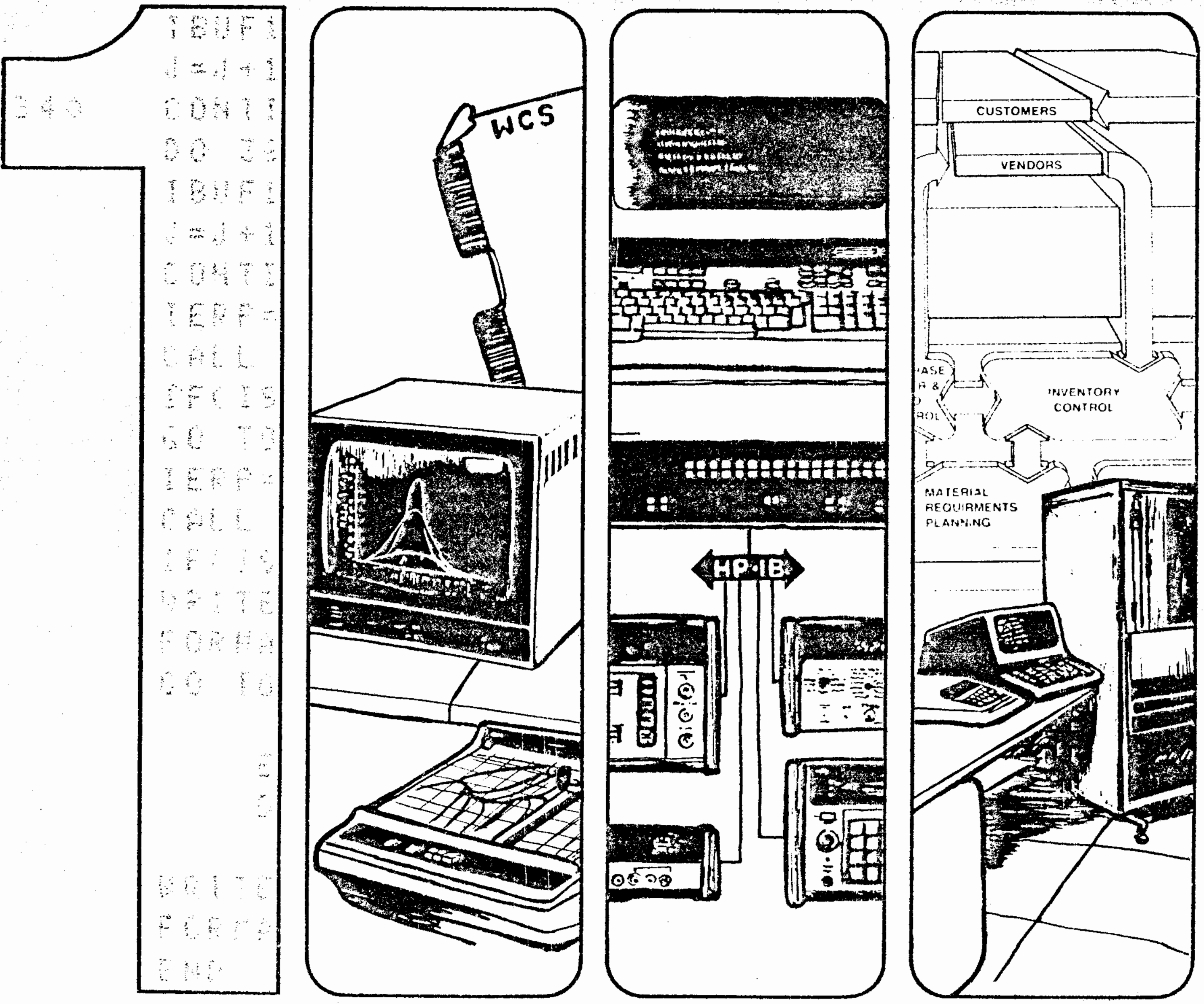
HEWLETT hp PACKARD

Computer Systems COMMUNICATOR



1978

Volume II Issue No. 5

EDITOR'S DESK

Well folks, this issue should probably be called DRIVER HINTS/1000. There are four articles on drivers, plus a question to Software Samantha on generating drivers into the system. It is most unusual to see such a run of similar articles in one issue. Even the calculator winning article was on drivers.

ANNOUNCING CALCULATOR WINNER!

Carl Reynolds of Hewlett-Packard in Rockville, Maryland, is this month's HP-21 winner. *Carl's* winning article on "Operating System Drivers" was judged to be the best based on the areas of clarity, completeness of subject covered and interest to the largest segment of our readership. Alas, once again we have no customer calculator winners, as *John Pezzano's* article is too short and *Alex Swartz's* article is for the OEM Corner.

We would like to apologize to *Harvey Bernard* for inadvertantly leaving his name off his winning article last month. Fortunately, we mentioned on page i that "Type 6 Files" was *Harvey's* article.

Good news for you folks who have not won a calculator yet. We are upgrading the calculator prize. Effective Volume 2 issue 6 all calculator winners will receive HP-32E's rather than HP-21's. We hope that this will be acceptable to everyone, as the HP-32E has many features which the HP-21 does not. Just in case you are unfamiliar with the Communicator/1000 Calculator Contest, the complete rules have been reprinted in this issue.

OEM CORNER ARTICLE

This month we are pleased to be able to print *Alex Swartz's* article "Software for the 2645 Terminal" in our OEM Corner. We hope that more OEM's will contribute to the OEM Corner. Guidelines for submitting an article to the OEM Corner have also been reprinted in this issue.

DEADLINES

The tentative deadlines for submitting articles for Volume 3 are:

Issue 1	January 12, 1979
Issue 2	March 16, 1979
Issue 3	May 11, 1979
Issue 4	July 13,1979
Issue 5	September 7, 1979
Issue 6	November 2, 1979

We hope that as many readers as possible will submit articles, either in hopes of winning a calculator, as a contribution to the OEM Corner, as hints for other users in the Bit Bucket or as letters for User's Queue or for Software Samantha.

CONTENTS

User's Queue

Operating Systems

Operations Management

OEM Corner

Bit Bucket

Bulletins

THE OEM CORNER OF THE HP-1000 COMMUNICATOR

Issue 3 marked the start of a major new section of the HP 1000 Communicator — the 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 standalone 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 issue 3, we had "A Modern Language for On-Line Systems" by ***David C. Hamilton*** of Theta Computer Systems in Van Nuys, California.

In issue 4 there were no articles.

In issue 5, we have "Software for the 2645 Terminal" by *P. Alex Swartz* of Computer Systems Consultants of Tucson, Arizona.

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 numberous 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.

It is expected that OEM software products complement the HP product line or present a more complete solution to a problem. HP, in contrast, sells tools of a general nature. Therefore, some explanation of this sort is permissable in the OEM's article. The article should present a technique or innovative idea of general interest to HP customers.

The readership of the Communicator is assumed to cover the full range from the neophyte to the expert. Therefore, the author may address any level of expertise he chooses. However, the clarity of presentation is always an important consideration, regardless of the assumed background of the reader.

The article should be a minimum of 4 typed, double-spaced pages. Only in unusual cases should an article be more than 10 type written pages.

All articles are subject to editorship and minor revisions. In general the author will be contacted if there is any question of changing the information content. Articles requiring major revision will be returned with an explanatory note. We hope not to return any articles and would like to work with all authors to avercome any objections. However, HP reserves the right to reject any articles judged not to be of general interest to HP customers.

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.

Address all Communications to:

Editor HP 1000 Communicator
Hewlett-Packard Data Systems Division
11000 Wolfe Road
Cupertino, California 95014

WIN AN HP-32E CALCULATOR!

Since its beginning in 1975, the Communicator has changed format several times. During this period, the primary source of technical articles has been employees of HP Data Systems Division. In order to increase the diversity of topics and number of articles we are soliciting articles from customers and other HP divisions. To make it worth your time, three free HP-32E hand-held calculators will be awarded per issue (one to a customer, one to an HP employee of HP Data Systems Division and one to an HP employee not from Data Systems Division) to the authors of the best feature-length articles which fall in one of the following categories:

Operating Systems
Instrumentation
Computation
Operations Management

The employees of the Technical Marketing department of Data Systems Division are not eligible for the calculator prize: all other HP employees are eligible. Customers and HP employees will not compete against each other, since HP employees have access to more information. Likewise, employees of Data Systems Division will not compete against employees not from Data Systems Division. A prize will be awarded even if there is only a single entry.

A feature-length article must meet the following criteria:

1. The topic must be of general interest to our readers and fall into one of the four categories above.

2. It must cover at least two pages in the 1000 Communicator, exclusive of listings and illustrations. At the current print size, this is approximately 1650 words.

The eligibility rules for receiving a calculator are:

1. No individual will be awarded more than one calculator per calender 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 the 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 HP Data Systems Division are not eligible.

The winning article will be the best article submitted based on the areas of clarity, completeness of subject coverage and interest to the largest segment of our readership. All entries will be judged by a team of at least three people in Technical Marketing.

All winners will be announced in the HP 1000 Communicator in the issue in which their articles appear. It is greatly appreciated if the text of the article and any listings are submitted in machine-readable form, i.e. a file on a magnetic tape, mini-cartridge or paper tape.

Address all communications to:

Editor HP 1000 Communicator
Hewlett-Packard Data Systems Division
11000 Wolfe Road
Cupertino, California 95014

USER'S QUEUE

LOCUS PART NUMBER ERROR IN FFT ARTICLE

Glenn Talbott/HP Data Systems Division

The article entitled "Microcoded Fast Fourier Transform For E-Series Computers" by ***Glenn Talbott*** that appeared in Volume II, Issue 3 of the COMMUNICATOR has an error in the part number for ordering the FFT. In the sixth paragraph in the article on page 32, the part number referenced should be

"22682-13396 for the mini-cartridge option ($50.00)".

I hope this did not cause any inconvenience.

NEW CONTRIBUTED PROGRAMS

Elizabeth Caloyannis/HP Data Systems Division

This article serves as an update for the Data Systems LOCUS Program Catalog (22000-90099).

The new contributed programs listed below are now available. Contact your local HP Sales Office to order Contributed Library material, or (if you are in the U.S.) you can use the Direct Mail Order form at the back of the COMMUNICATOR/1000.

22683-18901 TCMPR — Tape Cassette Compare

Tape Cassette Compare. This program compares the data on two tape cassettes, record by record. It lists the differences by word number. A summary of the comparison is output.

22683-18901	paper tape	$40.00
22683-13301	mini-cartridge	$40.00

22683-18902 ENTPT — Entry Point List

Program ENTPT produces an alphabetic list of entry points for an RTE-II system. The list includes the name and address of each entry point.

22683-18902	paper tape	$40.00
22683-13302	mini-cartridge	$35.00

22683-18903 ECARTE — Card Game

ECARTE is a two handed card game in which one is pitted against the computer. A player receives five cards. a card is turned to fix the trump suit, and the object is to win a specified number of tricks.

The program is conversational and it will optionally print out the game rules and instructions for play.

ECARTE is related to the card games EUCHRE, NAPOLEON, SPOIL FIVE and its variant FORTY-FIVE.

22683-18903	paper tape	$40.00
22683-13302	mini-cartridge	$35.00

22683-13304 RTEM — RTE-M Switch Program

The program RTEM converts a type 7 file (absolute binary) into a type 1 file which is a memory image of the corresponding RTE-M system. In addition, it supplies a bootstrap in the first record such that, if the type 1 file begins on cylinder 0, then the memory image may be booted into physical memory with the ROM boot (or equivalent) supplied with the HP 1000 system. This program is of greatest use in developing applications (using the resources of a large system) which will later be transported to a minimal hardware configuration.

The program is interactive — the only parameter in the RU or ON command is the terminal logical unit number, which will default to 1.

22683-13304	mini-cartridge	$35.00

22683-10905 PASCS — PASCAL-S Compiler/Interpreter

PASCS is a segmented program with two segments, PASCC and PASCI. It provides a compiler/interpreter system implementing PASCAL-S, a subset of the programming language PASCAL, on the HP 1000 series computer systems.

PASCC translates PASCAL-S programs into code for a hypothetical stack computer and then loads PASCI to interpret the generated code. Input data (if any) is read in by PASCC, listed on the output device and copied to a work area on the disc, where it will be read in by PASCI.

PASCAL-S differs from full PASCAL in the following aspects:

- Scalar and subrange types are omitted. There are no pointer types.
- Set and file structures are omitted, except for two standard files "input" and "output". These files are predeclared, but must be listed in the program heading if required.
- There are no packing options.
- WITH and GOTO statements are omitted. There are no labels.
- Procedures and functions cannot be used as parameters.
- Only the following standard objects are available:

  constants: true, false
  types: integer, real, boolean, char
  procedures: read, readln, write, writeln
  functions: abs, sqr, odd, chr, ord, succ, pred, round, trunc, sin, cos, exp, ln, sqrt, arctan, eof, eoln

22683-10905	800 bpi MT	$100.00
22683-11905	1600 bpi MT	$100.00

22682-13397 CNFG — New revision to CNFG

This Fortran program describes in an easy to read format, the current configuration and status of HP-IB in RTE-M, RTE-II, RTE-III and RTE-IV when using driver DVR37 (REV. 1840) with SRQ program scheduling.

The program is oriented to three concepts:

A. HP-IB equipment table information.
B. The HP-IB bus logical unit (when one is assigned).
C. The HP-IB device logical unit (when one is assigned).

