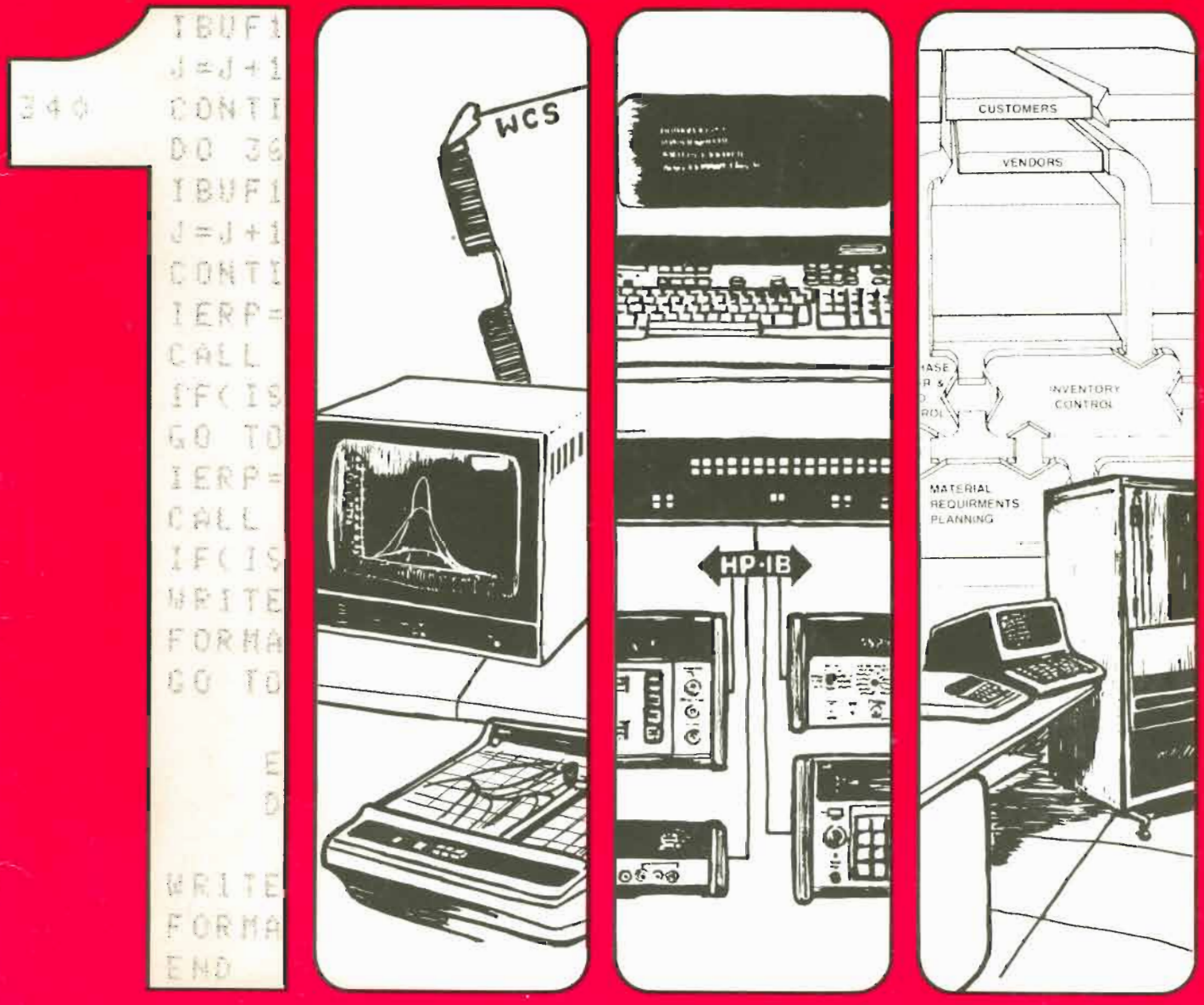
HEWLETT PACKARD

Hewlett-Packard Computer Systems

COMMUNICATOR



1979

Volume III Issue 2

HP Computer Museum
www.hpmuseum.net

For research and education purposes only.

HEWLETT-PACKARD
COMPUTER SYSTEMS

Volume III
Issue 2

COMMUNICATOR/1000

Feature Articles

Departments

EDITOR'S DESK

ABOUT THIS ISSUE

Volume III, Issue 2 of the Communicator/1000 showcases four very fine Feature Articles, two each in the areas of OPERATING SYSTEMS and COMPUTATION.

For this issue's OPERATING SYSTEMS category, Frank Slootweg of Hewlett-Packard Amstelveen, The Netherlands, has written an article which presents his solution to the problem of performing blockmode reads on HP 264X series CRT terminals. Frank has included the source code for two programs, one of which handles the problem on 2645A terminals (and similar). His other program works on 2640A/B terminals. Also in the OPERATING SYSTEMS section is a fine article by Harvey Bernard of the HP Rockville (Maryland) Eastern Technical Center. Adopting an informal and light manner, Harvey discusses the method of accommodating the testing of user-written drivers in RTE-IV.

