
```

INSTRUMENTATION

```
30  TNEXT=TLAST+IRATE-.5
32  IF(TORD(I).GE.TNEXT)GO TO 33
    I=I+1
    IF(I.GT.20) GO TO 34
    GO TO 32
33  TLAST=TORD(I)
    IDATA(J)=IDOR(I)/16
    ETIME(J)=TORD(I)
    J=J+1
36  I=I+1
C TEST TO SEE IF ALL DATA IN
    IF(J.GT.400) GO TO 34
    IF(I.GT.20) GO TO 34
    GO TO 30
34  RETURN
    END
    SUBROUTINE TABS(ITIN,TO)
C
C THIS ROUTINE CONVERTS TIME ARRAYS INTO REAL NUMBERS.
C
    DIMENSION ITIN(1)
    TO=ITIN(1)*.01+ITIN(2)+ITIN(3)*60.+ITIN(4)*3600.
    RETURN
    END
    END$
```

ASMB,L

```
* THIS IS A CORE RESIDENT PROGRAM DESIGNED TO AQUIRE DATA USING THE
* 2313 ANALOG SUB SYSTEM AND STORE THIS DATA IN DVR77. IT COLLECTS DATA
* FROM 16 INPUTS, TAKES A TIME READING, AND UPS THE DAS IF IT IS DOWN.
* IT USES THE FOLLOWING CALL:
*
* CALL EXEC(2,LU,BURF,21)
*
* DATA IN BURF IS TIME(5),ANALOG 0-16(16). THE PROGRAM IS CONFIGURED TO
* SAMPLE DATA ONCE PER SECOND SEQUENTIALLY ON INPUTS 0-15,SYSTEM 0,
* SLOT 04.
    NAM AINLC,1,25,2,1
    ENT AINLC
    EXT EXEC,$LIBR,$LIBX
A EQU 0
B EQU 1
AINLC NOP
    JSB $LIBR CHECK 2313 AV AND UP IT IF DOWN
    NOP
    LDA =D64 BASE LINK PRINTER 1650B
    ADA 1650B
    LDB A,I A REG=ADDRESS OF WORD 5 EQTS
    ELB
    SEZ CK BUSY AND WAIT
    JMP CONT
    ELB
    SEZ,RSS CK AVAIL +DOWN
    JMP CONT
    CME A2313 IS DOUN
    ERB,ERB
    STB A,I RESTORE TO EQTA AV=0
```

INSTRUMENTATION

```
CONT JSB $LIBX
      DEF **1
      DEF **1
      JSB EXEC TAKE 16 A2313 READINGS
      DEF **5
      DEF COD02
      DEF CN18
      DEF QUE
      DEF N6
      JSB EXEC TAKE TIME READING
      DEF **4
      DEF COD11
      DEF TIME
      DEF YEAR
      JSB EXEC DUMP DATA TO DVR80
      DEF **5
      DEF COD02
      DEF CN26
      DEF BUFR
      DEF N21
      JSB EXEC CALL EXIT
      DEF **3
      DEF N6
      DEF NUM0
      NOP
COD02 DEC 2
CN18 OCT 122
COD11 DEC 11
CN26 DEC 26
N21 DEC 21
NUM0 DEC 0
QUE DEC 1
      DEC 3
      DEC 2
      DEF GAIN
      DEC 3
      DEC 1
      DEF CHANL
      DEC 3
      DEC 0
      DEF SEQ
      DEC 4
      DEC 16
      DEF BUFR
      DEC 3
      DEC 1
      DEF CLEAN
BUFR BSS 16
TIME BSS 5
YEAR BSS 1
GAIN OCT 40200
      OCT 7
SEQ OCT 160200
CLEAN OCT 120000
CHANL OCT 120200
N6 DEC 6
      END AINLC
```

INSTRUMENTATION

ASMB,R,L,C

*
*
* THIS DUMMY DRIVER SERVES AS A DATA BUFFER AND INTERFACE FOR RTE
* PROGRAMS USING THE 2313 ANALOG SUB SYSTEM. IT IS USEFUL IN
* INSTRUMENT APPLICATIONS WITH LOW DATA RATES.(1 POINT/SEC TO ANY
* LOWER RATE. IT ELIMINATES THE NEED TO WRITE A NUMBER OF CORE
* RESIDENT PROGRAMS DEDICATED TO DIFFERENT INSTRUMENTS. IT REQUIRES
* ONE CORE RESIDENT PROGRAM WHICH READS 16 CHANNELS OF THE 2313
* SUB SYSTEM.THIS CORE RESIDENT PROGRAM WRITES TO THE DRIVER. ANY
* NUMBER OF DISK RESIDENTS CAN READ FROM IT.
* THE 16 CHANNELS ARE IN TWO GROUPS,FAST AND SLOW. THE FAST HAS
* BUFFERING FOR 20 POINTS THE SLOW 1. A STACK POINTER SHOWS THE MOST
* RECENT POINT. IN ADDITION TO DATA,THE TIME OF THE MOST RECENT POINT
* IS SAVED AND AN OFFSET FOR EACH OF THE 20 POINTS IS RETAINED. THE
* OFFSET RESOLUTION OF 10'S MILLI SEC.
* PROGRAMS WHICH READ FROM THIS DRIVER GET FAST OR SLOW STORED
* DATA. THE CALLING PROGRAM SELECTS THE DATA OF INTEREST. THE CALLS
* ARE:

* CALL EXEC(1,LU,IDATAF,186) FOR FAST DATA

* CALL EXEC(1,LU,IDATAS,13) FOR SLOW DATA

* IDATAF CONTAINS TIME(5),STACK PT(1),OFFSET TIMES(20),ANALOG 8-15(160)

* IDATAS CONTAINS TIME(5),ANALOG 0-7(8)

* REFER TO PROGRAMS AINLC AND SPR00 OR ANSAV FOR TYPICAL PROCEDURES.

* THE DRIVER IS CONFIGURED AT GEN TIME LIKE ANY OTHER.

* EQT ENTRY
* 35,DVR77,T=0
* INT TABLE
* 35,EQT,21
* ***INITIATOR SECTION***

*
* NAM DVR77
* ENT I.77,C.77
I.77 NOP
LDA EQT6,I GET CONTROL WORD
AND =B3
CPA =B1 IS THIS A READ
JMP READ YES GO TO READ
CPA =B2 IS THIS A WRITE
JMP WRITE YES GO TO WRITE
ILLR LDA =B2 NONE OF THE ABOVE
JMP I.77,I ILLEGAL CALL EXIT
READ LDA EQT8,I
CPA K13 IS THIS FOR SLOW DATA
JMP SLOWR YES
CPA K186 IS THIS FOR FAST DATA
JMP FASR YES
JMP ILLR NONE OF THE ABOVE
SLOWR LDA SDATP GET SOURCE ADDRESS
LDB EQT7,I GET DES. ADDRESS
MVW K13 MOVE DATA OUT
JMP EXIT
FASR LDA FDATP GET SOURCE ADDRESS
LDB EQT7,I GET DES. ADDRESS
MVW K186 MOVE DATA OUT
JMP EXIT

INSTRUMENTATION