The information may be obtained in several different ways, in whole or in part. The run statement is:

:RU,BS, <<input>> ,<<list>>, <<eqt#>>, <<lu#>>

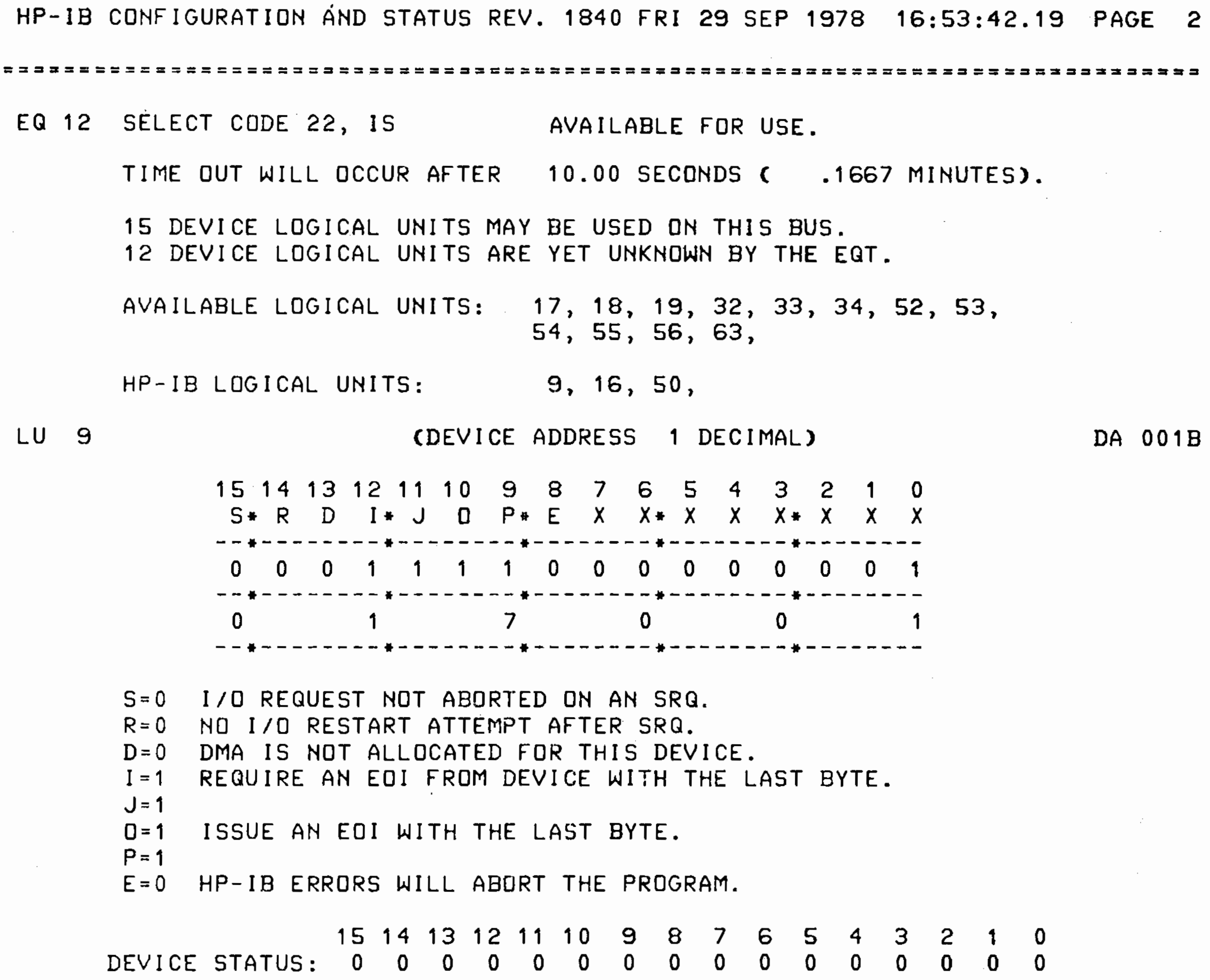
where <<input>> is not used, <<list>> is the list logical unit, <<eqt#>> is the equipement table number of the HP-IB device and <<lu#>> is the logical unit of the HP-IB bus or device logical unit. All parameters are optional and, if omitted, default to giving all HP-IB information for the system.

Besides supplying the information, BS will clean up EQT inconsistancies caused by assigning a logical unit an EQT and a device address, making a request to the logical unit then reassigning the logical unit without making a request to unconfigure the device. The operator at the input terminal is informed of the cleanup.

22682-13397	mini-cartridge	$40.00

Following is sample output from the program:

```
HP-IB CONFIGURATION AND STATUS REV. 1840 FRI 29 SEP 1978  16:53:42.19  PAGE  1
==============================================================================

EQ 11  SELECT CODE 21, IS           DOWN.

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

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

       AVAILABLE LOGICAL UNITS:    17, 18, 19, 32, 33, 34, 52, 53,
                                   54, 55, 56, 63,

       HP-IB LOGICAL UNITS:        15,

LU 15     THIS IS A DEVICE INDEPENDENT, GENERAL BUS LOGICAL UNIT       DA 000B

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

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

       THIS EQT IS DOWN, BUS STATUS INFORMATION IS NOT AVIALIABLE.
```



```
HP-IB CONFIGURATION AND STATUS REV. 1840 FRI 29 SEP 1978  16:53:42.19  PAGE  2
==============================================================================

EQ 12   SELECT CODE 22, IS           AVAILABLE FOR USE.

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

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

        AVAILABLE LOGICAL UNITS:    17, 18, 19, 32, 33, 34, 52, 53,
                                    54, 55, 56, 63,

        HP-IB LOGICAL UNITS:         9, 16, 50,

LU  9                       (DEVICE ADDRESS  1 DECIMAL)                   DA 001B

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

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

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

```
HP-IB CONFIGURATION AND STATUS REV. 1840 FRI 29 SEP 1978  16:53:42.19  PAGE  3

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

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

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

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

LU 50                     (DEVICE ADDRESS 15 DECIMAL)                  DA 017B

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

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

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

LETTERS

Dear Sirs,

"I typically have several projects under way simultaneously, and it is very desirable to keep the source and binary files grouped together. This keeps directory lists in order, removes extents, and simplifies back-up procedures.

Enclosed is a transfer file which I use to LIST, PURGE, DUMP, STORE, and COPY FMP files. Communicator readers will probably find it handy, since it eliminates the need for four separate transfer files (to LI, PU, DU & ST) and provides a file-name CO from cartridge to cartridge which eliminates extents.

To implement it, the filenames are added after line 72 of the listing by entering a SET (to 5G) and an IF command for each file. All files are assumed to be ASCII unless preceded by a "%" which indicates a BR type file. (BR files are not "LISTed".) The list of files can be as long as you wish.

One caution: do not change the number of commands between lines 25 and 72 because absolute jumps are made in lines 27, 30, and 67.

Craig B. Spengler
Aim Management Services, Inc.
P.O. Box 21304
Ft. Lauderdale, Fla 33335"

```
:SV,3
:PA,0G,SET PARAMETERS EG ::,COMMAND,FROM-LU,TO-LU,SECURITY-CODE
:IF,1G,NE,0
::
:IF,1G,NE,LI,2
:LL,3G
:DP,LL,3G
:IF,2G,NE,0
::
:IF,3G,NE,0
::
:***********************************************************
:** GLOBALS ARE USED AS FOLLOWS:
:**      1 = COMMAND
:**      2 = LU
:**      3 = CARTRIDGE NUMBER
:**      4 = SECURITY CODE
:**      5 = CURRENT "NAME" OF NAMR
:**      6 = CURRENT "TYPE" OF DATA FILE (AS OR BR)
:**      7 = CURRENT "NAME" INDEX IN LIST
:**      8 = BACKWARD TRANSFER AFTER NAME HAS BEEN ASS1GNED
:**      9 = FORWARD TRANSFER TO NAME ASS1GNMENT
:**     10 = UNUSED
:SE,,,,,,,,1,6
:***********************************************************
:** CALCULATE THE OFFSET FOR THE NEXT NAMR
:CA,8,7G,*,2,+,38
:CA,8,0,-,8G
:** NOW ACTUALLY SET IN THE NAME
:CA,9,7G,*,2,+,39
:IF,1G,NE,0,9G
::
```

```
:***************************************
:** RETURN TO HERE TO TRANSFER CONTROL
:** FIRST ASS1GN THE FILE TYPE
:SE,,,,,,AS
:CA,9,-19P,A,77400B
:IF,9G,NE,22400B
:SE,,,,,,BR
:** TEST FOR PURGE
:IF,1G,NE,PU,3
:SV,0
:1G,5G:4G:2G
:SV,3
:** TEST FOR LIST
:IF,1G,NE,LI,4
:IF,6G,EQ,BR,3
:SV,0
:1G,5G:4G:2G
:SV,3
:** TEST FOR DUMP
:IF,1G,NE,DU,3
:SV,0
:1G,5G:4G:2G,3G,6G
:SV,3
:** TEST FOR STORE
:IF,1G,NE,ST,3
:SV,0
:1G,2G,5G:4G:3G,6G
:SV,3
:** TEST FOR COPY
:IF,1G,NE,CO,3
:SV,0
:ST,5G:4G:2G,5G:4G:3G::-1,6G
:SV,3
:CA,7,7G,+,1
:IF,1G,NE,0,-41
:SV,0
::
:***********************************
:** START THE LIST OF NAMES HERE
:***********************************
:SE,,,,,&1GTVL                 << THIS IS THE FIRST FILE           >>
:IF,1G,NE,0,8G
:SE,,,,,%1GTVL                 << SECOND AND SUBSEQUENT FILES HERE >>
:IF,1G,NE,0,8G
:***************************************************
:** EXIT FROM HERE-RW MAGTAPE ON LU=8
:IF,2G,NE,8
:CN,2G,RW
:SV,0
:** DATA TRANSFER COMPLETED NORMALLY
::
```

USER'S QUEUE

Dear Sirs:

"For those who are bored with BLINK, ired with IDLE and want to add some lift to your lights, the enclosed program (from our "Fun and Games" Department) may be just what you need.

In this program, two separate strings of lights are rotated in opposite directions. This gives them the effect of bouncing off of the ends of the display register, and crashing into each other in the center. If you watch them long enough, your mind will begin to do funny things with the lights. The E and O lights are also alternately alternated on and off.

For maximum effect, turn off all the lights in your computer room and work in the dark like everyone else.

Yours 'till the lights go out.

Roger Jenkins
Dynalectron Corporation
Radar Backscatter Division
P.O. Drawer O
Holloman AFB, New Mexico 88330"

```
ASMB,R,B,L
      NAM BOUNC,1,32767
      ENT BOUNC
BOUNC EQU *
      CLE
      STO
AGAIN EQU *
      LDA PAT1        :: SHIFT
      RAL               :: PATTERN
      STA PAT1            :: ONE
*
      LDA PAT2        :: SHIFT
      RAR               :: PATTERN
      STA PAT2            :: TWO
*
      IOR PAT1        :: PUT THEM TOGETHER
      OTA 1             :: AND OUTPUT TO S REGISTER
*
      LDA PAT1        GET PAT ONE FOR TEST OF OVERFLOW
      CPA =B140001    IS IT TIME TO SWITCH O AND E?
      JMP *+2         YES
      JMP NOSW        NO - DON'T SWITCH O & E.
      CME             SWITCH E
      SOC             OVERFLOW CLEAR?
      JMP *+3
      STO             TURN O ON
      JMP *+2
      CLO             TURN O OFF
*
NOSW  EQU *
      LDA =B164000    :: NUMBER OF TIMES TO
      STA ISZER         :: EXECUTE ISZ TO
      ISZ ISZER           :: DELAY BLINK SO WE
      JMP *-1               :: MORTALS CAN SEE IT
*
      JMP AGAIN       PLAY IT AGAIN, SAM
*
PAT1  OCT 160000
PAT2  OCT 7
ISZER BSS 1
      END BOUNC
```

OPERATING SYSTEM DRIVERS

by Carl Reynolds / HP Rockville, MD.

To the computer neophyte, or even to the experienced "window" programmer (a programmer who makes use of the machine via a window and a deck of cards), input and output (I/O) are often merely natural, "automatic" functions somehow built into the computer. And, when most people discuss a machine's capabilities, they are much more likely to talk about calculation speed or the machine's architecture than the manner in which I/O is accomplished. The result is that we instructors at the Hewlett Packard Eastern Technical Center are often faced with a room full of blank expressions when we casually mention I/O programs known as system "drivers". Drivers are programs written to control I/O devices, and the purpose of this article is to provide a general introduction to these clever and interesting routines. I assume that the reader has at least some familiarity with operating system concepts, but a detailed knowledge of operating systems is not necessary in order to follow this discussion. More detailed, technical presentations on the subject of drivers are planned for future issues of the **Communicator.**

Once upon a time, everyone wrote his own I/O programs by making laborious use of the I/O instructions included in the machine's instruction set. Often it was discovered that it took more time to create a page of text than it took to create the world. Well, with all the difficulty involved, it didn't take long to see the need for standardized and shareable I/O routines. At first, these standardized routines were manifested in card decks to be appended to each user's program deck. Later, they became the largest part (by far) of the first operating systems (Freeman & Perry, 1977). As operating systems became more sophisticated, the proportion of the operating system code devoted to drivers declined. As anyone who has generated an operating system knows, however, drivers still comprise a substantial portion of the operating system.

Obviously, any code that programmers may use but need not write makes life easier for programmers. But aside from improving the quality of life for programmers, why should I/O routines be incorporated into the operating system?

One reason for including I/O routines in an operating system is to facilitate efficient handling of multiple programs. If an I/O driver is included in the operating system, all programs executing concurrently can make use of the driver's code. While one program is busy doing calculations, another can use the driver to write to the line printer. Later, the former program may make use of the same driver code to send its output to the printer. This allows a considerable saving of memory space over what would be required if each program had a separate copy of the driver code appended to it. Of course, if program swapping is involved (moving partially executed programs between memory and disc), swap time and disc space are also saved when the driver code remains memory resident while the programs are swapped in and out.

Savings of memory space are also realized when drivers are written to "talk" to all the devices of a certain type which reside on the system. Thus, we can have one piece of code which can be called upon to communicate with all of the terminals on a system. Likewise, another driver can handle all the line printers configured on the system. Such drivers have to be written cleverly so that they actually modify their own instructions "on the fly" as they address first one piece of equipment and then another.

System resident drivers also provide programmers with the opportunity to write device independent code. When a programmer jots down the following line of FORTRAN code,

```
WRITE(6,100)A,B,C
```

he anticipates the appearance of his data on the list device (logical unit 6). The programmer doesn't have to worry about whether the list device is a character printer or a line printer, is buffered or unbuffered, uses direct memory access (DMA) or not, or any of the other device specific details. The program will work as well next year when a new printer (with driver) is installed as it does today.

Obviously, drivers make use of the machine's input and output instruction set. However, the operating system will not permit user programs to make direct use of the machine I/O instructions. When a FORTRAN programmer, desiring input from a terminal referenced as logical unit 25, writes

```
READ(25,200)X,Y,Z
```

the compiler will generate code to notify the executive of the request to read from terminal 25. The executive, in turn, will cause the driver code for terminals to be executed. When the driver completes, the appropriate modules of the FORTRAN formatter will be called to convert the ASCII, base ten numbers input from the terminal into the computer's base two, floating point representation. Notice that the executive wants to control I/O itself. It will schedule the driver, thank you, and should your program be so foolish as to try to address a device directly (e.g. with an `LIA 14B` — Load into the A register from select code 14 octal — assembler language), the executive will promptly abort your program while mumbling something on the system console about a memory protect error. Why should the executive be so vigilant in protecting I/O as its domain?

For one thing, the executive wants to avoid conflicts. Suppose each program accessed drivers on its own. One program might call the driver while another program was also using the code. Those familiar with assembler language will recall that return addresses are stored in the subroutine (here a driver) entry location. The return point to the program which first accessed the driver would be lost along with at least some of the data being transferred. The preempted program would sit indefinitely like an abandoned lunar landing module. To avoid this possibility, the executive wants to keep track of which program is using any particular device. Of course, the executive will also maintain a list, based on program priority, of programs that are waiting to use each device and its associated driver. Once a program is allowed access to a driver, it will have exclusive use of the driver's code until control returns to the executive. Even if a higher priority program wants to use the driver to talk to a different device, it will have to wait a millisecond or so for the lower priority program to finish what it is doing. This fact sometines surprises people, but remember that in order to keep transfer of program control orderly, the executive has to protect program return point information. Therefore, the program currently using a portion of the driver's code to transfer a byte, say, of data will be allowed to finish using the code, and return, before the driver is turned over to another program's use.