The COMPUTATION section features two equally excellent articles. Larry W.Smith of the HP Fullerton (California) Sales and Service Office has submitted an article describing his method of fitting a curve to a series of data points. While the article succeeds in satiating the appetites of students of theory (Larry gives a conceptual demonstration of his method), pragmatists will also be happy to note that the source code of the subroutine appears at the end of the discussion. The second article of this section was contributed by Joel Dubois of HP Grenoble (France). Joel enlightens all of us to the "secrets" of the microprogramming base set, and astutely pinpoints a possible stumbling block for microprogrammers. His discussion is not only interesting, but timely as well.

Again, as in past issues, the field of Feature Articles for this issue was equally excellent all-around. And once again, it remained a difficult task to select the best articles to receive prizes of HP-32E hand-held calculators. A panel of three judges, all of whom are members of the HP Data Systems Technical Marketing Department, have selected the following two articles as this issue's winners:

Best Feature Article by an HP Employee in the Field	**A METHOD FOR SMOOTH CURVE FITTING** Larry W. Smith
Best Feature Article by an HP Division Employee not in Data Systems Technical Marketing	**MICROPROGRAMMING BASE SET SECRET** Joel Dubois

Unfortunately, we had no articles for this issue contributed by customers of Hewlett-Packard, but we encourage our customers to do so. There is a calculator to be won in that category!

Finally, an error in Volume II, Issue 6 stands to be corrected. An inaccurate flowchart appeared in Larry W.Smith's article "Shared EMA for RTE-IV", Figure 3, page 40. The corrected flowchart is herein printed. We regret the illustration error.