```

WRITE LDA 1717B      GET ID ADDRESS
      ADA =D14       BIAS FOR WORD 14
      LDA A,I        GET WORD FOR TYPE
      AND =B17
      CPA =B1
      RSS
      JMP ILLR
      LDA EQT7,I    GET SOURCE ADDRESS
      LDB SDATP     GET DES ADDRESS
      MVW K8        MOVE DATA
      LDB FSKPT     GET DES ADDRESS
      MVW K8        MOVE DATA
      STB FSKPT
      LDB PTIME     GET DES ADDRESS
      MVW K5
      LDA TIME+1    LOAD IN SECONDS
      MPY K100      MULT SEC BY 100
      ADA TIME      COMBINE 10'S MSEC
      LDB PINTV     LOAD INTERVAL POINTER
      ADB STKPT     LOAD OFFSET
      STA B,I       STORETIME
      LDA STKPT
      INA
      CPA =D20      INC STACK POINTER TEST
      JMP RESET     FOR RESET
      STA STKPT
EXIT  LDA =B4
      LDB EQT8,I
      JMP I.77,I
RESET CLA
      STA STKPT
      LDA IFSKP
      STA FSKPT
      JMP EXIT
*
*
C.77  NOP          ***CONTINUATOR SECTION***
      ISZ C.77     SPURIOUS INPUT
      JMP C.77,I
A     EQU 0
B     EQU 1
SDATA BSS 8
TIME  BSS 5
STKPT NOP
INTVL BSS 20
FDATA BSS 160
SDATP DEF SDATA
FDATP DEF TIME
IFSKP DEF FDATA
PTIME DEF TIME
FSKPT DEF FDATA
K100  DEC 100
K5    DEC 5
K13   DEC 13
K8    DEC 8
K186  DEC 186
PINTV DEF INTVL
.     EQU 1650B
*
*
*
EQT6  EQU .+13
EQT7  EQU .+14
EQT8  EQU .+15
END

```

The inclusion of the following article in the HP 1000 Communicator does not imply endorsement of the OEM's product by Hewlett-Packard. Hewlett-Packard assumes no responsibility for the information contained herein or responsibility for the OEM's product.

A MODERN LANGUAGE FOR ON-LINE SYSTEMS

*by David Hamilton
Theta Computer Systems*

INTRODUCTION

Hewlett-Packard HP-1000 computer systems offer users a combination of computing power and flexibility previously found only in much larger, more expensive equipment. These computers provide rapid random-access storage, good CRT communications capabilities, fast internal computation speeds and a powerful repertoire of byte-manipulation instructions. These capabilities make the HP-1000 computer systems ideally suited to the development of on-line systems which support the simultaneous access of many users to a common base of stored information.

Although HP-1000 computer systems represent a substantial reduction in the cost of computing, the cost of designing, programming and implementing on-line systems remains high. Users commonly experience problems in the development of on-line systems and find that the cost of designing, programming, and maintaining an on-line system equals or exceeds the initial hardware investment. However, software tools, structured programming techniques, and structured design approaches exist which have the potential for greatly reducing on-line system development time and cost.

Theta Computer Systems, a Hewlett-Packard OEM, has developed a new, general-purpose structured programming language with the specific intent of reducing the cost of developing on-line systems and applications programs. This new programming language is called QBOL. The purpose of this article is to briefly demonstrate the need for a new programming language and to illustrate a very small number of the QBOL language features which simplify the development of on-line systems on the HP-1000.

THE NEED FOR A NEW PROGRAMMING LANGUAGE

All on-line systems consist of three essential components. These components exist whether the function of the on-line system is data reduction, stress analysis, inventory management, or purchase order entry. The three essential components are as follows:

- CRT communications and user interface.
- Data Base access and maintenance.
- Task related computational functions.

All of the design and programming activities needed to develop an on-line system ultimately lie within one of these three areas. The QBOL language was designed with each of these three processing requirements in mind and the language embodies specific features which simplify design and programming in each area.

By contrast, FORTRAN and ALGOL were developed in the early days of data processing when batch-oriented 'number crunching' was the only economically feasible application of digital computers. Virtually all programming was done by scientists or engineers and consisted of little more than FORmula TRANslation which could be suitably accomplished with a rudimentary ALGOrithmic Language. FORTRAN and ALGOL were designed at a time when sequential, unit-record devices were the only I/O devices available. Ease of maintenance was a secondary requirement in programming and, as a consequence, neither FORTRAN nor ALGOL embodies structured programming concepts or satisfactory program annotation facilities.

BASIC, on the other hand, was developed at Dartmouth College in the early 1960's to facilitate the development of 'quickie' programs by inexperienced users. Customarily implemented as an interpreter rather than a compiler, BASIC typically imposes a tremendous program storage and execution overhead. This makes BASIC unsuitable for developing programs and systems which must perform many complex functions in the shortest possible time. As a 'quickie' language, BASIC is also unsuited to applications requiring extended arithmetic precision.



QBOL is a modern, powerful, general-purpose programming language designed to facilitate the development of on-line systems and programming applications of many varieties. Extremely easy to learn and use, QBOL nonetheless offers power and flexibility to solve advanced programming and system development problems. The QBOL compiler produces efficient relocatable code for RTE II/III. Language features include 48-bit arithmetic, comprehensive CRT and mass-storage I/O, full string and array manipulation, nested compound IF's, and much more.

The remainder of this article returns to the problem of the development of on-line systems, addresses each of the three essential functional areas which all on-line systems embody, and briefly highlights a few QBOL language features designed to facilitate programming in each area.

CRT COMMUNICATIONS AND USER INTERFACE

By definition, an on-line system accepts user requests for service either through a remote hard-copy device or CRT. In turn, the system performs functions specified by these requests and returns resulting information to the user. While the detailed, bit-by-bit transfer of information between the CRT and the computer mainframe is made extremely simple by existing Hewlett-Packard software facilities, the human factors involved in this communication present additional challenges.

Ideally, a good on-line system accepts function requests and displays results in a format which is convenient for the user. Assuming that CRT's are to be used, the system designer first lays out the detailed display format associated with each system functional request or information display. Good CRT layouts have the following characteristics:

- Function requests always appear in the same area of the CRT.
- Required or optional input parameters are clearly indicated in the display and consistently organized from one function to another.
- The system makes liberal use of CRT cursor positioning capabilities to direct the user to needed input fields.
- Information produced by the system is always displayed in a consistent format with each result field clearly labeled.
- Any error messages or warnings produced by the system are displayed in a reserved area of the screen which is used exclusively for this purpose.

The computer listing which follows is a complete QBOL program which demonstrates the simplicity of CRT communications, cursor-positioning and screen format control. All QBOL programs begin with a declaration of the variables, buffers, and character strings used within the program. The program procedure immediately follows the declaration of program constants and storage. Unless otherwise directed by the programmer, QBOL assumes that the first executable statement appearing in a program is to be the program primary entry point. The following QBOL program is for illustration purposes only and does not necessarily represent the best or easiest way to perform the indicated functions using QBOL. The terminal control sequences used in the program are for the Hewlett-Packard 2640 series CRT terminals.