The executive also wants to control I/O so that it can make efficient use of all the system resources, including the CPU and memory as well as I/O devices. When a program requests the executive to start a data transfer, the executive can decide whether or not the transfer will take some time to complete. Most will. In fact, if it took the computer one second, instead of one microsecond, to add two five digit numbers, it would take one and a half hours to punch one byte on a paper tape. Anyway, knowing that a program will have to wait for the data transfer to complete, the executive can schedule another program in the meantime. If the transfer is buffered through system controlled memory, the program waiting for I/O can even be swapped out of memory to the disc to provide memory space for another program. If the executive didn't control I/O, it wouldn't be able to make these decisions which dramatically increase system productivity.

It's important, too, that the executive maintain current information on the status of each I/O device. If the card reader broke while a program was performing unbuffered input (directly into the user program's memory space), the program might be suspended indefinitely in memory (a program cannot be swapped while waiting for unbuffered input). However, if the reader did break, the executive would observe that the device had "timed-out" (failed to respond). The executive would then declare the device device to be "down" and allow any program waiting for the device to be swapped out. Of course, a list of all such programs would be maintained by the executive, so that when the device were fixed and declared "up", the suspended programs might be rescheduled.

Yet another reason for delegating I/O responsibility to the executive is that all drivers must operate with the interrupt system turned off (except for interrupts from privileged devices). The executive is naturally reluctant to allow user programs to turn off the interrupt system because the ability to service interrupts (not interruptions — even the English language is suspended) is the heart of a real time operating system. When, for example, a patient's breathing ceases, you want some program scheduled to alarm the attendant. You don't want the request queued while two disorderlies play Othello (a game) with the interrupt system disabled. Consequently, user programs are not normally permitted to disable the interrupt system, and it follows that user programs must address the executive with I/O requests because the drivers do operate with the interrupt system turned off.

It is a complicated business insuring that access to I/O routines and devices is orderly. Therefore, the operating system must be written to control and coordinate the use of I/O devices. Now let's look at the drivers themselves in more detail.

The first thing to keep in mind about drivers is that, while we talk about device drivers, what we really have are controller drivers. Drivers are written to communicate with the device controllers, and the controllers, in turn, direct the actions of the peripheral devices. Device controllers may be resident on the printed circuit assemblies (PCAs) known as interface cards, or they may be composed of a combination of one or two PCAs and a separate controller "box". In any case, the driver must be written in accord with the controller's requirements. Under RTE, for example, a model 2645 terminal can be plugged into a 12880 or a 12966 interface card. If the terminal is controlled by the 12880 PCA, the required driver is DVR00. However, if the same device is interfaced (silly Webster doesn't realize that 'to interface' is a verb) through a 12966 PCA, DVR05 is required. Either arrangement provides access to the keyboard/CRT functions of the terminal, but with the latter controller and driver, programs can address, as distinct subchannels, the cartridge tape units (CTUs) and certain printers interfaced to the terminal itself. The point is that to write a driver, one must not only "know one's device," but one must also "know one's controller."

Most controllers have at least four basic components: a control line, a flag line, a one bit command register, and a multi bit (byte or word size) data buffer. The faster and more sophisticated the device, the more complex the controller is likely to be. In the case of the typical device, the quiescent (inactive) state is defined as control and command clear, and flag set. To start the reader for an eight bit input transfer, the driver will clear the flag line (you'll see what happens to the flag later), and set the control line. The act of setting the control line causes the command bit also to be set, and when the command bit is set the device will begin reading. At this point, the operating system can go do something else, for it may be some time, as far as the computer is concerned, before the transfer of a byte is accomplished.

When the device completes one cycle, the eight bit buffer on the card will be filled with data, the device will set the flag line to indicate that the transfer is complete, and the device will actually turn itself off by clearing the command bit. These events will generate an interrupt which the executive will recognize (it almost always checks for interrupts after it executes each instruction). Now the executive can recall the driver to bring the data into memory from the controller's buffer, and the driver can decide whether or not more data is needed and whether or not to restart the reader.

Output transfers are handled similarly, except that the data must be moved from memory into the controller's buffer before the output device is started.

If you think of data as food for the computer, we can draw a little analogy. When your body (the machine, miserable as it is) requests food (data), your brain (the executive) sets in motion the chain of commands (the driver) to operate your arms, hands, feet, etc. (peripheral devices) to bring a bite (byte) of food to the mouth (interface data buffer). From the mouth, of course, the food can be brought down the alimentary "bus" to the body.

So, what does a driver look like? If you looked at the code for an I/O driver, how would you know immediately that it was a driver you were looking at?

One obvious clue would be the presence of I/O instructions. Instructions such as STC (set control), CLF (clear flag), LIA (load into A), MIB (merge into B), and OTA (output from A) are only allowed in drivers. However, drivers also have a characteristic code structure which is dictated by their functions.

Thinking back to the example of the reader driver, notice that there were two distinct occasions on which the executive transferred control to the driver. First, the executive called the driver when the I/O request was initially made. Once the device was started, however, control was shifted to other programs currently scheduled. The next time the driver was called was when the device generated an interrupt. The driver then accepted the data and either restarted the device or did not.

The structure of a driver's code reflects these two stellar events in the life of the driver. Almost all drivers will have at least two entry points, one for the initiator section and one for the continuator/completion section. When an I/O request is first made, the executive traces the referenced logical unit number through the system tables to identify the particular physical device addressed and the identity and location of the associated driver. If the device is up and available (i.e. not busy with another transfer), the executive moves some key addresses to be used by the driver into specific base page locations, informs the driver of the number of words to be transferred and the memory address of the buffer, turns off the interrupt system, starts the time-out clock (sets, in a system table, a time-out value to be decremented by tics of the system clock), and transfers control to the initiator section of the driver.

The initiator section must decide from the information provided by the executive which of possibly many actual devices it is to communicate with (imagine a terminal driver on a system which includes twelve terminals). It then must actually write over certain of its own instructions so that the instructions will address the appropriate select code (I/O channel). The initiator must also check the request for legality (e.g. you aren't allowed to read from the punch), reset all operating conditions and flags, and start the device.

When control returns to the executive, the executive will reenable the interrupt system, flag the device as "busy" (in case another program tries to use the device), and transfer control to another program currently scheduled. Later, an interrupt will occur on the select code of the device we just started when the device completes one cycle.

Regardless of what program is being executed at the time of the interrupt, the instruction in the memory location corresponding to the device's select code will be executed. These memory locations are called "trap cells", and memory location 16 is the trap cell for the device plugged into select code 16. We say that control is "vectored" (this is another verb of which Webster is unaware) to the trap cell because, while the content of the trap cell is executed, the program counter register, which points to the next instruction which would have been executed, is not altered.

In the trap cell will be an instruction which transfers control to a module of the executive which handles interrupts. Here the contents of the program counter, as well as of all of the registers, will be saved so that later the executive can return to whatever was being done when the interrupt occurred. The executive will disable the interrupt system, decide which select code just interrupted, look up the appropriate driver, move some addresses useful to the driver into specific base page locations, reset the time-out value for the device, and transfer control to the continuator section, passing along the select code of the interrupting device.

The continuator will have to configure its instructions to address the particular select code involved, either accept and store or send out another unit of data, and decide whether the entire transfer request has been satisfied. If more data is to be handled, the continuator must restart the device and return to the executive in a manner which indicates that the device is still busy and the transfer still incomplete.

If all of the data has now been exchanged, the driver must set the device to its quiescent state, clear the time-out value, and return to the executive in a manner which both informs the executive that the transfer is complete, and provides a count of the number of words transferred. The executive will then clear the busy flag for the device (thus making the device available for use by other programs), restore the contents of all of the machine registers, reenable the interrupt system, and resume execution of the interrupted program.

Actually, most drivers are even more complex than I've indicated. For instance, some drivers will have special tasks to perform on power up, some drivers will use the dual channel port controller (DCPC) for direct memory access (DMA), some drivers will have special provisions for time-out processing (e.g. try the device one more time), and some drivers will have to handle interrupts generated when some devices are brought "on line". There is also a whole class of drivers for privileged devices. Privileged devices are devices which can interrupt at any time, even when the interrupt system is disabled for all ordinary peripherals. Drivers for privileged devices have a third entry point for a privileged section, and, since privileged drivers operate "behind the back" of the executive, they need to be sophisticated and fast in order to handle all the "house-keeping" tasks normally performed by the executive.

Notice that even a simple driver will usually contain pretty fancy code. One driver may be written to address several devices concurrently. For instance, as soon as the continuator section of the terminal driver finishes servicing an interrupt from select code 16, it is possible for it to now service an interrupt from select code 24. Transfers from two or more devices using the same driver may be interleaved.

Of course, fancy code also results from the need to make drivers fast and compact. After all, the less system overhead the better, and the less time the interrupt system is disabled the better.

In future articles we will discuss more of the details of driver structure and operation (including privileged drivers). I hope I've been successful in conveying the general need for, function of, and structure of system drivers. Given the critical functions and sophistication of most drivers, I suspect most users will agree that when you're on the I/O bus, it's a pleasure to leave the driving to HP and RTE.

References

Freeman, D. E., & Perry, O. R. *I/O Design: Data Management in Operating Systems.* Rochelle Park, N.J.: Hayden Book Co., l977.

Hayes, J. P. *Computer Architecture and Organization.* N.Y.: McGraw-Hill, 1978.

Hewlett-Packard Co. *RTE Operating System Driver Writing Manual.* Cupertino, Calif.: Hewlett-Packard Co., l978.

I want to thank the following people for their helpful comments on earlier drafts of this article: Harvey Bernard, Darrell Gordon, Ken Kelley, and Gary McCarney.

NOTES ON DCPC DRIVERS FOR RTE

by Larry B. Smith/HP Albuquerque, N.M.

The RTE Operating System Driver Writing Manual (92200-93005 5/78) is a great improvement over any previous HP documentation on how to write drivers for your RTE system. For a long time many features of RTE have not been documented and resulted in many questions by users who are attempting to write drivers for their system.

One of the most difficult tasks for a user is to generate an RTE system correctly. Due to the many possibilities of how the system can be configured, both hardware and software generation can be very difficult unless the user has generated several systems.

The area I would like to talk about is DCPC (Dual Channel Port Controller for Direct Memory Access (DMA)) usage and the equipment table. The question of assigning DCPC to a device at generation time is a very important one for it can greatly impact the performance of your RTE system.

I've seen many users adding DCPC to a line printer thinking that it will print faster. I've also seen DCPC assigned to CRT's, photo readers, tape punches, plotters, etc. Many users have fallen victim to just the opposite problem: not assigning DCPC to a device that requires it. Have you ever tried to boot-up an RTE system when DCPC was not assigned to the disc driver?

The DCPC drivers that I have looked at assume that the person who generated the system was smart enough to know which driver needs DCPC, which ones never use it, and which ones can allocate DCPC when they need it. However, this isn't the case in many situations.

With very little effort, some code could be added to a driver to check to see if it was assigned DCPC at generation time. A quick check of bit 15 in EQT 4 will tell the driver whether DCPC was assigned at generation time or not.

Let's take the case where a user forgot to assign DCPC to a driver at generation time. The driver could do a quick check of EQT 4 bit 15 and if it is not set the driver could set bit 15 and reject the request. On the next call from RTE, the system will see that bit 15 is set and assign DCPC to the driver before it is called. By rejecting the request, the first time, the driver can inform the programmer of the problem and fix the problem at the same time! Of course, when the system is re-booted, this will occur again, but the operator/programmer will be aware of the problem and can correct the problem during the next system generation.

Now let's take the case where a user assigned DCPC to a driver that never uses it.

The driver, in the initiation section, can do a quick check of EQT 4 bit 15 to see if DCPC is assigned (as stated before). If it is assigned, the driver can clear bit 15 so RTE won't assign a DCPC channel again, thus letting another driver have the DCPC channel that really needs it. This also applies to any driver that can allocate DCPC when it needs it and DCPC should not be assign at generation time.

Assigning DCPC to devices that never use it has a great impact on the performance of the RTE system. With only two DCPC channels to work with, and possibly several devices that require DCPC, some devices will have to wait for a DCPC channel to become available.

Another point I'd like to make concerns DCPC drivers that require DCPC but not for control requests. The mag tape driver is a good example. If DCPC was assigned at generation time it will be allocated to the driver whenever it is called by RTE. However, for any control request (ie: forward space file, rewind, etc.) DCPC is not required and the DCPC channel should be returned to RTE. Why tie up a DCPC channel while waiting for a mag tape to rewind?

I hope that these suggestions may help those who are writing drivers for their RTE system.

SOME NOTES ON THE RTE TIME-OUT FUNCTION

by Del Kittendorf / HP Orlando, Fla

Some devices which are interfaced to HP 1000 systems are capable of returning little or no useful status information to the user and/or operating system. In these cases, it is impossible to determine whether or not the device is up and ready for use. If a user makes an I/O call to such a device, there needs to be some method of detecting the malfunction in a timely manner to prevent the whole system from being hung up waiting for the device to come ready.

The RTE system time-out function was implemented to serve this purpose. The time-out is designed to wait some predetermined number of clock ticks (10ms each) between the initiation of the I/O request and its completion, then printing a diagnostic message on LU 1. The time-out value is set at generation time in the EQT phase (T=nnn), or on-line with the RTE "TO" command `TO, eqt, nnn`. Either way, the value entered is negated and stored in EQT word 14.

During the initiation of an EXEC I/O request, RTIOC picks up the value from EQT14, (if not zero), and places it in EQT15 just prior to transferring to the initiation section of the driver. The driver and device are allowed to process in their normal mode. In the meantime, the initiator will return control to RTE and the clock will begin to run again. On each TBG tick, RTIME's $CLCK routine will do an `ISZ` on each non-zero EQT15. If the `ISZ` produces a zero, processing is passed to $DEVT in RTIOC. $DEVT determines whether the driver will handle its own time-out, or if the system will handle it. If the former, control passes to the driver to process the time-out (See RTE Operating System Driver Writing Manual 92200-93005). If the latter, the system downs the device, suspends the program(s) and alerts the operator with the `I/O TO L#X E#Y S#Z` message.

If the device responds before the time-out period elapses, RTIOC resets EQT14 into EQT15 as before, prior to entering the continuation section. Thus, if the driver starts another I/O transaction, the time-out clock is properly set. If the driver signals completion, RTIOC will zero EQT15 as part of its clean-up.

Privileged drivers have their time-outs handled in much the same way, except interrupt processing bypasses RTIOC, therefore, any time-outs must be set long enough for the longest expected complete I/O operation, not the individual transaction between driver and device.