The Editor

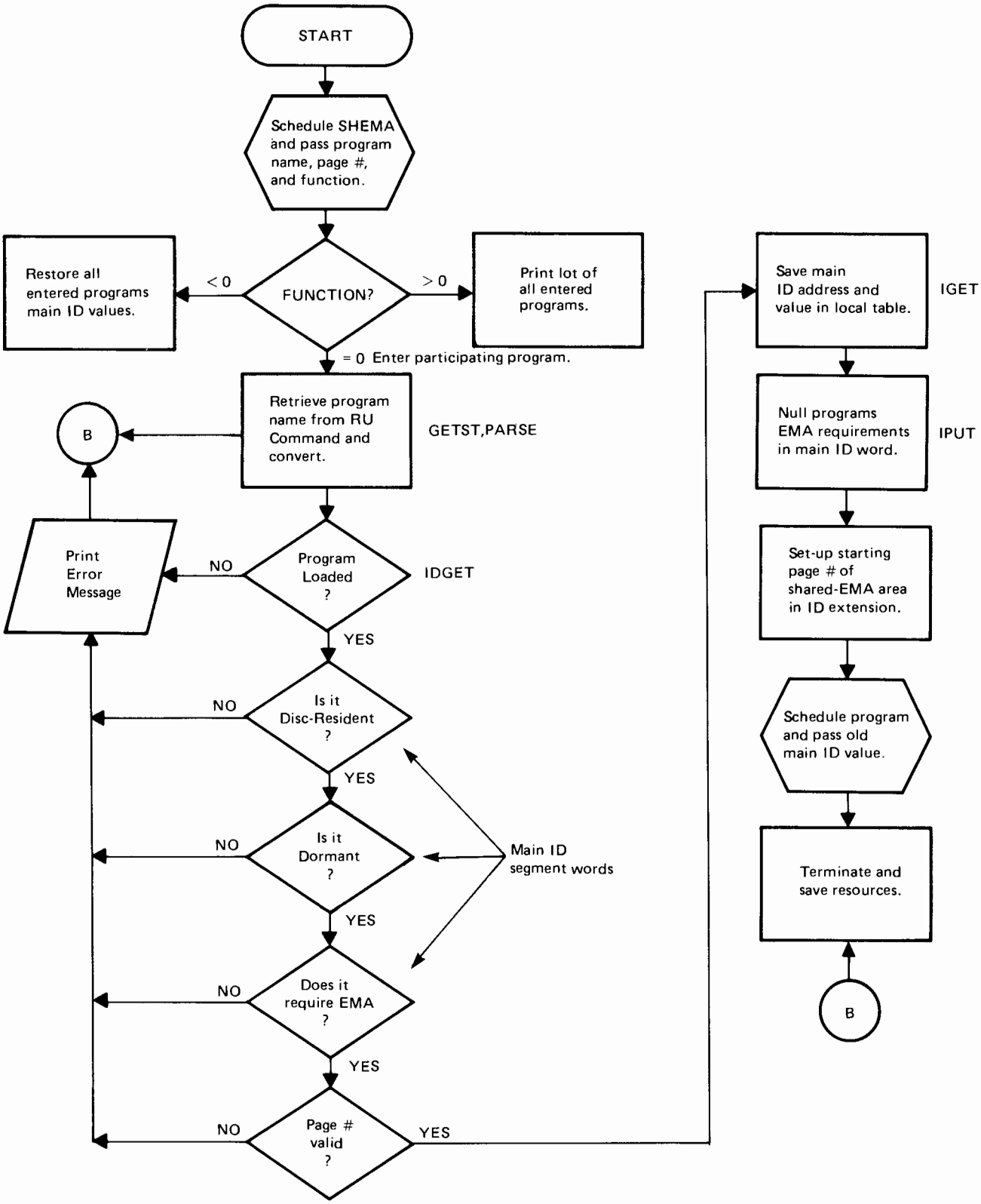

START
Schedule SHEMA and pass program name, page #, and function.
FUNCTION?
< 0
Restore all entered programs main ID values.
> 0
Print lot of all entered programs.
= 0 Enter participating program.
Retrieve program name from RU Command and convert.
GETST,PARSE
B
Program Loaded ?
NO
IDGET
Print Error Message
YES
Is it Disc-Resident ?
NO
YES
Is it Dormant ?
NO
YES
Main ID segment words
Does it require EMA ?
NO
YES
Page # valid ?
NO
YES
Save main ID address and value in local table.
IGET
Null programs EMA requirements in main ID word.
IPUT
Set-up starting page # of shared-EMA area in ID extension.
Schedule program and pass old main ID value.
Terminate and save resources.
B

Figure 3

EDITOR'S DESK

BECOME A PUBLISHED AUTHOR IN THE COMMUNICATOR/1000 . . .

The COMMUNICATOR is a technical publication designed for HP 1000 computer users. Through technical articles, the direct answering of customers' technical questions, cataloging of contributed user programs, and publication of new product announcements and product training schedules, the COMMUNICATOR strives to help each reader utililize their HP 1000's more effectively.

The Feature Articles are clearly the most important part of the COMMUNICATOR. Feature Articles are intended to promote a significant cross-fertilization of ideas, to provide in-depth technical descriptions of application programs that could be useful to a wide range of users, and to increase user understanding of the most sophisticated capabilities designed into HP software. You might think of the COMMUNICATOR as a publication which can extend your awareness of HP 1000's to include that of thousands of users worldwide as well as that of many HP engineers in Data Systems factories at Cupertino, California and Grenoble, France.

To accomplish these goals, editors of the COMMUNICATOR actively seek technical articles from HP 1000 customers, HP Systems Engineers in the Field, and Marketing and R&D Engineers in the factories. Technical articles from customers are most highly valued because it is customers who are closest to real-world applications.

WIN AN HP-32E CALCULATOR!

Authoring a published article provides a uniquely satisfying and visible feeling of accomplishment. To provide a more tangible benefit, however, HP gives away three free HP-32E hand-held calculators to Feature Article authors in each COMMUNICATOR/1000 issue! Authors are divided into three categories. A calculator is awarded to the author of the best Feature Article in each of the author categories. The three author categories are:

1. HP 1000 Customers;

2. HP employees not in Data Systems Division (e.g., HP Systems Engineers, users in other HP Divisions, etc.);

3. HP Data Systems Division employees not in the Technical Marketing Dept. (from which the COMMUNICATOR Editor is chosen).

Each author category is judged separately. A calculator prize will be awarded even if there is only one entry in an author category.

Feature Articles are judged on the following bases: (1) quality of technical content; (2) level of interest to a wide spectrum of COMMUNICATOR/1000 readers; (3) thoroughness with which subject is covered; and, (4) clarity of presentation.

What is a Feature Article? A Feature Article meets the following criteria:

1. Its topic is of general technical interest to COMMUNICATOR/1000 readers;

2. The topic falls into one of the following categories —

   OPERATING SYSTEMS
   DATA COMMUNICATIONS
   INSTRUMENTATION
   COMPUTATION
   OPERATIONS MANAGEMENT

3. The article covers at least two pages of the COMMUNICATOR/1000, exclusive of listings and illustrations (i.e., at least 1650 words).

There is a little fine print with regard to eligibility for receiving a calculator; it follows. No individual author will be awarded more than one calculator in a calendar year. In the case of multiple authors, the calculator will be awarded to the first listed author of the winning article. 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. Employees of Technical Marketing at HP's Data Systems Division factory in Cupertino are not eligible to win a calculator.

All winners of calculators will be announced in the issue of the COMMUNICATOR/1000 in which their articles appear. Again, all Feature Articles are judged by an impartial panel of three DSD Technical Marketing Engineers.