```
DEMO      START
*         DEMONSTRATE SIMPLE CRT I/O USING CURSOR POSITIONING, FORMAT
*         CONTROL, AND TEXT CONCATENATION
*
*         DECLARE CRT COMMAND BYTE SEQUENCES
*
CLEAR     A2      X''1B4A''      CLEAR SCREEN SEQUENCE
HOME     A2      X''1B48''      HOME CURSOR SEQUENCE
BLINK    A4      X''1B266441''  START BLINKING FIELD
ENDBLINK A4      X''1B26640''  END BLINKING FIELD
```

OEM CORNER

```
*
*
*           DECLARE GENERAL-PURPOSE CURSOR-POSITIONING SEQUENCE
*
CURSOR   REC9           CURSOR NAMES THE NEXT 9 CHARACTERS
        A3   X"1B2661"   ESCAPE, &, LOWER-CASE A
COLUMN   A2           TWO CHARS FOR COLUMN POSITION
        A1   X"63"       LOWER-CASE C TO INDICATE COLUMN
ROW      A2           TWO CHARACTERS FOR ROW
        A1   "R"         UPPER-CASE R FOR ROW AND END SEQUENCE
*
*           DECLARE USER RESPONSE AREA INPUT LENGTH, TERMINAL LU
*           AND SOME CANNED MESSAGES TO USE LATER
*           WORK        BEGIN A WORKING STORAGE AREA
REPLY    A5           5-CHARACTER INPUT BUFFER
LENGTH   I2           ONE-WORD INTEGER FOR INPUT LENGTH COUNT
LU       I2   26      USE MASTER TERMINAL (LU=1)
*
PATONBAK A26 "EXCELLENT! YOU ARE CORRECT"
ERRORMSG A15 "!!! WRONG !!!!"
*
*           BEGIN PROCEDURE BY SENDING OUT "HELLO" MESSAGES
*           THE QBOL PTERM VERB DIRECTS OUTPUT TO THE SPECIFIED LU
*
BEGIN    PTERM LU,"HELLO, THIS IS A DEMONSTRATION"
RETRY    PTERM LU,"I WILL NOW CLEAR-HOME AND ASK A QUESTION"
*        CONCATENATED FIELDS ARE SEPARATED BY COMMAS
*        PTERM LU,HOME,CLEAR,"WHO IS BURIED IN GRANT'S TOMB?"
*
*           NOW ACCEPT USER RESPONSE - THE QBOL GTERM VERB GETS
*           INPUT FROM THE SPECIFIED LU INTO THE SPECIFIED BUFFER
*           AND SETS THE LENGTH RECEIVED IN THE LENGTH FIELD
*
*        GTERM LU,REPLY,LENGTH
*
*           NOW VALIDATE THE ANSWER
*
IF       REPLY="GRANT"   IF THE ANSWER IS CORRECT,
        PTERM LU,PATONBAK SEND OUT A CONGRATULATORY REPLY
        STOP             AND TERMINATE PROGRAM EXECUTION
ELSE     BUT IF IT'S NOT,
        MOVE COLUMN,"50" SET UP A CURSOR SEQUENCE TO A SPECIAL
        MOVE ROW,"01"    ERROR AREA IN ROW 1, COLUMN 50
*        SEND ERRORMSG OUT, AND MAKE IT
*        BLINK TO GET HIS ATTENTION
*        PTERM LU,CURSOR,BLINK,ERRORMSG,ENDBLINK
*        WAIT 5          THEN WAIT 5 SECONDS
*        GOTO  RETRY     AND ASK THE QUESTION AGAIN
*
*           END          THAT'S THE WHOLE PROGRAM
```

DATA BASE ACCESS AND MAINTENANCE

The central characteristic of most on-line systems is that they provide a mechanism by which system users have access to a body of stored information and are able to interrogate, summarize, or modify this information base. Thus, an on-line stress-analysis system may contain data files which include physical properties of materials, physical component descriptions, or results of previously conducted tests. Similarly, inventory control system typically contain product descriptions, warehouse locations, and inventory counts. In either situation, the system user needs to randomly access the information files to obtain the specific information which he requires. Therefore, the system designer must provide a file accessing subsystem which permits rapid random retrieval, alteration, addition, or deletion of data stored within the system files. In addition to this random

accessing requirement, the system designer should plan for sequential access to facilitate the preparation of reports which summarize the information contained in entire files or segments of files. The following program demonstrates the extremely simple QBOL sequential and random-access I/O capabilities. Although the program has a 'batch' structure for brevity, the demonstrated QBOL statements work equally well in an on-line environment.

```

110      START
      EXTRN UPDAT          DECLARE NAME OF EXTERNAL SUBROUTINE
*
*          DECLARE FMP DATA CONTROL BLOCKS FOR TWO FILES
*
TRANFILE DCB290          DCB FOR TRANSACTION FILE
RANDFILE DCB290          DCB FOR RANDOM-ACCESS FILE
*
FLAG      12              DECLARE IO STATUS FLAG
*
TRANREC   REC14           TRANREC NAMES THE NEXT 14 CHARACTERS
*
TRANDATE  A6              DATE OF THIS TRANSACTION IN FORM MMDDYY
TRANAMT   I6              AMOUNT OF THIS TRANSACTION (48-BIT INTEGER)
RECNUM    I2              NUMBER OF RANDOM FILE RECORD TO
*                          WHICH THIS TRANSACTION APPLIES
*
*          NOW DEFINE FORMAT OF RANDOM ACCESS FILE RECORDS
*
RANDREC   REC12
LASTTRAN  A6              DATE OF LAST TRANSACTION AGAINST THIS REC
RANDTOT   I6              TOTAL OF TRANSACTIONS AGAINST THIS REC
*
*          BEGIN ACTUAL UPDATE PROCESS
*          OPEN SEQUENTIAL INPUT TRANFILE AND RANDOM-ACCESS UPDATE
*          FILE RANDFILE
*
OPEN TRANFILE,FLAG,"TRANS","I"
*
*          IF YOU WISH, YOU MAY TEST THE "OPEN" COMPLETION STATUS
*
IF FLAG.LT.0              TEST SUCCESSFUL OPEN
  PTERM 1,"UNABLE TO OPEN TRANFILE" IF UNSUCCESSFUL, NOTIFY
  STOP MASTER TERMINAL AND STOP
ENDIF
*
OPEN RANDFILE,FLAG,"RANDY","U"
*
*          RETRIEVE THE NEXT SEQUENTIAL RECORD FROM TRANSACTION FILE
*          PUT I/O COMPLETION CODE IN "FLAG" READ RECORD INTO
*          TRANREC, GO TO ENDTRAN IF END OF FILE
* PROCLOOP GET TRANFILE,FLAG,TRANREC,ENDTRAN
*
*          COMPLETE I/O STATUS RETURNED IN FLAG
*
IF FLAG.LT.0              TEST FOR ACCESS ERROR
  PTERM 1,"ACCESS ERROR ON TRANFILE" IF ERROR, NOTIFY
  STOP MASTER TERMINAL AND STOP
ENDIF
*
*          NOW RANDOMLY READ THE RECORD SPECIFIED BY RECNUM IN THE
*          TRANSACTION RECORD JUST READ
*
READ RANDFILE,FLAG,RANDREC,RECNUM

```

OEM CORNER