Time-outs vs. Devices

The following are some recommendations for time-outs and the reasons. Reference is by Hewlett Packard device model number:

HP 2737A, 2748A/B, 2758A PAPER TAPE READERS
HP 2753A/B, 2895A/B PAPER TAPE PUNCHES
HP 2752A, 2754B, 2749A TELEPRINTERS
HP 2600A, 2605A, 2615A, 2616A CRT TERMINALS
HP 264X, 263X TERMINALS
HP 2762A/B PRINTING TERMINALS
HP 2767A LINE PRINTER W/12653A OR 12566B INTERFACE

All of these devices are good examples of peripherals which return little or no useful status information to aid the users program to determine readiness. While time-out values for these devices could be set for two or three character times, it is wiser to set them longer.

For Tape Readers, a 3-5 seconds (300-500 TO value) is needed to allow reading of long leaders. Many users like to set the value still larger for both the readers and punches to allow the operator a few moments to realize there is no tape motion and ready the device before the time-out period expires. This eliminates the "I/O TO" message and the need to "UP" the EQT, etc.

For terminals a value of 30000 (approx. 5 min.), is a good idea to allow the terminal user the opportunity to respond under most conditions.

For the Line Printer, a form-feed from top of form (TOF), takes approx. 1.2 seconds. Therefore, minimum time-out should probably be 2 seconds (TO=200).

HP 2607A, 2613A, 2617A, 2618A, 2631 LINE Printers

As with the HP 2767A, the TOF from TOF is the longest operation. The recommended values are in Table 1.

HP 7970B/E MAG TAPE DRIVES

While the subsystem is able to return sufficient status, if a timeout is desired, the following considerations are apropo:

To rewind 3600 feet of tape at 180 inches/sec. requires 240 seconds, minimum. To allow time for the drive to start, stop and return to load point will add several seconds, therefore, 260 seconds (TO = 26000) is minimum recommended for use at generation. This will allow on-line system back-ups to end of tape on 3600 ft. reels, etc.

Forward space files, records, etc., take the place at 45ips. Sufficient time to handle the longest files in a search operation needs to be considered also. This is one of the reasons a time-out may not be desirable on a mag tape.

HP DISCS WHICH USE THE 13037A/B CONTROLLER

HP 7905A, 7906A, 7920A, 7925A Drives with the advent of the smarter controller and the use of a different technology in the drive, some new features need to be considered. All of the above drives have these features.

Certain disc errors may be recoverable, such as cylinder compare errors, head or sector compare errors and some data errors. These problems are detected in the drive and 13037A Controller. The controller takes no action on its own but rather reports the fact that the error has occurred through the interface to the driver. Should this happen, in the case of the cylinder compare error or head/sector compare error, the driver will ask for a recalibrate operation in the drive. Basically, to do this the driver pulls the heads back to track 0 and attempts to center the heads on the track. For the 7905A, 7906A, and 7920A the typical time for this operation to complete is 800 ms. Within the drive itself is a hardware watchdog timer set to time out if it takes longer than 1250 ms. For the 7925A, the times are 800 ms and 1667 ms, respectively. Should the drive detect a time-out, a drive fault will occur, the "T" LED in the drive will be on. The controller will report a not ready condition to the driver, and of course, the user will then get notification of trouble.

If the recalibrate goes OK, within ten retries, the system will continue, otherwise a not ready condition will be reported.

OTHER PERIPHERALS

Hopefully, by the time the reader has reached this point he/she has a fair understanding of the time-out function in the RTE environment. Other standard peripherals are included in Table I for reference with a brief note to describe the consideration which evoked the particular value chosen. Data acquisition and control peripherals have not been included due to the complexity of the individual items and the many options which might be included. Consultation with the Hewlett Packard Systems Engineer in your local office should help these and other situations not already covered.

Table 1. Standard Accessory Device Time-Outs

H.P. Device	Min. Recommended T.O. Value	Notes
2600A CRT Terminal	30000	1
2607 Line Printer	600	2
2608A Line Printer/Plotter	600	2
2610A Line Printer/Plotter	200	2
2613A Line Printer/Plotter	120	2
2615A CRT Terminal	30000	1
2617A Line Printer	100	2
2618A Line Printer	100	2
2631A Line Printer	300	2
2635A Printing Terminal	30000	1
2640A/B CRT Terminal	30000	1
2641A CRT Terminal	30000	1
2644A CRT Terminal	30000	1
2645A CRT Terminal	30000	1
2647A CRT Terminal	30000	1
2648A CRT Terminal	30000	1
2649A CRT Terminal	30000	1
2737A/B Paper Tape Reader	300-500	1,3
2748A/B Paper Tape Reader	300-500	1,3
2749A Teleprinter	30000	1
2752A Teleprinter	30000	1
2753A/B Paper Tape Punch	300-500	1
2754B Teleprinter	30000	1
2758A Paper Tape Reader	300-500	1,3
2762A/B Printing Terminal	30000	1
2767A Line Printe	200	2
2892A Card Reader	10	4
2895A/B Tape Punch	300-500	1
7210A Plotter	300-350	5
7260A Card Reader	300-500	1
7261A Card Reader	10	4
7900A Disc	20	6
7901A Disc	20	6
7905A Disc	1500	7
7906A Disc	1500	7
7920A Disc	1500	7
7925A Disc	2000	7
7970B/C/E Mag Tape	0	8
9866A Line Printer	50	9
9871A Line Printer	1000	2
9885A Flexible Disc	800	6

NOTES

1 Allow operator response
2 TOF from TOF
3 Allow for leaders
4 Card time and pick retry
5 Longest line
6 Worst case seek
7 Recalibrate
8 Forward space long file
9 New line

FAST REAL TIME I/O UNDER RTE

by John Pezzano
Holloman Air Force Base
New Mexico

One of the limitations of an RTE operating system is the amount of system overhead when doing input/output operations. Because of this overhead, many users prefer to use BCS systems for real-time I/O where fast response or minimum overhead is required. However, the use of BCS prevents one from taking advantages of RTE's multiprogramming capability, prioritizing of tasks, and system integrity. There are a number of alternatives to giving up RTE in favor of BCS. These include privileged drivers and doing your own I/O. A look at each of the methods is in order.

Privileged drivers enable the user to bypass system overhead on interrupt and permit faster response to interrupts than using standard drivers. There are limitations, however. The overhead to call the initialization section of the driver through RTE still exists and the driver completion return to RTE requires using the timeout feature of the operating system to do a timeout return to RTE. The initialization overhead in RTE-II amounts to 1.9 ms in a 21MX-M, 1.5 ms in a 2100 and 1.2 ms in a 21MX-E.

An alternative method where interrupt processing is not required is to use the RTE $LIBR/LIBX calls to shut off interrupts. The user can then do his I/O without the knowledge of the operating system. Another advantage of this is the elimination of an RTE driver which would have to be core resident. The program and its I/O subroutine can be disc resident and only loaded when required. The same I/O slot could be used for a different device and a different program without reconfiguring the whole operating system. Interrupt shutoff is particularly useful for quick I/O when a few instructions are given to a non-standard device without interrupt. An example of this is reading the count of an IRIG board where the only I/O commands are one "CLC" and three "LIAs" which can be handled as fast as issued. The limitations of $LIBR/$LIBX routines include some system overhead and the unavailability of DMA.

If system overhead is an extreme limitation, this can be further reduced if one understands RTE's operations. Since all EXEC and RESIDENT LIBRARY calls generate memory protect which RTE interprets and handles, one can use this capability to good advantage. If the JSB $CIC in the memory protect trap cell were modified to a JSB to a user routine, memory protect could be intercepted. If one used a special illegal operation such as CLC 77B (I/O) or JMP 5 (jump to protected memory) as an interrupt disable request, the special routine could check for this code. If the special request was made, interrupts would be left off and control returned to the calling program. Other interrupts such as parity, DMS violations, EXEC and Library calls, and illegal I/O could be channeled to $CIC. For an RTE-III system in a 21MX-M computer using DMS, interrupts can be shut off in 40 us. However, there are limitations to this. The program shutting off interrupt must restore it upon termination of I/O. Normal memory protect interrupts are slowed down by the additional overhead of the intercept routine. The use of privileged drivers in the system would require a large and therefore slower intercept routine. The intercept routine must be memory resident in RTE-II (it is always memory resident in RTE-C) and in the system map in RTE-III. Also for RTE-III, the dynamic mapping system must be taken into account when checking the interrupt and returning to the interrrupting system.

For RTE-C/II systems, programs using interrupt can modify the trap cell to directly call the program's interrupt handling routine. In RTE-II the program must be memory resident or core locked to prevent being swapped to the disc when interrupt processing is desired. In addition, the privileged driver requirements to save registers, DMA, etc., must be done. In RTE-III, since the system map is enabled on interrupt, this procedure is extremely difficult unless the program is built into the system at generation.

For programs requiring DMA, a dummy driver can be written that requests DMA. When the DMA channel is assigned to the driver, the driver can pass this to the program which can then use the channel as often as needed. This is particularly useful when multiple read or write requests must be made using DMA. For example, if 100K words must be read the program can obtain a channel and do ten requests for 10K words before returning the channel. As each block is read, the program disposes of it and reinitiates DMA without the system overhead necessitated by recalling the driver ten times. Limitations to this include the required handling of timeout, driver program communications, and for RTE-III, dynamic mapping considerations.

The capability to do fast real-time I/O under RTE exists. The use of non-standard techniques may be required by the advantages of prioritizing, system protection and the capability to develop and process on one CPU without changing operating systems is well worth the effort.

EDITOR'S NOTE: The supported way to do fast input/output in RTE is through a privileged driver. Furthermore, $CIC in RTE-IV is more complicated than in RTE-II or RTE-III and care must be taken when circumventing $CIC to avoid disrupting the parity handling algorithm. HP cannot assume liability for the information in this article and cannot assume liability for system integrity in systems where user programs modify $CIC or the trap cells or execute input/output instructions directly.

OPERATIONS MANAGEMENT

LARGER IMAGE PROGRAMS IN RTE-IV

by Todd Field/HP Data Systems Division

Ok folks, time for a short quiz. You have been promised larger program space in RTE-IV. You buy the upgrade kit, upgrade from RTE-III, put .DBRN into SSGA, add DS/1000 and/or RJE/1000 and notice that you still only have a 15 page partition for IMAGE programs. What do you do?

A. Shoot your local salesman (before he has a chance to give you a copy of this article)

B. Return the CPU to Cupertino

C. Buy a machine from a competitor

D. Give up

E. Move .DBRN to Table Area I

F. None of the above

A hint: the answer is NOT A, D or F. Right — the answer is E. To understand why moving .DBRN to Table Area I will increase your program space, consider what .DBRN does and how it is mapped into your program space.

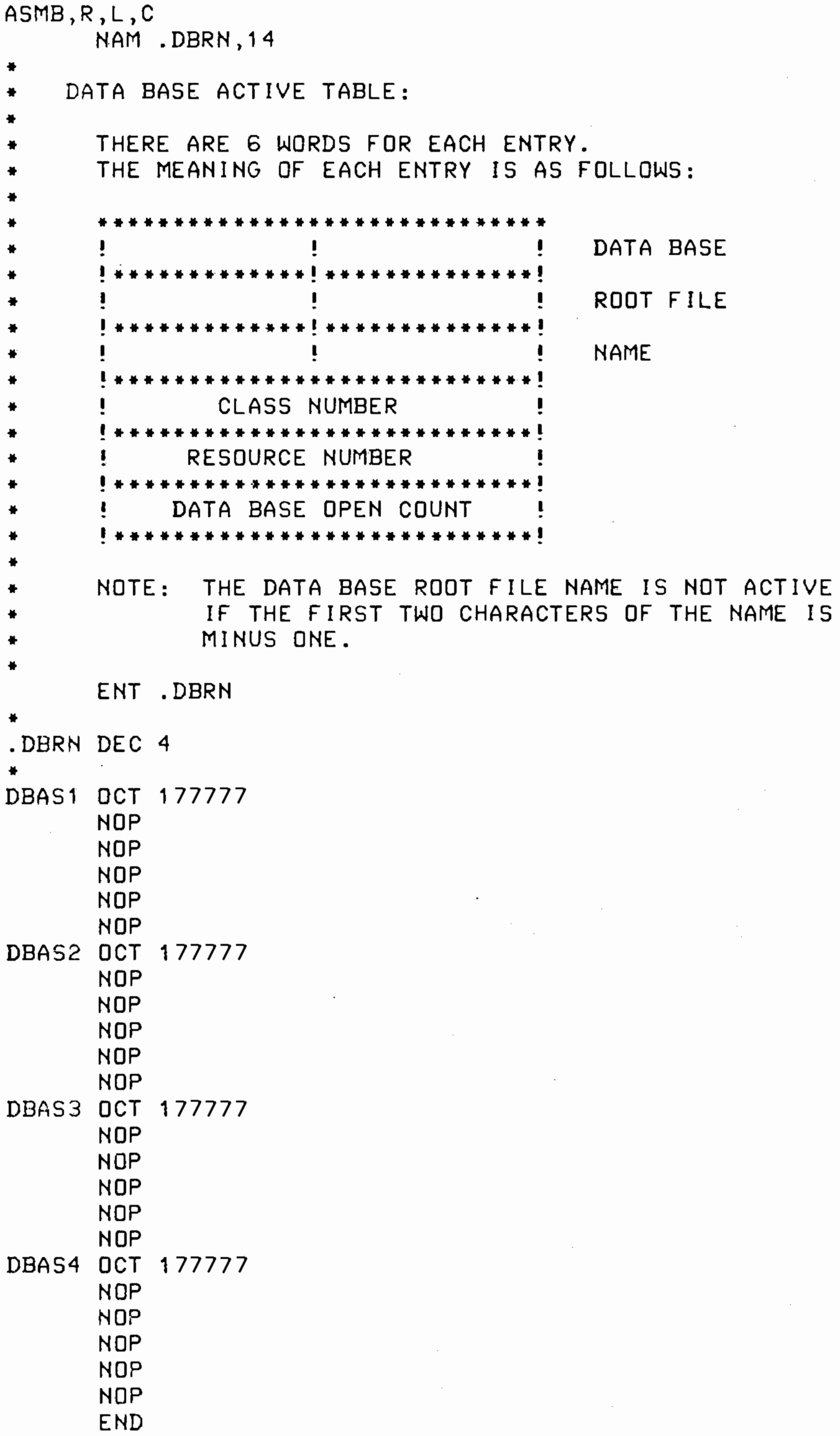
.DBRN (Data Base Activity Table) is used to indicate the name of the data bases currently open in mode 2, the class numbers of the volatile data (from the root file) stored in system available memory (SAM), the resource numbers used by DBLCK and DBULK to lock and unlock data bases and a count to indicate the number of users currently using each data base open in mode 2.

There are six words for each entry. There is a maximum of four entries in this table. The format of .DBRN is described in figure 1.

Note that the data base root file name is not active if the first word (the first two characters) of the name is minus one.

This module works for up to four data bases which are open in mode 2. If a new version of .DBRN is coded with more entries and relocated after the IMAGE library in the system generation, this restriction can be overcome.