A SPECIAL DEAL IN THE OEM CORNER

When an HP 1000 OEM writes a Feature Article that is not only technically detailed and insightful but also application-oriented as opposed to theoretical, then that OEM may ask that the article be included in THE OEM CORNER. A Feature Article included in THE OEM CORNER may contain up to 150 words of pure product description as well as a picture or illustration of the OEM'S product or its unique contribution. HP's objective is twofold: (1) to promote awareness of the capabilities HP 1000 OEMs' products among all HP 1000 users; and, (2) to publish an article of technical interest and depth.

IF YOU'RE PRESSED FOR TIME . . .

If you are short of time, but still have that urge to express yourself technically, don't forget the COMMUNICATOR/1000 BIT BUCKET. It's the perfect place for a short description of a routine you've written or an insight you've had.

THE MECHANICS OF SUBMITTING AN ARTICLE

If at all possible please submit an RTE File containing the text of your article recorded on a Minicartridge (preferrably) or on a paper tape along with the line printer or typed copy of your article. This will help all of us to be more efficient. The Minicartridge will be returned to you promptly. Please include your address and phone number along with your article.

All articles are subject to editorship and minor revisions. The author will be contacted if there is any question of changing the information content. Articles requiring a major revision will be returned to the author with an explanatory note and suggestions for change. We hope not to return any articles at all; if we do, we would like to work closely with the author to improve the article. HP does, however, reserve the right to reject articles that are not technical or that are not of general interest to COMMUNICATOR/1000 readers.

Please submit your COMMUNICATOR/1000 article to the following address:

Editor, COMMUNICATOR/1000
Data Systems Division
Hewlett-Packard Company
11000 Wolfe Road
Cupertino, California 95014
USA

The Editor looks forward to an exciting year of articles in the COMMUNICATOR/1000.

With best regards,

The Editor

LOCUS CHANGES AND ADDITIONS

This article updates the Data Systems LOCUS Program catalog (22000-90099). The following changes have been made in existing LOCUS programs.

The "DBLST-RTE2/3 Image Data Base Information Lister" program has been revised. It is now available on minicartridge:

22682-13380	Minicartridge	$35.00

The "F8-Fairchild F-8 Assembler for the HP2100-21MX" program has been revised. It is now available on minicartridge:

22682-13381	Minicartridge	$35.00

The "MEMAL-RTE2 Memory Allocation Diagram" program has been revised. It is now available on minicartridge:

22682-13383	Minicartridge	$40.00

The "ENTPT--Alphabetic List of Entry Points in RTE-2" program has been revised. It is now available on minicartridge:

22683-13302	Minicartridge	$40.00

The "PASCAL-S Compiler/Interpreter" program has been revised. It is now available on magnetic tape:

22683-10905	800 bpi Magnetic Tape	$70.00
22683-11905	1600 bpi Magnetic Tape	$70.00

The new contributed programs listed below are now available in LOCUS. Contact your local HP sales office to order Contributed Library programs (see LOCUS ORDERING INFORMATION at the end of this article).

22683-XXX24 RDWRT

"RDWRT" is a complete program which demonstrates the mass storage and file handling capabilities of the HP 1000 when used with the 3582A Spectrum Analyzer. Supplied with the package is a user program which prompts the operator to respond to the commands shown below:

SD — SAVE THE DISPLAY BUFFER IN A DISC FILE
ST — SAVE THE TIME BUFFER IN A DISC FILE
RD — RESTORE THE DISPLAY BUFFER TO THE 3582A FROM A DISC FILE
RT — RESTORE THE TIME BUFFER TO THE 3582A FROM A DISC FILE

The program allows the raw time waveform or the transformed frequency domain waveform from the 3582A to be stored in an HP 1000 disc file. At some time later these files may be restored to the instrument for comparison or they can be re-analyzed. The program allows:

1. storage and retrieval of frequency spectra.
2. storage and retrieval of the original time waveforms for
   a. later analysis,

b. later analysis of the experiment using different transfer functions (Flat Top, Hanning, or Uniform) or,

c. modification or preprocessing of the time waveform (i.e., passing it through a simulated filter) before spectrum analysis.

22683-13324 Minicartridge $40.00

22683-XXX25 SHEMA

This program allows any number of EMA programs to share as much physical memory that is allowed in any given configuration. This capability uses standard HP supplied software and requires only one subroutine call in each program before EMA sharing can be done. The scheme developed for this capability allows any program type to participate and does not make use of any other area of memory.

22683-13325 Minicartridge $40.00

22683-XXX26 IMBUI

Several functions and parameters are needed in fast neutron time-of-flight spectroscopy; they are referred to as scatterer integrals. The following quantities are computed by IMBUI for a disc scatterer:

a) The neutron time-of-flight spectrum Γ (t_d),
b) The distribution of scattering angles, B (θ_c),
c) The distribution of primary neutron energies, F (E_p),
d) The scattering probability, P, and
e) The integral Φ.