```
*
*           NOW CALL AN EXTERNAL ROUTINE TO UPDATE THE RANDOM-ACCESS
*           RECORD BASED UPON THE CHANGES IN TRANREC
*
*           CALL  UPDAT,RANDREC,TRANREC  PASS RANDOM RECORD AND TRANSACTION
*                               ADDRESSES TO EXTERNAL PROGRAM
*
*           RETURN HERE WHEN "UPDAT" IS FINISHED AND WRITE OUT
*           THE MODIFIED RANDOM-ACCESS RECORD
*
*           WRITE RANDFILE,FLAG,RANDREC,RECNUM
*
*           GOTO  PROCLOOP           CONTINUE PROCESSING TRANSACTIONS
*
*
*           COME HERE AT END OF TRANSACTION FILE
ENDTRAN  CLOSE TRANFILE
         CLOSE RANDFILE
         PTERM 1,"FILE UPDATE COMPLETE"  ELL MASTER TERMINAL WE ARE
         STOP                          DONE - TERMINATE PROGRAM
*
*           END
```

Although data base accessing and maintenance should be controlled by a single subsystem, the retrieved data records are typically manipulated by numerous different programs. Structured systems development dictates that the designer provide standard record layout definitions to be used by all programs in the system. Ideally, this should be accomplished by means of a programming language feature which permits the system developer to 'copy' these standard record definitions from separate source files. In this way, every program in the system automatically refers to the same data by the same name and each program listing automatically reflects the most up-to-date record layouts.

The following example illustrates a QBOL program which is to be called by the I/O program shown in the preceding example. Since this program manipulates the same data records as the calling program, the QBOL COPY feature is used to ensure that the program will use the same data names for the same data elements. The example assumes that two FMP source files exist. These files, named &TRANS and &RANDY contain the exact statements used to define TRANREC and RANDREC in the preceding example. As the example shows, the programmer only needs to code the two '+ +COPY' statements to include the record definitions in the program. At compile time, QBOL will insert the copied statements so that the programmer sees the record layouts in his program listing.

```
          START 7                A SEPARATELY COMPILED UPDATE PROCEDURE
*
*           COPY IN TRANSACTION FILE AND RANDOM-ACCESS DEFINITIONS
*           AS PART OF THIS COMPILE
*
*           ++COPY &TRANS
*
*           ++COPY &RANDY
*
*           ENTRY  UPDAT          DECLARE ENTRY POINT FOR EXTERNAL CALLERS
UPDAT     SUBR  RANDREC,TRANREC  THIS IS A SUBROUTINE WHICH PROCESSES
*                               EXTERNAL PARAMETERS RANDREC AND TRANREC
*
*           MOVE  LASTTRAN,TRANDATE  SET NEW LAST TRANSACTION DATE
*           SET   RANDTOT=RANDTOT+TRANSAMT  ADD THIS TRANSACTION AMOUNT TO
*                               PREVIOUS TOTAL FOR THIS ITEM
*
*           EXIT  UPDAT          RETURN TO CALLER
*
*           END
```

TASK-RELATED COMPUTATIONAL FUNCTIONS

The range of possible task-related computational functions performed by on-line systems is as widely varied as the requirements for these systems themselves. Nevertheless, certain computational procedures are common to virtually all on-line systems. A few of the most common functions are as follows:

- Extended-precision arithmetic to conveniently support the very common instances in which internal system results exceed the capacity of one 16-bit computer word.
- Boolean operations to test and set bits within words for status testing, masking operations, and bit array manipulation.
- Data format conversion to translate variable-length ASCII numbers entered by the system user to internal integer values for comparison or computation.
- Field editing to convert internal integer values into ASCII display strings inserting commas, decimal points, dollars signs, suppressing leading zeros, and so on.
- Compound logical or numeric comparison tests to simplify request analysis and internal program decision making.
- Variable-length character string comparison and movement.
- Manipulation of ASCII, integer, or mixed data arrays.

QBOL supports each of these functional requirements with simple, direct and consistent language features. The following QBOL statements illustrate only a fraction of these features. Variables used in the example statements are assumed to have been previously defined and to be the type and size suggested by the variable name.

```

*
*           EXTENDED PRECISION ARITHMETIC
*
SET  ONEWORD1=ONEWORD2+ONEWORD3      SINGLE PRECISION
SET  TWOWORD1=ONEWORD1[ONEWORD2      DOUBLE PRECISION RESULT
SET  THREWORD=ONEWORD+TWOWORD+THREWORD MIXED VARIABLE SIZES
SET  ONEWORD=THREWORD1=THREWORD2      OK IF RESULT FITS IN 16 BITS
SET  A=B+(C*D/E+13(F-X"FA"+0"77"))    COMPLEX EXPRESSIONS ARE POSSIBLE
*
*           BOOLEAN OPERATIONS
*
SET  INTEGER=X"5555"                   SET ALL EVEN BITS IN ONE WORD
SET  INTEGER=INTEGER.XOR.X"FFFF"      EXCLUSIVE OR NOW SETS ALL EVEN BITS
SET  A=B.OR.C.AND.D.XOR.E+2           MIX AND MATCH
*
*           EDIT NUMERIC FIELD FOR DISPLAY
*
MOVE MASK,"$$,###,###.XX-"           MOVE EDIT MASK TO VARIABLE
SET  INTEGER=10100000000
EDIT DISPLAY,INTEGER,MASK              DISPLAY IS $1,000,000.00
EDIT DISPLAY,INTEGER,"$$,###,###.XX-" SAME RESULT
*
*           NESTED COMPOUND LOGICAL AND NUMERIC TESTS
*           UP TO TWENTY LEVELS OF NESTING ARE POSSIBLE
*

```

OEM CORNER

```
IF  STRING1=STRING2          STRING COMPARE
AND  A.LT.B                   AND'ED WITH A NUMERIC TEST
OR   C+D.GT.E5F-G/H          OR'ED WITH A MORE COMPLEX NUMERIC TEST
    SET SAM=DOG              STATEMENT/S FOR TRUE CONDITION
    IF  STRING3.GE.STRING4    NESTED IF TEST
    SET DOG=SAM              DO IF NESTED IF IS TRUE
ELSE
    SET DOG=CAT              DO IF NESTED IF IS FALSE
ENDIF                        END NESTED IF
ELSE                          START OF FALSE STMTS FOR MAIN IF
    GOTO HENRY              FALSE CONDITION STATEMENT/S
ENDIF
```