To allow the IMAGE routines to share .DBRN, this module must be accessable to all IMAGE programs. Lets examine figure 2, a possible user map configuration under RTE-IV.



```
ASMB,R,L,C
      NAM .DBRN,14
*
*   DATA BASE ACTIVE TABLE:
*
*     THERE ARE 6 WORDS FOR EACH ENTRY.
*     THE MEANING OF EACH ENTRY IS AS FOLLOWS:
*
*     ******************************
*     !             !              !   DATA BASE
*     !*************!**************!
*     !             !              !   ROOT FILE
*     !*************!**************!
*     !             !              !   NAME
*     !****************************!
*     !       CLASS NUMBER         !
*     !****************************!
*     !     RESOURCE NUMBER        !
*     !****************************!
*     !    DATA BASE OPEN COUNT    !
*     !****************************!
*
*     NOTE:  THE DATA BASE ROOT FILE NAME IS NOT ACTIVE
*            IF THE FIRST TWO CHARACTERS OF THE NAME IS
*            MINUS ONE.
*
      ENT .DBRN
*
.DBRN DEC 4
*
DBAS1 OCT 177777
      NOP
      NOP
      NOP
      NOP
      NOP
DBAS2 OCT 177777
      NOP
      NOP
      NOP
      NOP
      NOP
DBAS3 OCT 177777
      NOP
      NOP
      NOP
      NOP
      NOP
DBAS4 OCT 177777
      NOP
      NOP
      NOP
      NOP
      NOP
      END
```

Figure 1.

TWO POSSIBLE MEMORY CONFIGURATIONS DESCRIBED BY USER MAP

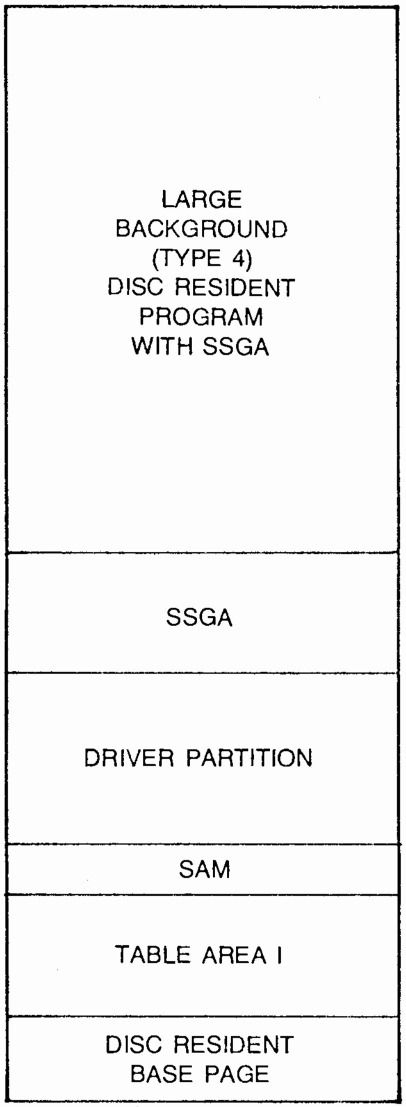


Figure 2

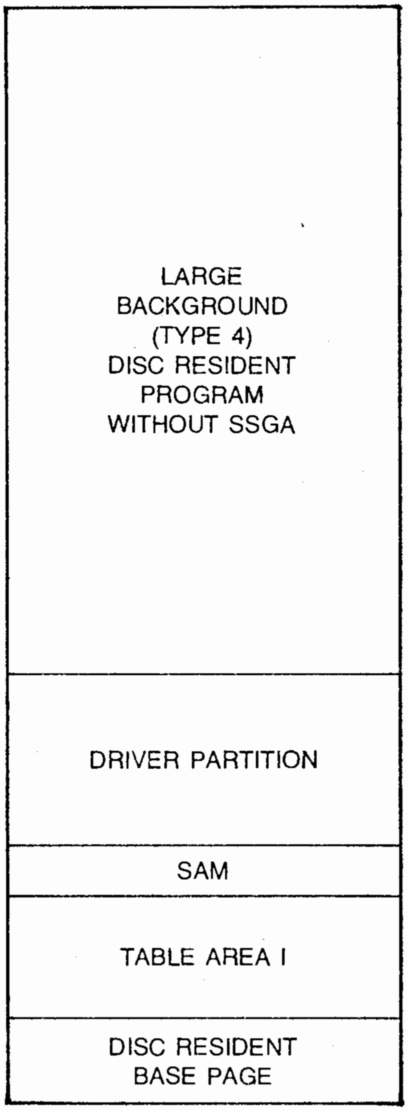


Figure 3

.DBRN must be put somewhere in this space. It cannot go anywhere in the user program space for the reason given in the preceeding paragraph. The driver partition is unavailable for the same reason. Base page is a very precious commodity in RTE-IV and furthermore there is no easy method to move a module to base page during generation. By the process of elimination, SSGA and Table Area I are left. The preferred place for all subsystem modules is, of course, SSGA and .DBRN should normally be put here. If, however, SSGA contains many other modules and .DBRN will not cause Table Area I to need an additional page, it is perfectly acceptable to move .DBRN to Table Area I. This means that programs using .DBRN need not have SSGA mapped into their partition, freeing up this area for additional program space as shown in figure 3.

A word of caution. Table Area I was not designed to be used as common and this method is not guarenteed to work in future revisions. Furthermore, if .DBRN causes Table Area I to cross a page boundry, every program in the system will lose a page of program space. If, however, you are in the circumstance described above, moving .DBRN will allow you to write larger IMAGE programs in RTE-IV.

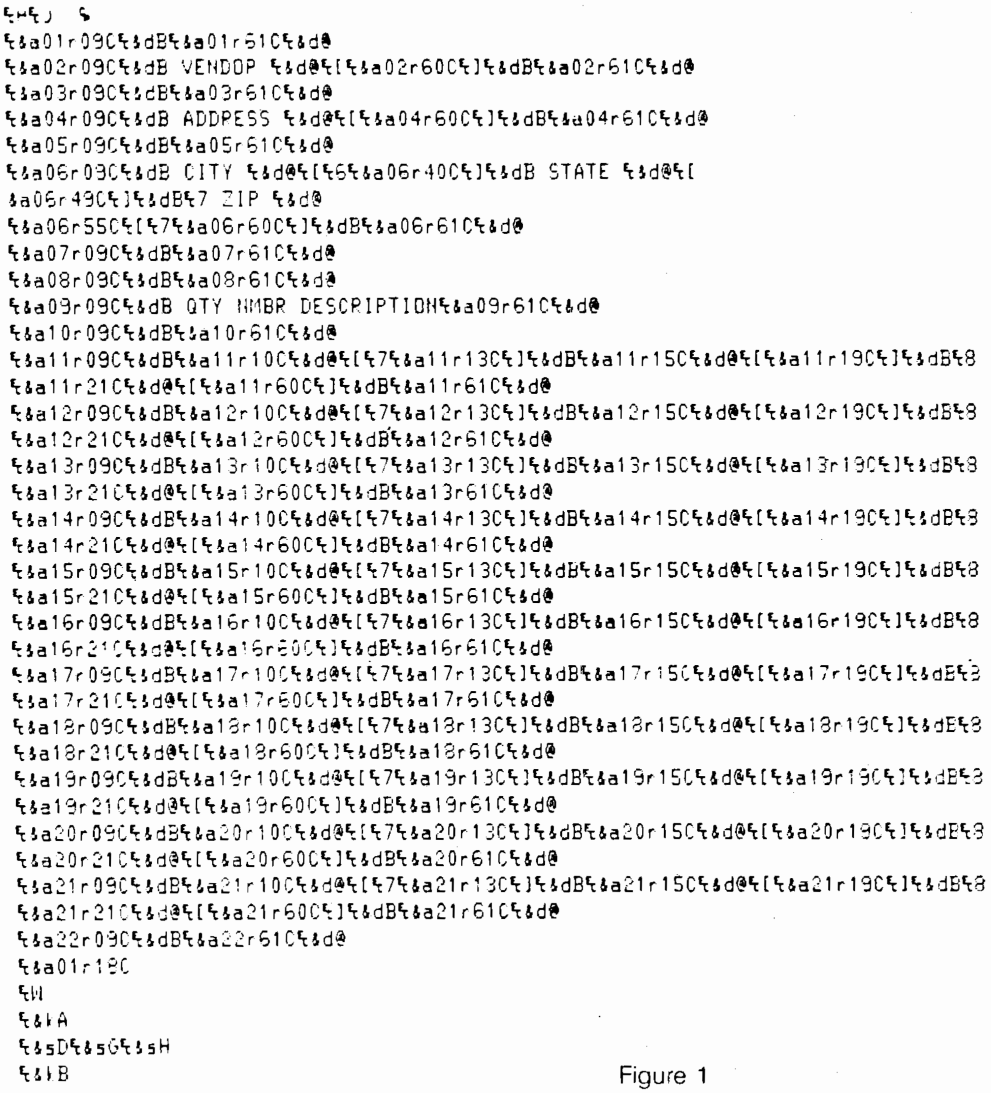
OEM CORNER

SOFTWARE FOR THE 2645 TERMINAL

by P. Alex Swartz/Computer Systems Consultants
Tucson, Arizona

The 2645 has proven itself to be an excellent terminal. The white on black display is easy on the eyes, the higher baud rates make terminal sessions more productive and increase throughput and it has good reliability. Soft keys and internal switches to alter communications modes, as well as options like alternate character sets, make it highly versatile.

In an effort to make the 264X series of terminals easier to use, my company has developed a software aid for screen formatting — a subroutine named X264X. Screen formatting is not difficult. The instruction manual gives a clear description and the ideas presented are not complicated at all. The hitch is in the method that must be used. Even a simple form requires hundreds of characters that include ESC and lower case characters. The sum total of these characters, when viewed from any distance, bears no resemblance to the desired form. Figure 1 is the 2645 code required to produce a simple order-entry form. The form is in inverse video with the "fill in the blanks" part in normal video. All inverse video areas are protected and some of the fill in areas are numeric-only fields. Figure 2 is what the form would look like on a 2645.



```
␛H␛J  ␛
␛&a01r09C␛&dB␛&a01r61C␛&d@
␛&a02r09C␛&dB VENDOP ␛&d@␛[␛&a02r60C␛]␛&dB␛&a02r61C␛&d@
␛&a03r09C␛&dB␛&a03r61C␛&d@
␛&a04r09C␛&dB ADDPESS ␛&d@␛[␛&a04r60C␛]␛&dB␛&a04r61C␛&d@
␛&a05r09C␛&dB␛&a05r61C␛&d@
␛&a06r09C␛&dB CITY ␛&d@␛[␛6␛&a06r40C␛]␛&dB STATE ␛&d@␛[
&a06r49C␛]␛&dB␛7 ZIP ␛&d@
␛&a06r55C␛[␛7␛&a06r60C␛]␛&dB␛&a06r61C␛&d@
␛&a07r09C␛&dB␛&a07r61C␛&d@
␛&a08r09C␛&dB␛&a08r61C␛&d@
␛&a09r09C␛&dB QTY NMBR DESCRIPTION␛&a09r61C␛&d@
␛&a10r09C␛&dB␛&a10r61C␛&d@
␛&a11r09C␛&dB␛&a11r10C␛&d@␛[␛7␛&a11r13C␛]␛&dB␛&a11r15C␛&d@␛[␛&a11r19C␛]␛&dB␛8
␛&a11r21C␛&d@␛[␛&a11r60C␛]␛&dB␛&a11r61C␛&d@
␛&a12r09C␛&dB␛&a12r10C␛&d@␛[␛7␛&a12r13C␛]␛&dB␛&a12r15C␛&d@␛[␛&a12r19C␛]␛&dB␛8
␛&a12r21C␛&d@␛[␛&a12r60C␛]␛&dB␛&a12r61C␛&d@
␛&a13r09C␛&dB␛&a13r10C␛&d@␛[␛7␛&a13r13C␛]␛&dB␛&a13r15C␛&d@␛[␛&a13r19C␛]␛&dB␛8
␛&a13r21C␛&d@␛[␛&a13r60C␛]␛&dB␛&a13r61C␛&d@
␛&a14r09C␛&dB␛&a14r10C␛&d@␛[␛7␛&a14r13C␛]␛&dB␛&a14r15C␛&d@␛[␛&a14r19C␛]␛&dB␛8
␛&a14r21C␛&d@␛[␛&a14r60C␛]␛&dB␛&a14r61C␛&d@
␛&a15r09C␛&dB␛&a15r10C␛&d@␛[␛7␛&a15r13C␛]␛&dB␛&a15r15C␛&d@␛[␛&a15r19C␛]␛&dB␛8
␛&a15r21C␛&d@␛[␛&a15r60C␛]␛&dB␛&a15r61C␛&d@
␛&a16r09C␛&dB␛&a16r10C␛&d@␛[␛7␛&a16r13C␛]␛&dB␛&a16r15C␛&d@␛[␛&a16r19C␛]␛&dB␛8
␛&a16r21C␛&d@␛[␛&a16r60C␛]␛&dB␛&a16r61C␛&d@
␛&a17r09C␛&dB␛&a17r10C␛&d@␛[␛7␛&a17r13C␛]␛&dB␛&a17r15C␛&d@␛[␛&a17r19C␛]␛&dE␛3
␛&a17r21C␛&d@␛[␛&a17r60C␛]␛&dB␛&a17r61C␛&d@
␛&a18r09C␛&dB␛&a18r10C␛&d@␛[␛7␛&a18r13C␛]␛&dB␛&a18r15C␛&d@␛[␛&a18r19C␛]␛&dE␛8
␛&a18r21C␛&d@␛[␛&a18r60C␛]␛&dB␛&a18r61C␛&d@
␛&a19r09C␛&dB␛&a19r10C␛&d@␛[␛7␛&a19r13C␛]␛&dB␛&a19r15C␛&d@␛[␛&a19r19C␛]␛&dB␛3
␛&a19r21C␛&d@␛[␛&a19r60C␛]␛&dB␛&a19r61C␛&d@
␛&a20r09C␛&dB␛&a20r10C␛&d@␛[␛7␛&a20r13C␛]␛&dB␛&a20r15C␛&d@␛[␛&a20r19C␛]␛&dB␛3
␛&a20r21C␛&d@␛[␛&a20r60C␛]␛&dB␛&a20r61C␛&d@
␛&a21r09C␛&dB␛&a21r10C␛&d@␛[␛7␛&a21r13C␛]␛&dB␛&a21r15C␛&d@␛[␛&a21r19C␛]␛&dB␛8
␛&a21r21C␛&d@␛[␛&a21r60C␛]␛&dB␛&a21r61C␛&d@
␛&a22r09C␛&dB␛&a22r61C␛&d@
␛&a01r18C
␛W
␛&kA
␛&sD␛&sG␛&sH
␛&kB
```