22683-13326 Minicartridge $40.00

22683-XXX27 SYSTK

Enables message communication between two terminals in a MTM environment. Up to ten 80-character message lines may be sent from one terminal to another. Control may be maintained at one terminal or passed along with or without a message to the second terminal. Functions are determined through four commands preceeded by a program prompt (?):

1) TEXT — Read the current control TTY's message (Ten lines or a ?? at the start of a line returns the prompt).

2) SEND — Output the current message buffer and retain control.

3) GIVE — Output the current message buffer (if any) and pass control to the next user.

4) END — Terminate SYSTK.

22683-13327 Minicartridge $40.00

22683-XXX28 SMESS

Enables system console (or initiator) to output a message to all TTY's in a MTM environment. Up to ten 80-character message lines may be sent via the TEXT/SEND command. The first message sent also includes the Gregorian date and current time. SMESS functions are determined through three commands preceeded by a program prompt (?):

1) TEXT — Read up to ten lines of message from the system console (or initiator). Ten lines or a ?? at the start of a line returns the prompt.

2) SEND — Output the message to all online terminals.

3) END — Terminate SMESS.

22683-13328 Minicartridge $50.00

22683-XXX29 MESIO--Interactive I/O Subroutines

Three subroutines which allow the user to perform interactive and/or batch I/O along with alpha to numeric and numeric to alpha conversions.

MESIO — Outputs messages and accepts single or multiple entries according to several different structures. Direct writes and reads are also supported for batch mode.

ATON — Interrogates any alpha string and returns either a real or integer value.

NTOA — Formats either real or integer numbers into an alpha string.

22683-13329 Minicartridge $50.00

22683-XXX30 CMPCT--Absolute Binary Compactor

Pack absolute binary code in records of up to 128 words in length. Arrange the load in ascending memory location order. Provide a means to identify undefined areas and to selectively preset these areas.

The binary input file is examined and a table having one bit for each word found is constructed. The bounds and number of records found are output. If requested, this table is searched for undefined strings and a list of the undefined strings is output on the selected logical unit.

The format of the listing is as follows:

String number, size, decimal and binary begin and end. Following completion of the preset, the input file is rescanned, retrieving N pages of data per scan. The data is then formed into binary absolute records using the previously constructed table as a guide. The value of N in this version is set to four (4096) but the user can change N if more or less memory is available.

22683-13330 Minicartridge $50.00

22683-XXX31 UNBLK--DEC RT-11 to HP RTE Source Converter

This program converts Digital Equipment Corporation (DEC) RT-11 source files on magnetic or paper tape into HP RTE disc files. The original source can be FORTRAN, RT-11 assembler, text, data, or results. The program converts carriage returns, line feeds, tabs, nulls, and apostrophes into acceptable RTE syntax. However, there are other RT-11 peculiars that the program cannot resolve into acceptable RTE format. Therefore, the user must further modify the resulting disc file as required. However, much time is saved by automatically converting most of the source first with this program.

This conversion also includes suggestion of an appropriate RTE disc file name based on the RT-11 file name and extension (type). For FORTRAN files, the control statements FTN4,L,C and END$ are placed at the beginning and at the end of the file, respectively.

22683-13331	Minicartridge	$40.00

22683-XXX32 JKEYS--Define User Softkeys

This program allows the HP 264X terminal user to interactively define the contents of all softkeys and to store the results in a file. The file can then be dumped to the terminal with a two line leader describing each softkey. The program is completely self-explanatory and performs all necessary error checking.

22683-13332	Minicartridge	$40.00

22683-XXX33 LDISC--FMP Corrupt File Analyzer

This program allows the user to analyze and verify the integrity of all files on a cartridge, for files that are not of Types 0, 1, 2, or 6. The program opens each file and reads all records and reports any errors to a list device. A resulting statistical printout occurs showing such items as total storage, number of files, average record length, and others. The program will also work for files contained on LU 2 and LU 3.

22683-13333	Minicartridge	$40.00

22683-XXX34 DIRC--Edit FMP Disc Assigned Space

The DIRC program is an HP/1000 interactive RTE Fortran-IV program that is used to edit FMP assigned disc space for 7900 and 7905 disc subsystems. The program permits disc file editing by sector by addressing the file track and sector addresses. Disc file entries in the directory can be edited within specified restrictions such as not permitting file address modifications.

22683-13334	Minicartridge	$50.00

22683-XXX35 JFORM--CRT Forms Creator

This program allows the HP 264X terminal user to interactively create forms and store the result in an FMP file. The program loads soft keys with various enhancement options that are used to create the desired form. The program is self-explanatory and performs extensive error checking.

22683-13335	Minicartridge	$40.00

22683-XXX36 RELIA--Inverse Assembler

This program will perform an inverse assembly of a relocatable module which can exist in an FMP file or the disc-resident library. The relocatable code can be generated by an assembler or compiler and can be input via an FMP Type 0 file. The program uses approximately 9.5K of memory and 3.5K of table size for evaluating 700 unique symbols. The base page requirements are approximately 1000 octal. The program can be scheduled interactively or from a RELIA command file. Included is a comprehensive help processor which gives information on all commands.