```
*
*      MANIPULATE VARIABLE-LENGTH CHARACTER STRINGS
*
MOVE STRING1,STRING2          MOVE STRING2 TO STRING1
MOVE STRING1(2,4),STRING2(1,3) CHARS 1-3 OF STRING2 INTO
                                CHARS 2-4 OF STRING1
*
MOVE STRING1,"ABCXF"
MOVE STRING1(4,5),"DE"      STRING1 BECOMES ABCDEF
SET  X1=3
SET  X2=5
MOVE STRING2,STRING(X1,X2)   STRING2 GETS "CDE"
*      USE STRING ANYWHERE AN ALPHA VARIABLE IS PERMITTED
IF  STRING(X1,X2).LT.STRING2(X3,X4)
*
*      CONVERT VARIABLE-LENGTH ASCII INPUT TO INTEGER,
*      SET FLAG IF INPUT NOT NUMERIC
*      GTERM LU,INPUT,LENGTH
*      CONV INTEGER,INPUT(1,LENGTH),FLAG
```

CONCLUSION

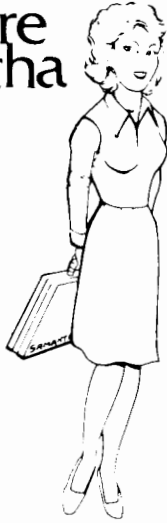
Modern Hewlett-Packard hardware technology represents a substantial reduction in the cost of computing. The modern QBOL programming language represents a similar reduction in the cost of programming. QBOL embodies a complete spectrum of language features directly related to the development of on-line systems and most programming applications. As a result, users may now consider the development of on-line systems and programming applications which were not economically feasible in the past.

Persons desiring to receive, at no charge, a complete programming reference manual for QBOL should address inquiries to:

DAVID C. HAMILTON
THETA COMPUTER SYSTEMS
6919 VALJEAN AVENUE
VAN NUYS, CA 91406
(213) 994-4775

End user single-payment licenses for the QBOL compiler are priced at \$5,000.00. The license entitles the user to reproduce the software for his own use but not for resale.

Software Samantha



Samantha has not received any questions from readers for this issue. However, she would like to announce the availability of a new manual called "RTE Operating System Driver Writing Manual", HP part number 92200-93005. This manual has over 90 pages of very detailed information on how to write drivers for RTE II, RTE III, RTE-M or RTE IV. Standard drivers (which operate with the interrupt system off) as well as privileged drivers are covered. The manual includes much "peripheral information" about the interaction of the driver with the operating systems. Among the many topics covered are: time-out processing, DCPC processing and driver requests for DCPC assignment, power up/down sequence, mapping done by a driver and much, much more. It is already in wide use by the HP field organization and has received many words of praise for its clarity and completeness of data.

Contact your local HP office for ordering information on this manual.

Samantha invites all questions from our readers of a technical nature on any aspect of HP-1000 systems. All letters will be answered, whether or not they are chosen for inclusion in the Communicator.

Address: Software Samantha
HP-1000 Communicator
Hewlett-Packard Data Systems Division
11000 Wolfe Road Cupertino, California 95014

RTE-II/III to RTE-IV UPGRADE COURSE AVAILABLE

If you are one of the many customers who are planning to upgrade your existing RTE-II or RTE-III Operating System installation to the new RTE-IV Operating System, take note: A two day *RTE-II/III to RTE-IV Upgrade Course* is available. This course, which assumes a thorough knowledge of RTE-II/III as a prerequisite, will provide you with detailed information on all of the new features of RTE-IV. Class time is divided between lecture material which explains the new features, and hands-on lab time with the RTE-IV Operating System. Also supplied is a complete set of new manuals, such as the RTE-IV Programming and Reference Manual and the RTE-IV Generation Manual. Course fee is \$250.00 in the United States. Contact your local HP representative for a course data sheet and the current schedule of classes.



NEW TRAINING PROGRAM FOR HP-1000 COMPUTER SYSTEMS

Hewlett-Packard has developed a new Customer Training Program for HP-1000 Computer Systems. The introduction of this program coincides with the introduction of the new RTE/IV operating system and the new HP-1000 F-Series Computers. The current program is diagrammed on the next pages. Detailed descriptions of each course follow on the succeeding pages.

Highlight of the program is the brand new *HP-1000 Disc-Based RTE Operating System Course (22991A)*. This two week course covers the use and programming of the RTE-IV operating system in an HP-1000 environment, including program preparation (FORTRAN IV Compiler, assembler, editor, and loader), EXEC calls to invoke system functions, system software generation, and use of the Batch-Spool Monitor. All of the new features of RTE-IV, such as EMA (extended memory area), reconfiguration on boot-up, and the new programming and debugging aids, are covered in detail.

Also new is the *HP-1000 Memory-Based RTE Operating System Course (22992A)*. This two week course covers the use and programming of the RTE-M operating system in an HP-1000 environment, including program preparation, EXEC calls, system software generation, and the file manager.

The third new course is the three-day *Introduction to HP Minicomputers Course (22951B)*. This course provides an entry point into HP computer training for those students who have had no previous experience with minicomputer systems. Students will be exposed to the concepts of HP minicomputer architecture, operating systems, and high-level languages, thereby satisfying the prerequisite of the HP-1000 Disc- and Memory-Based RTE courses described above.

More detailed information on these and the other courses in the training program is provided later in this section, where we have reprinted the Customer Training Section of the HP-1000 Active Software Data Book. This useful guide, which can be obtained from your local HP representative, contains information on all active HP-1000 software products and associated training courses.

Unfortunately, no schedules for the new courses were available at print time. However, by the time you receive this issue, a revised course schedule should be available. We invite you to obtain one from your local HP representative. In any case, the next issue of the Communicator will contain a complete schedule of the new courses.

SETTING UP A TRAINING PROGRAM

We encourage you to discuss your training requirements with your local HP representative. This person is trained to assist you in setting up an optimum training plan for your needs. However, the following comments about the new training program may help you to prepare in advance for this discussion.

In general, courses should be taken in the sequence indicated in the training program diagram, starting from the left, and proceeding toward the right. Completion of each course in sequence will ensure that all needed prerequisites are satisfied.

If you have not had any previous experience with minicomputer systems, you should start your training with the three day *Introduction to HP Minicomputers Course*. Otherwise, you can skip this course, and begin your training with either the *HP-1000 Disc-Based or Memory-Based RTE Operating System Course*. Which one you choose will depend upon the type of system in your installation. Note however, that both of these courses require a thorough knowledge of FORTRAN programming as a prerequisite.

All HP-1000 Computer System users should plan to take one of the Operating Systems Courses described above. Further training is optional, depending upon the nature of your programming tasks. For example, if you are planning to:

- Design a data base using the IMAGE/1000 software, ...
You should take the *IMAGE/1000 Data Base Management Course (22977A)*
- Connect instruments to your HP-1000 via the HP-IB, ...
You should take the *HP-IB in a Minicomputer Environment Course (22980B)*
- Operate your system as part of a distributed systems (DS/1000) network, ...

BULLETINS

You should take the *DS/1000 User's Course* (22987A). Furthermore, if you are to be designated as the Network Manager for your DS/1000 network, you should follow this course with the *Theory of Operation of DS/1000 Course* (22961B). And if your network will include an HP-3000 System, you should continue your training and take the one-day *Theory of Operation for DS/1000-to-HP 3000 Course* (22962B).

- Write programs in HP Assembly Language,...

You should take the *HP-1000 Assembler Programming Course* (22952B). (Note that this course is a prerequisite for the Driver Writing and Microprogramming Courses mentioned below.)

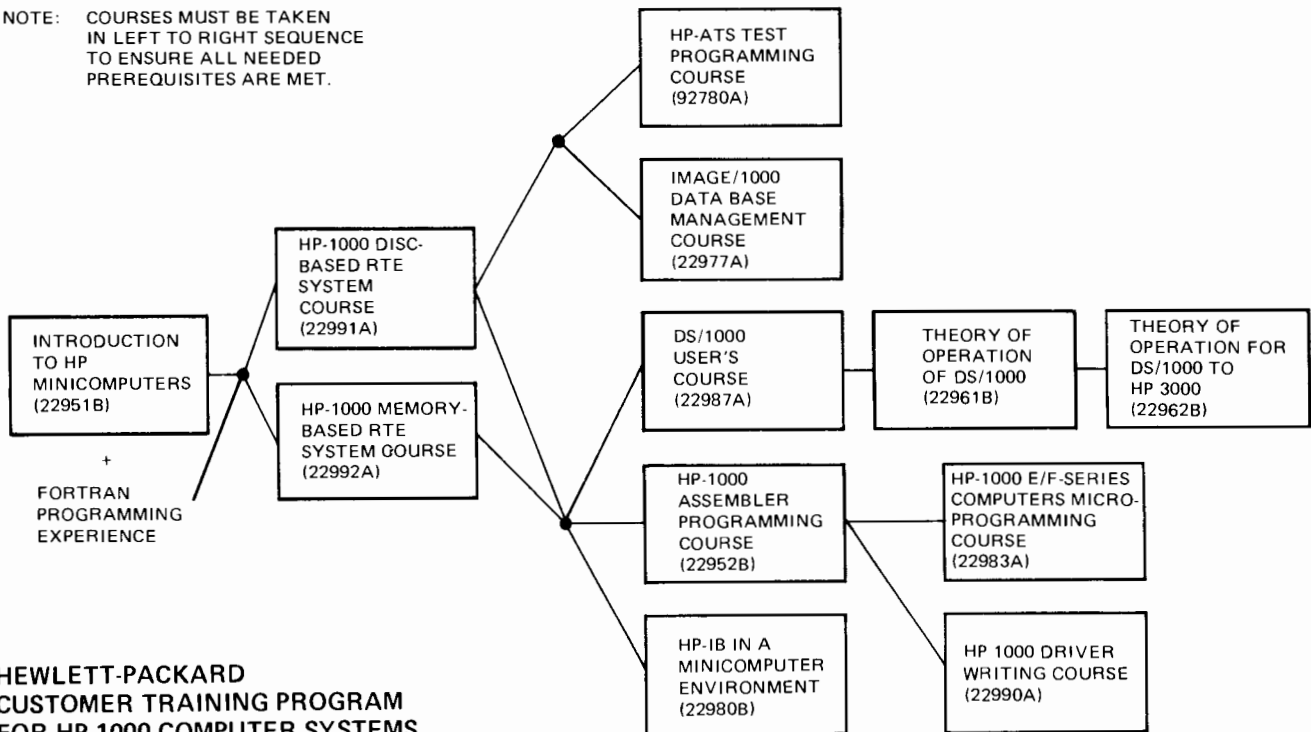
- Interface your own peripheral equipment to your HP-1000 System,...
- Customize your computer for your application using the Microprogramming feature of the HP-1000,...
- Write test programs for your HP-ATS System,...

SUMMARY

After reviewing the new customer training program discussed in this section, choose a tentative training plan that satisfies your needs. Then discuss your plan with your local HP representative. This person can assist you with your course selection, provide you with the latest course schedule, and register you in the appropriate courses at the nearest customer training center.

See you in class!

NOTE: COURSES MUST BE TAKEN IN LEFT TO RIGHT SEQUENCE TO ENSURE ALL NEEDED PREREQUISITES ARE MET.



**HEWLETT-PACKARD
CUSTOMER TRAINING PROGRAM
FOR HP 1000 COMPUTER SYSTEMS**

TRAINING SCHEDULE

The current schedule for customer training courses on HP 1000 computer systems products is given in this section. Included are courses offered both in U.S. and in Europe during the upcoming months.

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 U.S. training centers only. For prices of courses at European training centers, please consult your local HP sales office.

DATA SHEETS

Data sheets giving detailed information on each of the courses scheduled are available from your local HP representative.

REGISTRATION

Requests for enrollment in any of the above courses should be made through your local HP representative. That person 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.

U. S. TRAINING CENTER SCHEDULES, LOCATIONS, AND RATES

Course Number	Title		CUPERTINO Customer Training Center	FULLERTON Customer Training Center	ROCKVILLE Customer Training Center	Data Systems Division (Cupertino)	Data Terminals Division (Cupertino)	Customer Service Division (Sunnyvale)	Boise Division (Boise)
	Length	Price							
22965B	RTE-II-III		Jul 10 Jul 24 Aug 7 Aug 21	Jul 17 Aug 14	Jul 10 Jul 24, Aug 7 Aug 21				
	10 days	1000							
	(Course includes RTE-II/III operating system, batch spool monitor and file manager.)								
22985A	RTE-M		Aug 14						
	5 days	500							
22977A*	IMAGE		Jul 24	Jul 31	Aug 21				
	5 days	500							
22952B*	1000 ASMB		Aug 14		Jul 17 Aug 28				
	5 days	500							
22987A*	DS/1000 User's Course		Jul 10 Aug 21		Jul 10				
	5 days	500							
22961B*	DS/1000 Theory of Op.		Aug 28		Jul 17				
	4 days	400							
22962B*	DS/1000 HP 3000 Theory of Op.		Sep 1		Jul 21				
	1 day	100							
22990A*	RTE-Driver Writing		Jul 31		Jul 5				
	3 days	300							
22980B*	HP-IB Minicomputer Environment		Aug 21						
	4 days	400							

*22965B RTE-II/III is a prerequisite for these courses. Other prerequisites may also apply – refer to the data sheet for each course for more information.

U. S. TRAINING CENTER SCHEDULES, LOCATIONS, AND RATES (Continued)

Course Number	Title		CUPERTINO Customer Training Center	FULLERTON Customer Training Center	ROCKVILLE Customer Training Center	Data Systems Division (Cupertino)	Data Terminals Division (Cupertino)	Customer Service Division (Sunnyvale)	Boise Division (Boise)
	Length	Price							
22983A*	21MX-E Microprogram- ming		Jul 17						
	5 days	500							
92780A*	HP-ATS Automatic Test System					Jul 10			
	5 days	1000							
13294A	Dev. Terminal						Jul 10		
	5 days	500							
22940A	21 Maint.							Jul 10 Aug 7	
	10 days	1000							
22941A	21MX Maint.							Jul 24 Jul 31 Aug 21 Aug 28	
	5 days	500							
22942A	7900 Maint.							Jul 31 Aug 28	
	5 days	500							
22945A	7905 Maint.							Jul 17 Aug 7	
	5 days	500							
91302A	2645 Maint.								
	3 days	300							
22943A	7970B Maint.								
	5 days	600							
22944A	7970E Maint.								
	5 days	600							

*22965B RTE-III/III is a prerequisite for these courses. Other prerequisites may also apply – refer to the data sheet for each course for more information.

U.S. TRAINING CENTER ADDRESSES

Cupertino

CUSTOMER TRAINING CENTER
19310 Pruneridge Avenue
Cupertino, CA 95014
(408) 996-9800

DATA SYSTEMS DIVISION
11000 Wolfe Road
Cupertino, CA 95014
(408) 257-7000

DATA TERMINALS DIVISION
19400 Homestead Road
Cupertino, CA 95014
(408) 257-7000

Fullerton

CUSTOMER TRAINING CENTER
1430 E. Orangethorpe Avenue
Fullerton, CA 92631
(714) 870-1000

Rockville

CUSTOMER TRAINING CENTER
4 Choke Cherry Road
Rockville, MD 20850
(301) 948-6370

Sunnyvale

CUSTOMER SERVICE DIVISION
974 East Arques Avenue
Sunnyvale, CA 94086
(408) 735-1550

Boise

BOISE DIVISION
11311 Chinden Boulevard
Boise, Idaho 83702
(208) 377-3000

EUROPEAN TRAINING CENTER SCHEDULES AND LOCATIONS

Course Number	Title	Boblingen	Amsterdam	Madrid	Winnersh	Milan (M) Rome (R)	Stockholm	Grenoble	Orsay	Vienna	Brussels
	Length										
22965B	RTE-II/III 10 days (Course includes RTE-II/III operating system, batch spool monitor and file manager.)	Jul 31 Jul 31 Sept 25 Oct 23	Oct 9 Dec 4	Oct 23	Jul 24 Sep 4	Jul 3 (M) Oct 9 (M)	Aug 28 Oct 9 Nov 27		Jul 17 Aug 28 Nov 6	Aug 7 Oct 9	Oct 2
22985A	RTE-M 5 days	Jul 10 Sep 18									
22977A*	IMAGE 5 days	Jul 17 Sep 4		Jul 3 Nov 6	Aug 29 Oct 23				Jul 3 Sep 25	Oct 23	
22952B*	1000 ASMB 5 days	Jul 24 Sep 11 Oct 23	Sep 11 Nov 6	Oct 16	Jul 10 Aug 14 Oct 16	Sep 11 (M)	Sep 11 Dec 11		Sep 18		
22987A*	DS/1000 User's Course 5 days	Jul 3 Oct 9			May 29					Jul 10	
22961B*	DS/1000 4 days	Aug 28								Jul 17	
22962B*	DS/1000 HP 3000 Theory of Op. 1 day	Sep 1								Jul 21	
22990A*	RTE Driver Writing 3 days										
22980B*	HP-IB Minicomputer Environment 4 days	Aug 14									
22983A*	21MX-E Micro- programming 5 days										

*22965B RTE-II/III is a prerequisite for these courses. Other prerequisites may also apply – refer to the data sheet for each course for more information.

EUROPEAN TRAINING CENTER SCHEDULES AND LOCATIONS (Continued)

Course Number	Title	Boblingen	Amsterdam	Madrid	Winnersh	Milan (M) Rome (R)	Stockholm	Grenoble	Orsay	Vienna	Brussels
	Length										
92780A*	HP-ATS Automatic Test System										
	5 days										
13294A	Dev. Terminal										
	5 days										
22940A	2100 Maint.										
	10 days										
22941A	21MX Maint.										
	5 days										
22942A	7900 Maint.										
	5 days										
22945A	7905 Maint.										
	5 days										
22945A	7905 Maint.										
	5 days										
91302A	2645 Maint.										