Figure 1

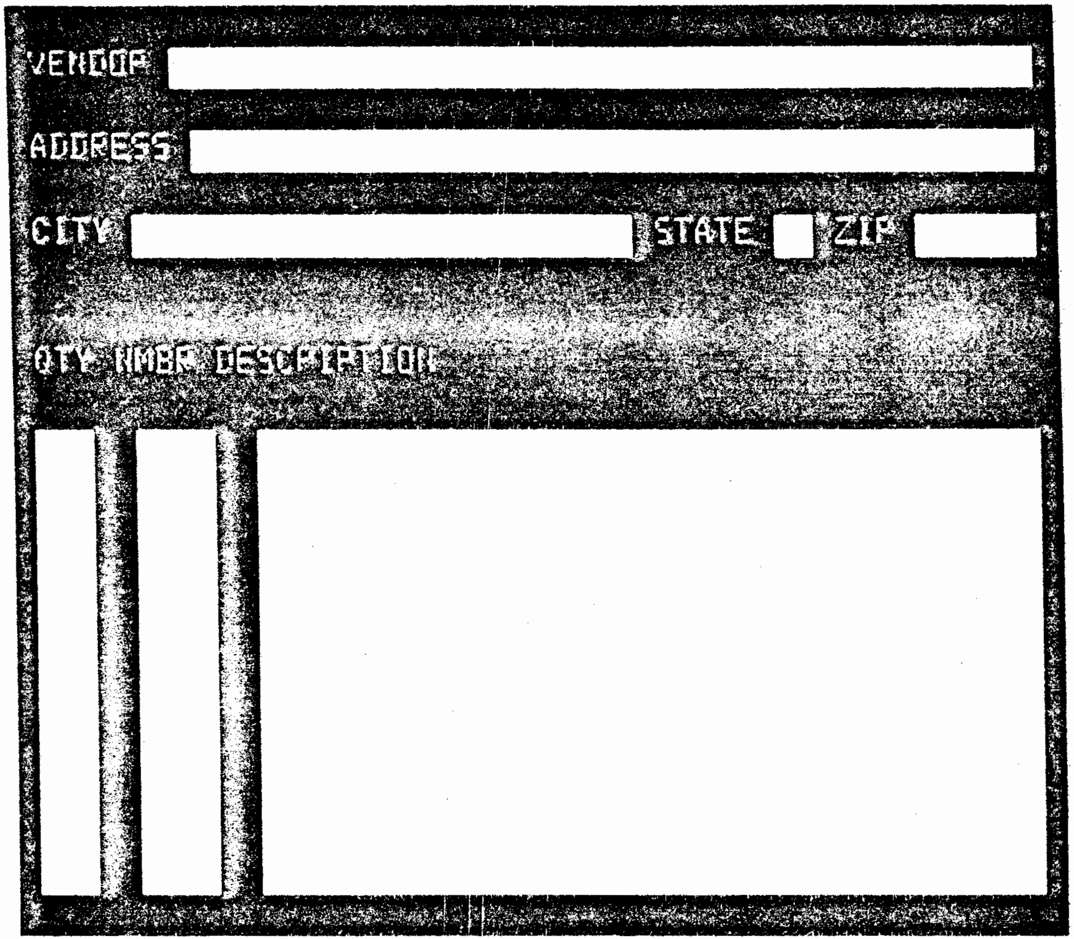


Figure 2

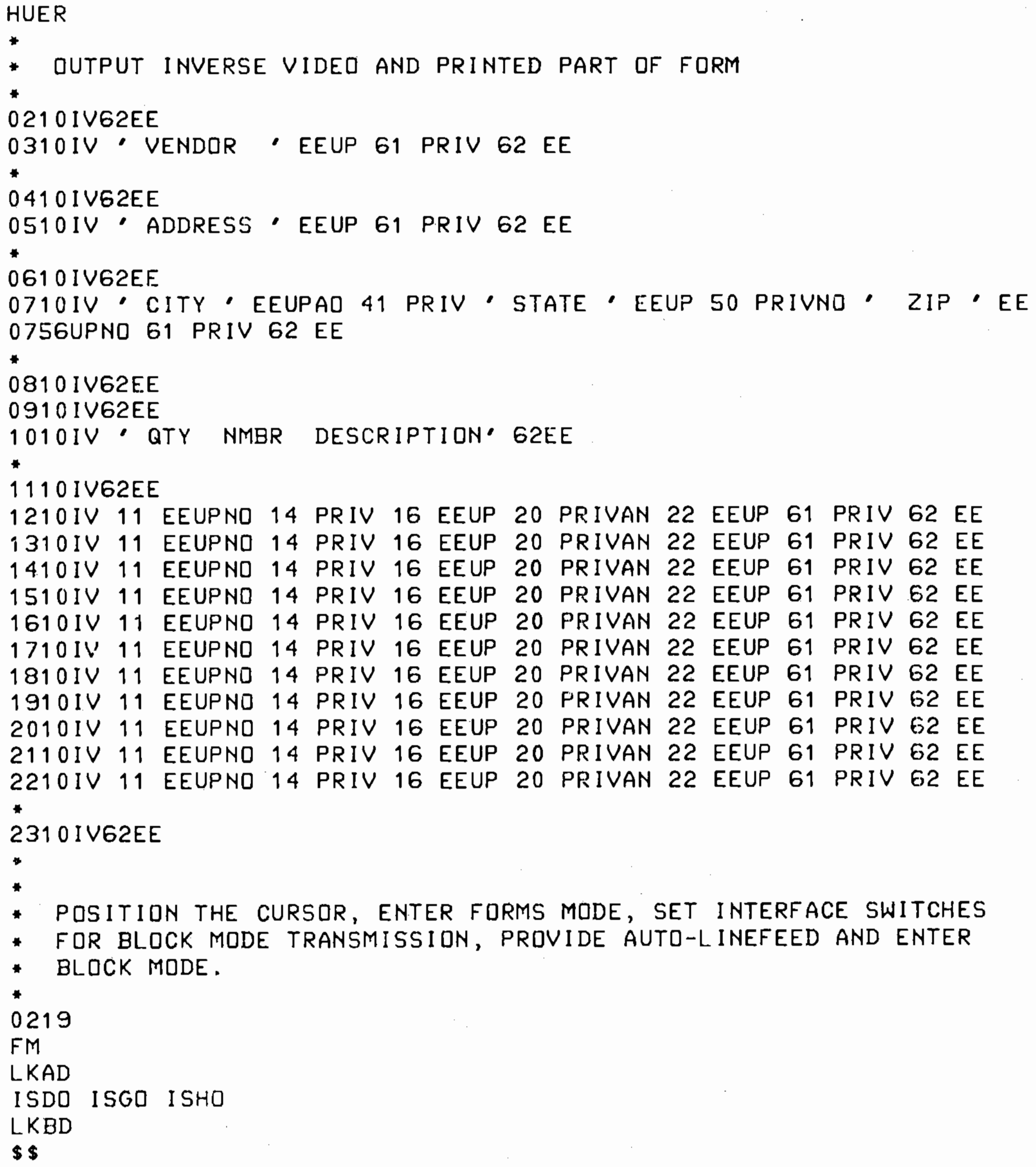
In developing such a form, many problems are encountered. First, one has to find the mistakes. A missing or wrong character will destroy at least a section of the form. It is not so simple to determine where a mistake is in a mass of characters like Figure 1. When a mistake is found, the problem of editing arises. The editing terminal must support lower case and ESC presents a special difficulty — it may not print and it may affect the operation of the terminal. If a 264X terminal is used, the "DISPLAY FUNCTIONS" can be used. Listing the form for documentation purposes presents the same lower case and ESC character problems.

Further problems occur months later if the form needs to be modified. If the boss says he wants the form changed so that the inverse video around "CITY" ends flush with the "VENDOR" and "ADDRESS" fields and he wants a fourth column for price added and he wants the whole form shifted right 5 spaces so it's centered better, then a major project is at hand. For the form in Figure 1, every column address number needs changing and you have to splice in the new field, hoping that line lengths won't increase to a size unacceptable to the compiler or the form output program.

There are a few other problems that aren't evident with this form, such as how one dynamically changes a form under program control, or having the input to one form call up another. These are more complicated problems that require examination for each project but a good general solution to forms generation will be a great deal of help in solving them.

The above problems prompted the development of X264X. The basic principles call for a symbolic language that can be translated by an interpreter and for using only upper case ASCII characters for ease of editing. X264X recognizes a unique 2 letter mnemonic for each feature of the terminal and translates this to the proper code to send to the terminal. Therefore, instead of working with "ESC&dB" to turn on inverse video, one uses "IV". "HU" sends the cursor home, "ER" erases the screen, "FM" invokes forms mode, and so on. Cursor motion is controlled by 2 consecutive 2 digit numbers, indicating absolute line and column number, or one 2 digit number, indicating current line and new column. "1332" would mean move to line 13, column 32. There are mnemonics for every terminal function, including soft key programming, latching key control, internal interface switch control, display enhancements, cursor control, display manipulation, alternate character sets and tabing.

Figure 3, when translated by X264X, produces the same screen form as Figure 1. In fact, Figure 1 is X264X's translation of Figure 3. Lets examine a few lines of Figure 3. Line 1 says "home the cursor up and erase the screen". Line 5 says "move to line 2, col 10, turn on inverse video, move over to col 62 and end the enhancement (turn off inverse video)". Line 6 says "move to line 3, col 10, turn on inverse video, print 'VENDOR' (which will appear in inverse video), turn off inverse video, start an unprotected field, extend it to col 61, turn field protect back on, turn on inverse video, move to col 62 and turn inverse video off". Line 20 defines a "numeric only" field (NO) and also uses the (AN) mnemonic to get the terminal back into alpha-numeric mode.



```
HUER
*
*  OUTPUT INVERSE VIDEO AND PRINTED PART OF FORM
*
0210IV62EE
0310IV ' VENDOR  ' EEUP 61 PRIV 62 EE
*
0410IV62EE
0510IV ' ADDRESS ' EEUP 61 PRIV 62 EE
*
0610IV62EE
0710IV ' CITY ' EEUPAO 41 PRIV ' STATE ' EEUP 50 PRIVNO '  ZIP ' EE
0756UPNO 61 PRIV 62 EE
*
0810IV62EE
0910IV62EE
1010IV ' QTY  NMBR  DESCRIPTION' 62EE
*
1110IV62EE
1210IV 11 EEUPNO 14 PRIV 16 EEUP 20 PRIVAN 22 EEUP 61 PRIV 62 EE
1310IV 11 EEUPNO 14 PRIV 16 EEUP 20 PRIVAN 22 EEUP 61 PRIV 62 EE
1410IV 11 EEUPNO 14 PRIV 16 EEUP 20 PRIVAN 22 EEUP 61 PRIV 62 EE
1510IV 11 EEUPNO 14 PRIV 16 EEUP 20 PRIVAN 22 EEUP 61 PRIV 62 EE
1610IV 11 EEUPNO 14 PRIV 16 EEUP 20 PRIVAN 22 EEUP 61 PRIV 62 EE
1710IV 11 EEUPNO 14 PRIV 16 EEUP 20 PRIVAN 22 EEUP 61 PRIV 62 EE
1810IV 11 EEUPNO 14 PRIV 16 EEUP 20 PRIVAN 22 EEUP 61 PRIV 62 EE
1910IV 11 EEUPNO 14 PRIV 16 EEUP 20 PRIVAN 22 EEUP 61 PRIV 62 EE
2010IV 11 EEUPNO 14 PRIV 16 EEUP 20 PRIVAN 22 EEUP 61 PRIV 62 EE
2110IV 11 EEUPNO 14 PRIV 16 EEUP 20 PRIVAN 22 EEUP 61 PRIV 62 EE
2210IV 11 EEUPNO 14 PRIV 16 EEUP 20 PRIVAN 22 EEUP 61 PRIV 62 EE
*
2310IV62EE
*
*
*  POSITION THE CURSOR, ENTER FORMS MODE, SET INTERFACE SWITCHES
*  FOR BLOCK MODE TRANSMISSION, PROVIDE AUTO-LINEFEED AND ENTER
*  BLOCK MODE.
*
0219
FM
LKAD
ISDO ISGO ISHO
LKBD
$$
```

Figure 3

X264X went thru several stages of refinement and several useage conventions were set up before it became a truly useful tool. One addition is the ability to offset an entire form by specifying an offset line and column number that is added to all absolute cursor positioning commands. Thus, when the boss says "move it right 5 spaces", adding "OS0106" to the front of the form file is all it takes. (Instead of referencing coordiantes from 0,0, they are referenced from 1,1 because this makes it easier to transcribe layouts from paper) Another helpful feature is a debug mnemonic which places a pause of user specified length between each command. This allows a programmer to watch the form coming out in slow motion and see exactly where a mistake occurs. A third useful feature allows internal conversion, so X264X puts the translation in a buffer and returns it to the calling program instead of outputting it to a terminal. This allows the actual code needed to generate a form to be stored in a disk file, which, when dumped to a terminal, will produce the form directly.

We have determined that it is very beneficial to have a single program or subroutine that can be passed a file name and logical unit number and have it pass the information to X264X. This reduces the amount of programming for each project and allows the use of fully debugged output software. It also allows useage of comment lines in the form file, as the starred lines in Figure 3 are. These are ignored by the subroutine and are never seen by X264X. The comments are invaluable for documentation. It has also been found that having one line of code in the form file correspond to one line of the form is a good idea.

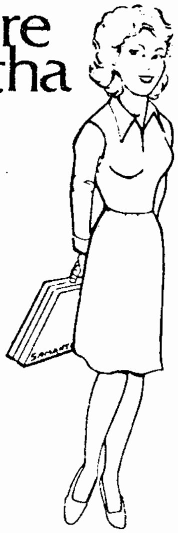
Using X264X, it is a simple matter to produce complex forms. The screen is layed out on something like a coding sheet, where every character location on the screen is represented by a cell. This layout can be transcribed directly into X264X mnemonics, line for line, by anyone who knows the terminals capabilities. The resulting form files are all upper case ASCII and can be edited like any other ASCII file. Comment lines and the inherent readability of the mnemonics make it easy to modify a file months after it has been in use and it can be done just as easily by a different programmer.

X264X is written in assembler for HP 1000 Series computers. It occupies less than 700 words of memory and requires no special subroutines. It buffers its I/O and uses REIO for output. It can be used with any 264x terminal. It will operate thru DVR00 or DVR05 and can be used at any baud rate with no detectable slowing of the terminal.

X264X, along with some utility software, is available from Computer Systems Consultants, 3732 N. Cherry, Tucson, AZ, 85719, 602-326-5482. Relocatable binary is provided on paper tape or HP cassette for a one time charge of $295. The instruction manual is available seperatly for $10, which can later be applied to software purchase.

BIT BUCKET

Software Samantha



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

Dear Samantha,

"When doing RTE-III system generations, I have noticed that the size of the operating system generated depends on the order in which I/O drivers are loaded. Can you give me an explanation for this phenomenon? Can you also tell me if there is any way, other than trial and error, of determining the best way in which to load drivers to obtain the smallest system?

Yours truly,
Linda Steeves
Senior Systems Analyst
Maritimes Weather Office
Environment Canada
P.O. Box 5000
Bedford, Nova Scotia BON 1B0"

Dear Linda,

This is a good question. I would like to discuss drivers and system generation, both in RTE-III and RTE-IV.