22683-10936	800 bpi Magnetic Tape	$50.00
22683-11936	1600 bpi Magnetic Tape	$50.00

22683-XXX37 FRSEQ--Fortran Source Label Resequencing

This program accepts Fortran source language statements and sequentially resequences all statement labels as they occur in columns 1 through 5. The program is interactive and the resulting output is also printed on the terminal. The only syntax check performed in the area of errors is to check for an undefined label.

22683-10937	800 bpi Magnetic Tape	$50.00
22683-11937	1600 bpi Magnetic Tape	$50.00

22683-XXX38 DISC--Disc Track Configuration Printout

This program locates the physical disc track map table entry point ($TB32) on disc and then proceeds to print a compact table of all defined disc subchannels ordered by subchannel number. The program requires no interactive input, but checks for a corrupt system.

22683-13338	Minicartridge	$40.00

22683-XXX39 LOADR--RTE2 Interactive Relocating Loader

"LOADR" is a RTE-II interactive relocating loader. It reads relocatable code from any input device or FMP file and produces an absolute load module that is ready for execution. In addition to its linking functions, the LOADR's command parameter options may also be used to list all active programs in the system, purge permanent programs, and add or replace permanent programs.

22683-10939	800 bpi Magnetic Tape	$70.00
22683-11939	1600 bpi Magnetic Tape	$70.00

22683-XXX40 CAMP--Motorola M6800 Assembler for the 2100/21MX

The program CAMP runs on an HP 2100 or 21MX series computer and assembles Motorola M-6800 microprocessor source code. It is a two-pass assembler which will run in a DOS, RTE, or BCS environment. An 8K computer is required for BCS, and 16K for DOS.

The input to CAMP is a paper tape source containing symbolic language instructions. The output is a line printer (or TTY) listing of the symbol table, the code generated by the assembler, and the source code. The tape punch output (formatted output) is ready for loading. This is a modified version of a previously contributed assembler for the Intel 8080 microprocessor.

22683-13340	Minicartridge	$40.00

22683-XXX41 UCU--Users' Code Utility

This group of five utility programs aid in housekeeping of files stored in FMP disc files under RTE. A two character user code can be attached to each file (UCODE), the FMGR directory can be selectively listed (ULIST,ALIST), file names having common UCODEs can be collected (UFILE) or directory sorted in order of file size (BLIST).

UCODE is an interactive program that adds a character pair (UCODE) to a type 3, 4, or 5 file. The UCODE is stored in the directory in the place normally used for record size in type 2 and type 6 files.

ULIST is a program which can selectively list the file names in the file directory, with their associated UCODES.

ALIST is similar to ULIST except an in-core sort is performed before listing.

BLIST provides a directory list of the file names in order of size of each entry in blocks.

22683-13341	Minicartridge	$70.00

22683-XXX42 PURGE--FMGR File Purge Utility

Program PURGE is an RTE utility program which allows the user to quickly purge a group of files with similar names. If the user wishes to purge %FILE1, &FILE1, #FILE1, FILE1, he enters "-FILE1" as the qualifier. If he wishes to purge all files beginning with the characters XYZ, he can enter "XYZ---". It does not work on the system LU's (LU 2 or LU 3). Before a file is purged, a message is printed so that the operator is sure of what is about to be purged.

22683-13342	Minicartridge	$40.00

22683-XXX43 GSAVE/GRSTR--Disc LU Save/Restore

GSAVE, GSAV2, GRSTR, GRST2 are RTE utility programs which save (verify) and restore (verify) any file manager disc cartridges. GSAVE (GSAV2) allows from one to twenty disc LU's to be save to mag tape with a minimum of operator entry. The operator enters the string of disc LU's and the starting mag tape file number. The program then saves and verifies each disc LU, prints appropriate messages to the TTY, and in the event of a verify error, it will automatically proceed to the next LU (after printing a message to that effect). It also creates its own header with TBG time, date, LU number, CR number, CR label, first FMGR track, and the next available track.

GRSTR (GRST2) will selectively retrieve a GSAVE file from tape and restore it to any disc LU. It will restore a different LU, different size LU, or a different system. In the event that the destination disc LU is not large enough, GRSTR will print a message and abort.

GSAVE/GRSTR use a 6144 word buffer, while GSAV2/GRST2 use a 2048 word buffer. These programs do not work on LU 2 or LU 3.