	3 days										
22943A	7970B Maint.										
	5 days										
22944A	7970E Maint.										
40270A	Intro to HP Computers	Jul 17 Sep 25							Jul 31 Oct 9		
	5 days										
22965B-H01	FORTTRAN IV	Oct 16		Oct 2							
	5 days										

*22965B RTE-III/III is a prerequisite for these courses. Other prerequisites may also apply – refer to the data sheet for each course for more information.



EUROPEAN TRAINING CENTER ADDRESSES

Boblingen

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

Brussels

Avenue du Col Vert, 1
Groenkraaglaan
B-1170
Brussels, Belgium
Tel: (02) 672 22 40

Stockholm

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

Madrid

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

Amsterdam

Van Heuven Goedhartlaan 121
Amstelveen - 1134
Netherlands
Tel: 02 672 22 40

Orsay

Quartier de Courtaboeuf
Bolte Postale No. 6
F-91401-Orsay
France
Tel:(01) 907 7825

Grenoble

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

Vienna

Handelskai 52
Postfach 7
A - 1205 Wien
Tel: (0222) 35 16 21-32
Telex: 75923
Cable: Hewpack Wien

Milan

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

Winnersh

King Street Lane
Winnersh, Wokingham
Berkshire RG11 5 AR
Tel: Wokingham 784774
Cable: Hewpie London
Telex: 8471789



User training services

NOTE: These three pages are reprinted from the latest edition of the HP 1000 Active Software Data Book.

HP 1000 user training services supporting active software products include the courses listed in this section. The course listings are subdivided into regularly-scheduled courses whose times of presentation are listed on the Hewlett-Packard Computer Systems Group Course Schedule and request-scheduled courses, which are scheduled by the respective Hewlett-Packard Technical Center when there are sufficient requests to justify presentation of the course material. Any of the courses listed here may be presented at suitably-equipped customer's facilities by arrangement with the nearest Hewlett-Packard Technical Center. Technical Center addresses and telephone numbers are listed on the Hewlett-Packard Computer Systems Group Course Schedule.

Regularly-scheduled training courses

22951B Introduction to HP Minicomputers

Description: This course provides an entry point into HP computer training for those customers who have had no previous experience with minicomputer systems. Upon completion of the course, the student will be familiar with the concepts of:

1. HP minicomputer architecture.
2. Operating systems.
3. High level languages.

Length: 3 days.

Lab: Provides a hands-on introduction to the hardware and software operation of HP 1000 minicomputers. This includes operation of the computer front panel, system boot-up procedures and on-line loading and execution of programs.

Prerequisites: None. Students may be either hardware or software oriented.

22991A HP 1000 Disc-Based RTE System Course

Description: This course covers the operation of the RTE-IV operating system in an HP 1000 system environment. This includes program preparation using standard compiler, assembler, editor, and loader; disc usage; system software generation; and use of the Batch-Spool Monitor (BSM), including the file manager.

Length: 10 days.

Lab: Provides hands-on experience in operating, programming, and generating the RTE-IV system, including BSM.

Prerequisites: Demonstrated proficiency in FORTRAN programming (such as completion of a FORTRAN programming course) and completion of the Introduction to HP Minicomputers course (22951B) or equivalent minicomputer experience.

22992A HP 1000 Memory-Based RTE System Course

Description: This course covers the use of the RTE-M operating system in an HP 1000 system environment. This includes program preparation using the standard flexible disc-based FORTRAN IV compiler, assembler, editor, re-locating and absolute loaders; system software generation; and use of the file manager.

Length: 10 days.

Lab: Provides hands-on experience in operating, programming, and generating the RTE-M system, and in on-line program loading and removal.

Prerequisites: Demonstrated proficiency in FORTRAN programming (such as completion of a FORTRAN programming course) and completion of the Introduction to HP Minicomputers course (22951B) or equivalent minicomputer experience.

22987A DS/1000 User's Course

Description: This course covers the fundamentals of the HP DS/1000 Distributed Systems Network, including: network philosophy, operator commands, remote I/O remote file access, remote EXEC calls, program-to-program calls, system software generation, and store-and-forward communications. Information is provided on both memory-based and disc-based RTE systems operation in addition to information on an HP 3000 MPE link.

Length: 5 days.

Lab: Provides hands-on experience in programming of a multi-node DS/1000 distributed systems network.

Prerequisites: Completion of either the HP 1000 Disc-Based RTE System Course (22991A) or the HP 1000 Memory-Based RTE System Course (22992A), or equivalent RTE experience. The HP 3000 Comprehensive Introduction Course (22801A) is also recommended for those customers whose networks include an HP 3000 node.

22961B Theory of Operation of DS/1000

Description: This course provides a thorough exposure to the internal functioning of the DS/1000 software as it relates to an HP 1000-to-HP 1000 link. Topics covered include communications management, microcoded driver, remote file manager, network configuration, link protocol, generation, and performance evaluation. Information is provided on the level of program listings, flowcharts, and tables.

NOTE: Customers whose networks include an HP 3000 node should also take the one-day DS/1000 to HP 3000 Theory of Operation Course (22962B).

Length: 4 days.

Lab: Provides hands-on programming of a DS/1000 network, use of system utilities, diagnostics, and troubleshooting tools. System generation and network configuration are covered in detail.

Prerequisite: Completion of the DS/1000 User's Course (22987A).

22962B Theory of Operation for DS/1000-to-HP 3000

Description: This course provides a thorough exposure to the internal functioning of the DS/1000 software as it relates to an HP 1000-to-HP 3000 link. Topics covered include communications management, network configuration, link protocol, HP 1000 as a master to MPE, and HP 1000 as a slave to MPE. Information is provided on the level of program listings, flowcharts, and tables.

Length: 1 day.

Lab: None.

Prerequisite: Completion of the Theory of Operation of DS/1000 Course (22961B), which is normally taken earlier in the same week.

22977A IMAGE/1000 Data Base Management System Course

Description: This course covers the creation, building, back-up, and modification of data bases using the IMAGE/1000 Data Base Management System. It also includes the writing of programs to access a data base and the use of QUERY to access a data base.

Length: 5 days.

Lab: Provides hands-on experience with IMAGE/1000 software on an HP 1000 system.

Prerequisite: Completion of the HP 1000 Disc-Based RTE System Course (22991A) or equivalent disc-based RTE experience.

22952B HP 1000 Assembler Programming Course

Description: This course covers the operation of the RTE assembler in an HP 1000 computer system environment. Major emphasis is placed on the development of assembly language programs for use in an RTE operating system.