In RTE-III, the size of the system could be influenced by the order of the drivers if you were using current page linking. Depending on how many drivers crossed page boundaries, you could have more or less words for links, therefore causing a difference in size. This is magnified by the fact that the next area following the system area, the memory resident library, is aligned on a page boundary. If the number of link words increased enough to push this area over into a new page, you would see a page difference in length.

In RTE-IV, with the use of driver partitions, the logical size of the system would not be affected by the order of the drivers. However, the number of links used by the drivers is affected. This is some concern to the user because the driver links are allocated from the end of the system communication area down. Links used for the system or user programs are allocated from the low end of base page up. The more links used by the drivers, the fewer that are available for these other uses.

The algorithm that RTE-IV uses to allocate drivers to driver partitions is that of "first fit". This means that it goes through the list of drivers looking for one that will fit in the partition's remaining space. If it finds none, it uses the first of the remaining drivers to start a new driver partition and begins searching the list again for a driver to fit in the remaining space of that partition. Therefore, the order that the drivers are relocated in the generation is not necessarily the order in which they will appear in the partitions.

As to the method you can use to determine the best way to load drivers, the answer is that a little trial and error is necessary. After the first generation has been done, you can see from your generation listing what the size of the drivers are, and if you have specified MAP LINKS, you also know how many links they used. You can use this information to rearrange the drivers in a more efficient order. If a driver spans a page boundary, it is not always clear where the best place to cross the boundary is. If there are many drivers that fit this description in your system, some trial and error manuevering may be necessary. However, a good first step, and in some systems, last step, is to order the drivers to minimize crossing of page boundaries or make sure your largest driver starts close to a page boundry.

Dear Samantha,

"I have one problem with RTE-II. To save ID segments, I :SP'd my programs in type 6 files. This procedure works OK unless the first program schedules a second program, etc. In this case, a procedure file must be generated to place both programs in the system by :RPing them. An alternative is to add the programs as permanent programs using the LOADR.

However, if one decides to add the programs to the system using the On-line GENERATOR, several problems occur:

1. All subroutines within the relocatable program are added to the disc resident library. This creates problems if two different programs use the same subroutine name. A GEN 08 error occurs, and the second subroutine entry point will replace the entry of the first subroutine.

   Is there any way to make subroutines local to their Fortran or Assembler main?

2. In the event that the main program has local COMMON, this COMMON becomes background COMMON, thereby reducing the background program area, and interfering with any programs trying to make legitimate use of background COMMON.

   Is there any way to avoid this problem?

Very truly yours,
Gene Olig
Project Engineer
Giddings & Lewis Machine Tool Company
P.O. Box 590
Fond Du Lac, Wisconsin 54935"

Dear Gene,

This seems to be my issue for generator questions.

Unfortunately, there is no way to make subroutines local to their main during generation, nor to avoid using background COMMON rather than local COMMON during generation. To do so would require a major restructuring of the GENERATOR to allow it use local symbol tables rather than just one large symbol table as is the case now. You can prevent the subroutines from being included in the disc resident library by making the subroutines type 8 programs (used to satisfy externals, then disgarded). If you are having GEN 08 or COMMON problems in your generation, your best bet is to load the offending programs on-line using the LOADR.

Thank you for writing. I passed your suggestions on to the lab and they are looking in to ways to alleviate your problems.

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

BIT BUCKET

CONTROLLING 'PROGXX ABORTED' MESSAGES IN RTE-III

by Darrell Gordon/HP Rockville, MD.

If you are one of the many HP users doing forms data display on your 2645 system console, or are doing graphics data representation with a 2647A or 2648A graphics console as LU 1, you will find annoying 'PROGXX ABORTED' messages right on top of your precious display. As we all know, any time a user does a 'OF,PROGX,N' (where N is 1 or 8) from any terminal, this message shows up on the system console, sometimes right over our displays. Also, when a user program causes a system error such as Dynamic Mapping, Memory Protect, etc., the same message occurs. The latter messages are actual system messages and must not be ignored if normal processing is to continue. However messages of the former type are informational only, and in practically all cases, can be ignored. The assembly listing that follows is a small program which can be used to 'turn off' system output of the abort message. It can later be run to restore the message processor in RTE. Please note that if the messages are turned off, only the 'PROGXX ABORTED' message is eliminated. In the event of a system error message (MP, DM, SC, etc.), that message (i.e. `MP PROGXX 41343`) will still appear on LU 1 and you will still know that a program abnormality exists.

HOW IT WORKS . . .

In the EXEC part of memory, there exists an entry point, '$ABRT', which processes programs which have been hard aborted. Within $ABRT, a call is made to the routine '$SYMG' which is responsible for outputting the abort message (the call is in the form: JSB $SYMG). The program listed below, ABMSG, recovers the JSB $SYMG word from $ABRT, saves it, and writes a nop (all zeros) over its location. In this way, any calls to $SYMG that $ABRT might make are avoided and program execution falls right through the NOP where execution continues. Since ABMSG does an EXEC 6 on itself with IOPTN = 1, its resources (in this case the saved JSB $SYMG code) are not lost when it terminates.

To use ABMSG, schedule it under RTE with zero as a first parameter to turn off the message and with a non-zero first parameter to restore the messages. Remember that if you run ABMSG from FMGR, MTM places your terminal LU in the first parameter and, since it is not zero, attempts to restore the message to the system. If the user attempts to restore the message before it has been disabled, or turn off the message without it first being on, a sequence error results and the program terminates.

This program has been used successfully many times and hopefully, it will be useful to you.

PROGRAM ABMSG

```
ASMB,R,L,C
      NAM ABMSG
      ENT ABMSG
      EXT $LIBR,$LIBX,$ABRT,EXEC,RMPAR
ABMSG NOP
************************************************************************
*
* THIS PROGRAM IS DESIGNED TO ELIMIATE 'PROGX ABORTED' MESSAGES FROM
*  THE SYSTEM CONSOLE WHERE GRAPHICS DISPLAYS MIGHT BE BEING GENERATED.
*  IT IS A CONTRIBUTED PROGRAM ONLY, AND NOT SUPPORTED BY HEWLETT-PACKARD.
*
*************************************************************************
START JSB RMPAR     GET SCHEDULING PARAMETERS
      DEF *+2
      DEF BUFR
      LDA BUFR      GET FIRST WORD
      SZA           TURN OFF OR ON?
      JMP RPLAC     ON-GO TO REPLACEMENT CODE
      LDA JS$AB     GET WORD
      SZA           BETTER BE ZERO FOR TURNING OFF-
      JMP SEQER     ELSE SEQUENCE ERROR- GO PROCESS
```

BIT BUCKET

```
      JSB $LIBR      GO PRIVILEGED
      NOP            (NOT RE-ENTRANT SUBROUTINE)
      LDA $ABRT+23   GET 'JSB $SYMG' CODE
      STA JS$AB      STORE FOR LATER
      CLA            FORM NOP IN 'A'
      STA $ABRT+23   AND OVER-WRITE JSB
      JSB $LIBX      TURN ON INTERRUPT SYS
      DEF *+1
      DEF *+1
      JSB EXEC
      DEF RETRN
      DEF SIX        TERMINATE
      DEF ZERO        SELF AND
      DEF ONE          SAVE RESOURCES
RETRN JMP START      START HERE WHEN RESCHEDULED
*
*    THIS CODE REPLACES PREVIOUSLY OVER-WRITTEN JSB WORD
*
RPLAC LDA JS$AB
      SZA,RSS
      JMP SEQER      SEQUENCE ERROR
      JSB $LIBR      GO PRIVILEGED
      NOP            (NOT RE-ENTRANT SUBROUTINE)
      LDA JS$AB      GET SAVED 'JSB $SYMG' WORD
      STA $ABRT+23    AND RESTORE IT
      JSB $LIBX      TURN ON INTERRUPT
      DEF *+1
      DEF *+1
      CLA            CLEAR A REG
      STA JS$AB       AND CLEAR TEMPORARY SAVE WORD
      JSB EXEC       CLOSE UP SHOP
      DEF RTN
      DEF SIX        TERMINATE
      DEF ZERO        SELF AND
      DEF ONE          SAVE RESOURCES
RTN   NOP
*                                 *
*    SEQUENCE ERROR PROCESSOR     *
*                                 *
SEQER JSB EXEC       WRITE ERROR MSG ON CONSOLE
      DEF *+1+4
      DEF D2         WRITE FUNCTION
      DEF CONWD      CONTROL WORD
      DEF ERARY
      DEF ERLNG
      JSB EXEC       NOW CLOSE UP SHOP
      DEF *+1+4
      DEF SIX
      DEF ZERO
      DEF ONE
      DEF ZERO
SIX   OCT 6
D2    OCT 2
ONE   OCT 1
JS$AB NOP
BUFR  BSS 5
CONWD OCT 1
ZERO  OCT 0
ERARY ASC 11,/ABMSG: SEQUENCE ERROR
ERLNG DEC -22
      END ABMSG
```

EDITOR'S NOTE:

Darrell is now working on an RTE-IV version of this program which should prove to be very useful for GRAPHICS/1000.

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.

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 HP 1000 Computer Systems 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 on the next page, 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 four 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 (22980C).

- Operate your system as part of a distributed systems (DS/1000) network. . .
  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. . .
  You should take the *HP 1000 Driver Writing* course (22990A) to learn how to write device drivers for your own peripherals.

- Customize your computer for your application using the Microprogramming feature of the HP 1000. . .
  You should take the *HP 1000 E/F Series Computers Microprogramming* course (22983B).

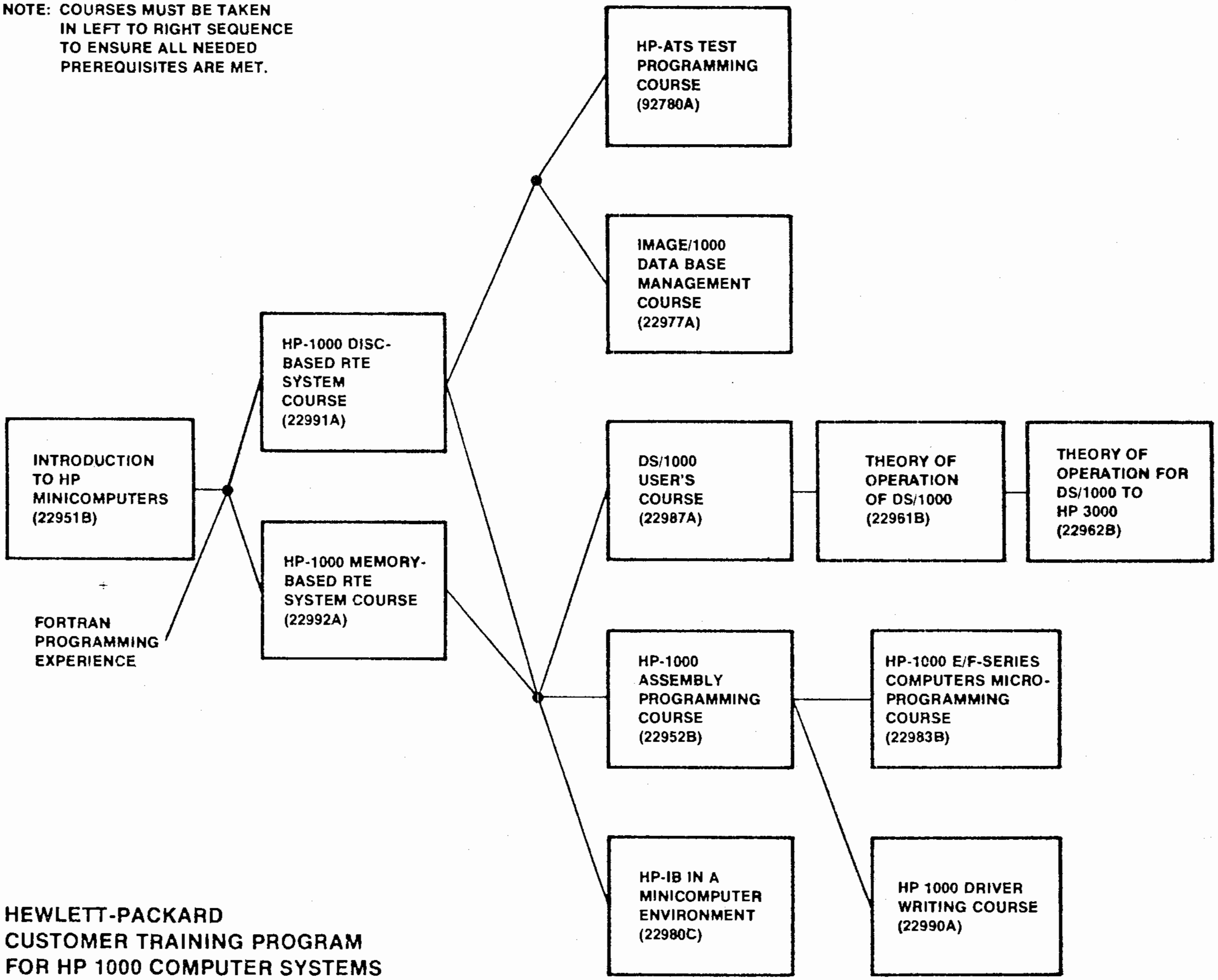
- Write test programs for your HP-ATS system. . .
  You should take the *HP-ATS Test Programming* course (92780A).

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.


INTRODUCTION TO HP MINICOMPUTERS (22951B)
+
FORTRAN PROGRAMMING EXPERIENCE
HP-1000 DISC-BASED RTE SYSTEM COURSE (22991A)
HP-1000 MEMORY-BASED RTE SYSTEM COURSE (22992A)
HP-ATS TEST PROGRAMMING COURSE (92780A)
IMAGE/1000 DATA BASE MANAGEMENT COURSE (22977A)
DS/1000 USER'S COURSE (22987A)
THEORY OF OPERATION OF DS/1000 (22961B)
THEORY OF OPERATION FOR DS/1000 TO HP 3000 (22962B)
HP-1000 ASSEMBLY PROGRAMMING COURSE (22952B)
HP-1000 E/F-SERIES COMPUTERS MICRO-PROGRAMMING COURSE (22983B)
HP-IB IN A MINICOMPUTER ENVIRONMENT (22980C)
HP 1000 DRIVER WRITING COURSE (22990A)

HEWLETT-PACKARD
CUSTOMER TRAINING PROGRAM
FOR HP 1000 COMPUTER SYSTEMS

BULLETINS

NEW COURSES

Since the last issue of the Communicator/1000, two new courses have been added to the schedules that appear on the following pages. First is a brand new, two-week long **Memory-based RTE System Course (22992A)**, which covers the operation and programming of the RTE-M operating system. This course replaces the old one-week long RTE-M Course (22985A), which will shortly be obsoleted.

The second new course is the **Advanced RTE Workshop**, which is currently offered only at the Cupertino Customer Training Center. This course is taught by some of Hewlett-Packard's most experienced Systems Engineers and presents an in-depth discussion of the internal operation of the RTE-IV operating system.

More detailed information on both of these courses is given below.