22683-13343	Minicartridge	$50.00

22683-XXX44 DL--Alphabetized Directory Listing

Program DL is an RTE utility program which will give an alphabetized directory listing on the line printer. It works on any file manager disc LU (including LU 2 and LU 3), and will not modify the disc LU in any way. It gives only one entry per disc file (i.e. no extents are shown). All of the following information is listed:

1) File name
2) File type
3) Number of extents for this file
4) Total number of blocks in this file
5) Security code (alphanumeric)

In addition, the following information is listed in the header:

1) Cartridge label
2) LU number
3) CR number
4) Next FMGR track
5) Next sector
6) First FMGR track

22683-13344	Minicartridge	$50.00

22683-XXX45 RAT4--FTN4 Preprocessor

RAT4 is preprocessor which inputs RATFOR source code and produces FTN4 code. RATFOR is a structured language which can be easily understood by anyone familiar with FORTRAN. It is described in the following reference:

Kernighan, B. and Plauger, P., "Software Tools", Addison-Wesley, 1976.

This version of RATFOR runs on an HP/1000 under RTE-IV, and requires a 22 page partition. It has a built-in macro processor which allows user expansion of the language.

22683-10945	800 bpi Magnetic Tape	$70.00
22683-11945	1600 bpi Magnetic Tape	$70.00

LOCUS ORDERING INFORMATION

Starting May 15, 1979, the LOCUS programs are not orderable anymore by direct mail. Please direct all your orders to the nearest sales office. These orders will be forwarded to Data Systems Division for further processing.

LETTER TO THE EDITOR

To the Editor:

Here is a Fortran-IV compile and load transfer file which I have found to be quite useful. Parameters are defaulted so that for a single compile and load, one need only type

TR,*FTNCL,&FILE

For less simple cases, provision has been made for specifying the list unit, the compiler option string, command input for the loader, and the file name or LU for the relocatable output.

```
:SV,2,9,IH
:*
:* FTN4,COMPILE AND LOAD PROCEDURE FOR RTE-IV
:*
:* R.B. GILBERT
:* PRINCETON UNIVERSITY
:* VERSION OF 4/19/79
:*
:* USAGE:
:*  TR,*FTNCL,P1,P2,P3,P4,P5
:*   P1 IS THE SOURCE LU OR FILE
:*   P2 IS THE LIST LU OR FILE (DEFAULTS TO USER'S CONSOLE OR
:*       TO LU 6 IF FMGR NOT SCHEDULED FROM A CONSOLE.)
:*   P3 IS THE COMPILER OPTION STRING (OPTIONAL)
:*   P4 IS THE LOADER COMMAND INPUT FILE OR LU
:*   P5 IS THE RELOCATABLE OUTPUT FILE OR LU.  (IF THE SOURCE
:*       FILE NAME BEGINS WITH AN AMPERSAND, P5 DEFAULTS TO THE
:*       SOURCE FILE NAME WITH THE FIRST CHAACTER CHANGED
:*       TO A PERCENT SIGN.  P5 MUST BE SPECIFIED IF THE
:*       SOURCE IS READ FROM AN LU
:IF,5G,NE,,3
:* REPLACE FIRST CHAR OF SOURCE FILE NAME WITH A PERCENT SIGN.
:CA,5,1G
:CA,-19:P,-35P,AND,377B,OR,22400B
:IF,2G,NE,,5
:* LIST LU WAS NOT SPECIFIED, DEFAULT IT.
:IF,-40P,NE,1,2
:CA,2,0G
:IF,,EQ,,1
:CA,2,6
:RU,FTN4,1G,2G,-,48,3G
:* WE WILL SKIP LOADR IF THE COMPILE FAILS
:IF,2P,EQ,0,1
:IF,,EQ,,1
:IF,3P,EQ,0,2
:AN,  LOADR NOT EXECUTED DUE TO COMPILER ERROR
:IF,,EQ,,1
:RU,LOADR,4G,5G,2G
:CN,2G
:SV,9G,,IH
:SE
::
```


Sincerely,

Richard B. Gilbert
Princeton University
James Forrestal Campus
Princeton, New Jersey

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

Dear Samantha,

I am somewhat puzzled by your response to Ranjana Shah's question regarding disc space on LU 2 and LU 3 (Volume II, Number 6).

I agree that one should load and save programs on-line instead of permanently loading at generation. However, I do not see how your suggested transfer file will serve to save space on LU 2. When a Type 6 file is RP'ed, the ID part of the file is copied to a free ID segment. The program's disc address pointer (ID Word 26) is set to point to the beginning of the absolute program code in the Type 6 file. Therefore, the Type 6 file is not truly "purged" as long as an ID segment points to it (FMGR does not release that space). Furthermore, the PK command will generate an error if there are program ID segments which point to LU 2.

Sincerely,

Hans Abendschoen
HP Frankfurt

Dear Hans,

I stand corrected. It appears that there is no way to conserve space on LU 2 and LU 3 given a large number of Type 6 files executing simultaneously. Thanks for your interest and keen attention.