Length: 5 days.

Lab: Provides extensive hands-on experience in the coding, editing, assembly, and debugging of RTE assembler programs using an HP 1000 system.

Prerequisites: Completion of either the HP 1000 Disc-Based RTE System Course (22991A) or the HP 1000 Memory-Based RTE System Course (22992A), or equivalent RTE experience.

22990A HP 1000 Driver Writing Course

Description: This course covers the techniques and requirements for developing RTE device drivers for use in an HP 1000 system. Topics covered include: HP 1000 Computer family hardware and software I/O structure, interrupt-driven drivers, RTE driver structure and operation, use of DCPC by drivers, and privileged RTE drivers.

Length: 3 days.

Lab: Provides extensive hands-on programming experience in the development of RTE drivers.

Prerequisites: Completion of either the HP 1000 Disc-Based RTE System Course (22991A) or the HP 1000 Memory-Based RTE System Course (22992A) and the HP 1000 Assembler Programming Course (22952B), or equivalent RTE and assembly language experience.

22980B HP-IB in a Computer Environment

Description: This course provides an introduction to HP-IB concepts and theory as they apply to use in HP 1000 Computer System controlled measurement systems as well as training in the programming of HP-IB on an RTE system.

Length: 4 days.

Lab: Provides hands-on experience with a typical HP 1000 computer controlled HP-IB instrument system.

Prerequisite: Completion of either the HP 1000 Disc-Based RTE System Course (22991A) or the HP 1000 Memory-Based RTE System Course (22992A) or equivalent RTE experience.

22983A HP 1000 E/F-Series Computers Microprogramming Course

Description: This course covers the theory and use of HP microprogramming hardware and software to prepare, alter, and install microprograms for HP 1000 E-Series computers.

Length: 5 days.

Lab: Provides hands-on experience with preparation and installation of microprograms.

Prerequisite: Completion of either the HP 1000 Disc-Based RTE System Course (22991A) or the HP 1000 Memory-Based RTE System Course (22992A) and the HP 1000 Assembler Programming Course (22952B), or equivalent RTE and assembly language experience.

HP ATS Test Programming Course

Description: This course is intended for customer's test engineers who write test programs for an HP-ATS Automatic Test System. Topics covered include Automatic Test System architecture, test monitor, instrument programming in BASIC, UUT interfacing, and switching control.

Length: 5 days.

Lab: The student uses a multi-terminal system in laboratory sessions to develop programming skills.

Prerequisite: A strong background in electronic test equipment and test procedures as well as training or experience in BASIC or FORTRAN programming. The student must also complete at least the first week of the HP 1000 Disc-Based RTE System Course (22991A).

Request-scheduled training courses

22960A HP 1000 M-Series Computers Microprogramming Course

Description: This course covers the theory and use of HP microprogramming hardware and software to prepare, alter, and install microprograms for HP 1000 M-Series computers

Length: 5 days.

Lab: Provides hands-on experience with preparation and installation of microprograms.

Prerequisite: Completion of either the HP 1000 Disc-Based RTE System Course (22991A) or the HP 1000 Memory-Based RTE System Course (22992A) and the HP 1000 Assembler Programming Course (22952B) or equivalent RTE and assembly language experience.

Maintenance training courses

Maintenance training is also available. See the current HP Computer Systems Group Course Schedule for the list of available courses.

Ordering, registration, and scheduling information

Information on tuition for scheduled courses is provided in the HP 1000 Computers Selection and Configuration Guide and in the HP 1000 Computer Systems Configuration and Site Preparation Guide. Registration and course scheduling information is provided in the current HP Computer Systems Group Course Schedule. All of these documents are available from your Hewlett-Packard Representative.

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 (if quantity is greater than 1 discount is 40%)	_____	\$48.00	_____	_____
	TOTAL DOLLARS for 5951-6111			_____	_____
5951-6112	COMMUNICATOR 2000 (if quantity is greater than 1 discount is 40%)	_____	25.00	_____	_____
	TOTAL DOLLARS for 5951-6112			_____	_____
5951-6113	COMMUNICATOR 3000 (if quantity is greater than 1 discount is 40%)	_____	48.00	_____	_____
	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 _____

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 (if quantity is greater than 1 discount is 40%)	<u>3</u>	\$48.00	<u>\$144.00</u>	
				<u>57.60</u>	
	TOTAL DOLLARS for 5951-6111				<u>\$86.40</u>

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	<u>X X</u>	<u>1</u>	\$10.00	<u>\$10.00</u>	
		<u>x x</u>	<u>2</u>	10.00	<u>20.00</u>	
				10.00		
	TOTAL DOLLARS					<u>\$30.00</u>

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 (if quantity is greater than 1 discount is 40%)		\$48.00		
	TOTAL DOLLARS for 5951-6111				
5951-6112	COMMUNICATOR 2000 (if quantity is greater than 1 discount is 40%)		25.00		
	TOTAL DOLLARS for 5951-6112				
5951-6113	COMMUNICATOR 3000 (if quantity is greater than 1 discount is 40%)		48.00		
	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 _____

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 (if quantity is greater than 1 discount is 40%)	<u>3</u>	\$48.00	<u>\$144.00</u>	
				<u>57.60</u>	
	TOTAL DOLLARS for 5951-6111				<u>\$86.40</u>

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	<u>XX</u>	<u>1</u>	\$10.00	<u>\$10.00</u>	
		<u>xx</u>	<u>2</u>	10.00	<u>20.00</u>	
				10.00		
	TOTAL DOLLARS					<u>\$30.00</u>

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.

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  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		Extended Total	
				Each			

*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:

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

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.

COMPUTER SYSTEMS COMMUNICATOR

NOT TO BE USED FOR ORDERING COMMUNICATOR SUBSCRIPTIONS



**CORPORATE PARTS CENTER
Direct Mail
Parts and Supplies Order Form**

SHIP TO:

NAME _____	CUSTOMER REFERENCE # _____
COMPANY _____	TAXABLE *? _____
STREET _____	
CITY _____ STATE _____	ZIP CODE _____

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

Special Instructions	Sub-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. Check or Money Order, made payable to Hewlett-Packard Company, must accompany order. When completed, please mail this form with payment to:	Your State & Local Sales Taxes		
	Handling Charge	1	50
	TOTAL		

HEWLETT-PACKARD COMPANY
 Mail Order Department Phone: (415) 968-9200
 P.O. Drawer #20
 Mountain View, CA 94043

Most orders are shipped within 24 hours of receipt. Shipments to California, Oregon and Washington will be made via UPS. Other shipments will be sent Air Parcel Post, with the exception that shipments over 25 pounds will be made via truck. No Direct Mail Order can be shipped outside the U.S.

Although every effort is made to ensure 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.