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, relocating 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.

ADVANCED RTE WORKSHOP

Workshop Scope: This workshop will introduce the system analyst/programmer to the internal design and operation of the Real Time Executive operating system.

Who Should Attend: This 5-day workshop is designed for systems analyst/programmers who need to tune their systems for maximum performance. Since a considerable amount of material will be covered, attendees should be prepared for a very full week.

Prerequisites: The attendees must have at least 6 months experience in using the Real Time Executive and must have experience in using HP Assembly language.

Workshop Length: 5 days, 8: a.m. to 6:00 p.m.

Registration and Fee: Request for enrollment should be made through the Northern Neely training registrar (408) 996-9800, through your local HP Sales Office, or through your Sales Representative and should be accompanied by a purchase order or check for $800 made payable to Hewlett-Packard.

WORKSHOP SCHEDULE

OUTLINE FOR ADVANCED RTE IV WORKSHOP

	MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY
8:00	Introduction	Review Labs	Resource Numbers	Review Hw #3	Review Hw #2, 4
			LU Locks		
		Operator Requests		Re-Entrant Processing	PowerII
		—Trace "ON,XYZ" From		—LIBR/LIBX	Systems Library
9:00	Hardware Overview	Keyboard to $XEQ		REIO	Utilities
				SAM	
				—Users of SAM	
			COFFEE BREAK		
10:00			Review Hw #1	—SAM Management	
	RTE Overview	Program Dispatching	Program States		Performance
		Partition Assignment	—State Diagram		Measurement
	RTE Modules		--$LIST		
11:00				I/O Drivers	
				—Initialiszation	Lab Seminar
				—Continuation	
				—Completion	
				—Priviledged	
12:00					
			LUNCH		
1:00	DMS	I/O Processing	TBG Time Tick		EMA
	—Phy./Log. Memory	Overview	—Trace From Interr.		—EMA in Fortran
	—RTE Maps		to $XEQ		—EMA in Assembler
2:00	Boot Process	Exec Calls			—EMAST, MMAP
	—Trace From Front	--Trace Exec2			—EMAP, EMIO
	Panel Thru $STRT	Call from MP Thru I/O			
			COFFEE BREAK		
		Completion	Class I/O		
3:00			MTM		
	CMM4/DBUGR	Parity Errors	—Trace From Keyboard	Lab	Exam
			Thru RPN		
4:00	Lab	Lab	Lab		
5:00					
6:00					

TRAINING SCHEDULE

The current schedule for customer training courses on HP 1000 computer systems products is given in this section. Included are courses offered in the U.S., Europe, and in International areas 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 or International 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	LOS ANGELES Customer Training Center	WASHINGTON D. C. Customer Training Center	CHICAGO Customer Training Center	ATLANTA Customer Training Center	DALLAS Customer Training Center	NEW YORK Customer Training Center
	Length	Price							
22951B	Intro to HP mini's		Apr 2	Jan 8 Mar 12 Apr 30	Jan 8 Feb 12 Mar 19 Apr 16	Jan 29 Apr 2		May 21	Mar 5 May 7
	4 days	400							
22991A*	HP 1000 DISC RTE		Jan 8 Jan 22 Feb 5 Feb 26 Mar. 12 Mar 26 Apr 16 Apr 30 May 14 Jun 4	Jan 15 Feb 12 Mar 19 Apr 16 Mar 7	Jan 15 Jan 29 Feb 26 Mar 12 Mar 26 Apr 16 Apr 30 May 14	Neb 12 Apr 16	Jan 29	Mar 19	Mar 12 Apr 16 May 14
	10 days	1000							
	(Course includes RTE-IV operating system, batch spool monitor and file manager.)								
22992A*	HP 1000 Memory RTE		Feb 26						
	10 days	1000							
22977A*	IMAGE		Jan 22 Apr 23	Feb 26 May 21	Feb 5 Mar 12 May 21	Feb 26 Apr 30			Apr 2 Jun 4
	5 days	500							
22952B*	1000 ASMB		Feb 5 Apr 30	Jan 29 Apr 2 Jun 4	Jan 29 Mar 5 Apr 23	Mar 5			Mar 26 Apr 30
	5 days	500							
22987A*	DS/1000 User's Course		Mar 26		Jan 8 Apr 30				
	5 days	500							
22961B*	DS/1000 Theory of Op.				Jan 15 May 7				
	4 days	400							
22962B*	DS/1000 to HP 3000 Theory of Op.				Jan 19 May 11				
	1 day	100							
22990A*	RTE-Driver Writing		Feb 12 May 21		Feb 19 May 28				
	3 days	300							

*These courses carry prerequisites — refer to the training program diagram and discussion on the previous pages for more information.

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

Course Number	Title		CUPERTINO Customer Training Center	LOS ANGELES Customer Training Center	WASHINGTON D. C. Customer Training Center	CHICAGO Customer Training Center	ATLANTA Customer Training Center	DALLAS Customer Training Center	NEW YORK Customer Training Center
	Length	Price							
22980C*	HP-IB Minicomputer Environment		Jan 15 Mar 19						
	4 days	400							
22983B*	HP 1000 E/F Microprogramming		Jan 29 May 7						
	5 days	500							
	Advanced RTE Workshop		Jan 8 Mar 12 May 14						
	5 days	800							

*These courses carry prerequisites — refer to the training program diagram and discussion on the previous pages for more information.

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

Course Number	Title		Data Systems Division (CUPERTINO)	Data Terminals Division (CUPERTINO)	Customer Service Division (CUPERTINO)	Boise Division (BOISE)	
	Length	Price					
92780A*	HP-ATS Automatic Test System		Feb 26				
	5 days	1000					
13294A	Dev. Terminal			Jan 8 Feb 26			
	5 days	500					
22940A	2100 Maint.				Jan 22 Feb 26		
	10 days	1000					
22941A	21MX/XE Maint.				Jan 8 Jan 15 Jan 29 Feb 5		
	5 days	500					
22942A	7900 Maint.				Jan 15 Feb 12		
	5 days	500					
22945A	7905 Maint.				Jan 8 Feb 5 Feb 12		
	5 days	500					
91302A	2645 Maint.						
	3 days	300					
22943A	7970B Maint.						
	5 days	600					
22944A	7970E Maint.						
	5 days	600					

*These courses carry prerequisites — refer to the training program diagram and discussion on the previous pages for more information.

EUROPEAN TRAINING CENTER SCHEDULES AND LOCATIONS

Course Number	Title	Length		Boblingen	Amsterdam	Madrid	Winnersh	Milan (M) Rome (R)	Stockholm	Helsinki	Orsay	Vienna	Brussels
22951B*	Intro to HP minis	4 days	400						Jan 22 Apr 02 Oct 08			Sep 3	
22965B	RTE-II/III (Course includes RTE-II/III operating system, batch spool monitor and file manager)	10 days									Jan 8 Feb 12 Mar 26 May 7 Jun 11	Jan 15	
22991A*	HP 1000 DISC RTE (Course includes RTE-IV operating system, batch spool monitor and file manager.)	10 days	1000						Jan 29 Mar 5 Apr 23 Sep 10 Oct 15 Nov 19	Jan 15 Mar 5		Mar 19 Sep 10	Jan 8 Mar 12 May 28 Oct 1
22985A	RTE-M	5 days									Mar 5		
22977A*	IMAGE	5 days								Feb 5	Mar 12 Jun 25 Sep 24	Jan 29 Apr 02	
22952B*	1000 ASMB	5 days							Feb 26 Sep 03 Nov 12	Apr 02	Mar 5 May 08	Sep 24	
22987A*	DS/1000 User's Course	5 days								Feb 19	Feb 5		
22961B*	DS/1000 Theory of Op	4 days											
22962B*	DS/1000 to HP 3000 Theory of Op	1 day											
22990A*	RTE Driver Writing	3 days											
22980C*	HP-IB Minicomputer Environment	4 days									Apr 17		
22983B*	HP 1000 E/F Micro-programming	5 days											

*These courses carry prerequisites — refer to the training program diagram and discussion on the previous pages for more information

EUROPEAN TRAINING CENTER SCHEDULES AND LOCATIONS (Continued)

Course Number	Title	Length	Boblingen	Amsterdam	Madrid	Winnersh	Milan (M) Rome (R)	Stockholm	Grenoble	Orsay	Vienna	Brussels
92780A*	HP-ATS Automatic Test System	5 days										
13294A	Dev. Terminal	5 days										
22940A	2100 Maint.	10 days										
22941A	21MX/XE Maint.	5 days							Feb 26			
22942A	7900 Maint.	5 days							Mar 5			
22945A	7905/06 Maint.	5 days							Feb 19 Apr 02			
22984A	7920 Maint.	5 days							Apr 09			
91302A	2645 Maint.	3 days							Jan 22			
22943A	7970B/E Maint.	5 days							Jan 15 Mar 26			
40270A	Intro to HP Computers	5 days							Jan 22 Apr 09 Jul 02			
22965B-H01	FORTRAN IV	5 days										

*These courses carry prerequisites. Refer to the training program diagram and discussion on the previous pages for more information.

INTERCONTINENTAL TRAINING CENTER SCHEDULES AND LOCATIONS

Course Number	Title	Montreal	Toronto	Australia Blackburn, VIC (B) Pymble, NSW (P)	Japan	
	Length					
22951B	Intro to HP mini's	Feb 19**	Jan 08			
	4 days					
22991A*	HP 1000 DISC RTE	Mar 19**	Feb 19	Feb 26 (B) May 21 (P) Jul 09 (B) Sep 17 (P) Oct 22 (B)		
	10 days					
	(Course includes RTE-IV operating system, batch spool monitor and file manager.)					
22992A*	HP 1000 Memory RTE					
	10 days					
22977A*	IMAGE			Apr 30 (B) Jul 09 (P) Aug 20 (B) Nov 12 (P) Dec 03 (B)		
	5 days					
22952B*	1000 ASMB			Jun 11 (P) Jul 30 (B) Oct 08 (P) Nov 12 (B)		
	5 days					
22987A*	DS/1000 User's Course			May 28 (B) Oct 22 (P)		
	5 days					
22961B*	DS/1000 Theory of Op.					
	4 days					
22962B*	DS/1000 to HP 3000 Theory of Op					
	1 day					
22990A*	RTE-Driver Writing			Jun 18 (P) Aug 06 (B) Oct 01 (P) Nov 19 (B)		
	3 days					

*These courses carry prerequisites — refer to the training program diagram and discussion on the previous pages for more information.

**These courses are taught in French.

U.S. TRAINING CENTER ADDRESSES

Atlanta

CUSTOMER TRAINING CENTER
450 Interstate North Parkway, NW
Atlanta, Georgia 30339
(404) 955-1500

Boise

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

Cupertino

CUSTOMER TRAINING CENTER
19320 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

CUSTOMER SERVICE DIVISION
19310 Pruneridge Avenue
Cupertino, CA 95014
(408) 996-9383

Dallas

CUSTOMER TRAINING CENTER
201 E. Arapaho Road
Richardson, Texas
(214) 231-6101

Los Angeles

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

Washington, D.C.

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

New York

CUSTOMER TRAINING CENTER
120 Century Road
Paramus, N.J. 07652
(201) 265-5000

EUROPEAN TRAINING CENTER ADDRESSES

Amsterdam

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

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

Grenoble

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

Helsinki

Nahkahousuntie 5
00211 Helsinki 21
Tel: 90-692 30 31

Madrid

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

Milan

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

Orsay

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

Stockholm

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

Vienna

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

Winnersh

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

INTERCONTINENTAL TRAINING CENTER ADDRESSES

Blackburn, Australia

CUSTOMER TRAINING CENTER
31-41 Joseph Street
Blackburn, Victoria, Australia

Pymble, Austria

CUSTOMER TRAINING CENTER
31 Bridge Street
Pymble, New SouthWales, Australia

Montreal

CUSTOMER TRAINING CENTER
275 Hymus Boulevard
Pointe Claire, Quebec, Canada H9R1G7
(514) 697-4232

Toronto

CUSTOMER TRAINING CENTER
6877 Goreway Drive
Mississauga, Ontario, Canada, L4V 1M8
(416) 678-9430

HEWLETT-PACKARD
COMPUTER SYSTEMS COMMUNICATOR ORDER FORM

Please Print:

Name ______________________ Date ______________

Company ______________________

Street ______________________

City ______________ State ______________ Zip Code __________

Country ______________________

□ HP Employee Account Number __________ Location Code __________

□ DIRECT SUBSCRIPTION

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

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

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

TOTAL ORDER DOLLAR AMOUNT __________

□ SERVICE CONTRACT CUSTOMERS

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

Number of additional copies __________

FOR HP USE ONLY

CONTRACT KEY

5951-6111 Number of additional copies __________

5951-6112 Number of additional copies __________

5951-6113 Number of additional copies __________

Approved ______________________

HEWLETT-PACKARD
COMMUNICATOR SUBSCRIPTION AND ORDER INFORMATION

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

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

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

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

SAMPLE

☒ DIRECT SUBSCRIPTION

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

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

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

SAMPLE

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

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

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

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

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

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

HEWLETT hp PACKARD

CONTRIBUTED SOFTWARE
Direct Mail Order Form

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

Please Print:

Name ______________________________ Title ______________________

Company ____

Street ____

City __________________________ State _______________ Zip Code ________

Country ____

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

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

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

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

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

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

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

ORDERING INFORMATION

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

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

For example:

22113A-K01

The different options are.

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

NOTE

Specify 800 BPI or 1600 BPI Magnetic tape.

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

Not all options are available for all programs.

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

For example:

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

Only those options listed in each abstract are available.

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

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

HEWLETT-PACKARD
LOCUS CONTRIBUTED SOFTWARE CATALOG
DIRECT MAIL ORDER FORM

Please Print:

Name ______________________________ Title ______________

Company ____

Street ____

City ______________________________ State ________ Zip Code ______

Country ____

□ HP Employee Account Number ______ Location Code ______

Part Number	Description	Qty.	List Price Each	Extended Total
22000-90099	Locus Contributed Software Catalog		$15.00	
		Your State & Local Sales Taxes*		
		Handling Charge		1.50
		TOTAL		

*If no sales tax is added, your state exemption number must be provided: #________

If not, your order may have to be returned.

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.

NOT TO BE USED FOR ORDERING COMMUNICATOR SUBSCRIPTIONS

HEWLETT hp PACKARD

CORPORATE PARTS CENTER

Direct Mail
Parts and Supplies Order Form

SHIP TO.

NAME ____________________

COMPANY ____________________ CUSTOMER REFERENCE # ____________

STREET ____________________ TAXABLE ? ____________

CITY ____________ STATE ____________ ZIP CODE ____________

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

Special Instructions

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

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

Check or Money Order, made payable to Hewlett-Packard Company, must accompany order.

When completed, please mail this form with payment to:

HEWLETT-PACKARD COMPANY
Mail Order Department
P.O. Drawer #20
Mountain View, CA 94043

Phone: (415) 968-9200

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.

Printed in U.S.A. 11/78

Part No. 5951-6111