Sincerely,

Samantha

Inverse Relocatable Assembler For RTE and DMS

Larry W. Smith/HP Fullerton

Have you always wanted to know what the contents of a relocatable module looks like? Or, perhaps, has it always been somewhat of a mystery as to what object code the FORTRAN compiler produces? This article will present a program that will convert "any" relocatable module into assembly language source statements. The name of this program is RELIA. It was developed by Roy Murphy of Hughes Aircraft Company, Culver City, and exhibits a rather complete set of features.

The term "inverse assembly" was given several years ago to that process which performs the opposite of a language translator or assembler. A traditional inverse assembler converts relocatable (non-directly executable or non-linked) object code back into source statements of a language, usually assembly language. Using two examples, this article will illustrate how the inverse assembler RELIA operates in an RTE-IV, III, II, or M environment.

Earlier inverse assemblers were used in situations where source level conversion from one language to another was either extremely difficult and/or prohibitively expensive. In some cases, they were written to recover original source code which was lost or not easily obtainable. Although the resulting output of most compilers or assemblers usually results in a more condensed (yet functionally equivalent) form where most of the source level identification such as comments are lost, it still could be more feasible to work with the inversely assembled output rather than with the the condensed or object version. This is especially true if changes need to be made.

The program RELIA has many useful features. Some of these include the ability to handle the entire EIG instruction set, command files, module searching, and automatic record checksum. The program can be run interactively or from a previously prepared command file. A comprehensive and thorough error analysis is made for all commands as they are received and the operator 'BR' command is recognized. If a command format and or its usage is forgotten, the help command ?? can be entered to give information on a specific command or all commands. The program is completely self-contained, requiring no external subroutines.

The following is a list of all commands available to RELIA:

```
 ?? -  REQUEST INFO ON COMMAND CODE STRUCTURE
 BS -  BACKSPACE INPUT FILE TO BEGINNING OF CURRENT MODULE
 CL -  CLOSE THE CURRENT INPUT FILE
 CR -  CREATE AN OUTPUT DISC FILE
 EX -  EXIT RELIA
 FM -  FIND MODULE IN CURRENT INPUT FILE
 IA -  DO INVERSE ASSEMBLY OF CURRENT INPUT FILE
 LB -  DO INVERSE ASSEMBLY OF DISC RESIDENT LIBRARY ROUTINE
 LC -  CHANGE COMMAND LU
 LL -  CHANGE LU OF OUTPUT LIST UNIT
 LM -  LIST MODULES
 LO -  CHANGE LOG LU
 P1 -  DO PASS 1 OF INVERSE ASSEMBLY
 P2 -  DO PASS 2 OF INVERSE ASSEMBLY
 PA -  LINE PRINTER PAGE EJECT
 PU -  PURGE A DISC FILE
 RE -  OPEN AN INPUT FILE
 RW -  REWIND INPUT FILE
 SE -  SEARCH FOR ENT NAME IN RELOCATABLE FILES
 SN -  SEARCH FOR PROGRAM NAME IN RELOCATABLE FILES
 ST -  PRINT SYMBOL TABLE OF CURRENT INPUT FILE
 SX -  SEARCH FOR EXT NAME IN RELOCATABLE FILES
 TR -  TRANSFER TO/FROM AN FMP COMMAND FILE

FOR MORE INFORMATION ON ANY COMMAND, TYPE:

??,<COMMAND CODE>
```

The above printout was obtained by entering the help command '??'. The numerous functions that can be performed on a relocatable FMP file or directly on the disc-resident library are apparent. As an example, let's assume that you have a relocatable file that contains several modules and you're not exactly sure what you have. The command 'LM' can be used to scan the file and produce a listing of all modules along with entry points and external references. In order to do this, an FMP file must first be opened. This requires the 'RE' command, similar to the relocating loader. The complete console session with operator inputs underlined is as follows:

```
:RU,RELIA
/RELIA RE,%LUPRN::13
/RELIA LM

   # 1          NAM LUPRN,3,90
                ENT LUPRN
                EXT .MPY,.DIV,.DIO.,.IIO.,.IAY.,.DTA.,EXEC
                EXT CLRIO,IOR,IAND,IODVC,RMPAR,PAGE,IGET
                EXT DATE,MEMSZ,DVICE

   # 2          NAM DVICE,7,99
                ENT DVICE
                EXT .MPY,.ENTR

   # 3          NAM DATE,7,99
                ENT DATE
                EXT .MPY,.DIV,.ENTR

   # 4          NAM PAGE,7,99
                ENT PAGE
                EXT .ENTR,EXEC

   # 5          NAM IODVC,7,99
                ENT IODVC

   # 6          NAM MEMSZ,7,99
                ENT MEMSZ
                EXT .ENTR,$MATA

 END OF FILE
/RELIA EXIT
:
```

The LM command produces an abbreviated listing with no inversely assembled source code. To obtain the inversely assembled listing, the IA command could be used after first rewinding the file with the RW command.

The following example demonstrates searching for NAM, EXT, AND ENT names on FMP cartridges. This capability allows the user to mount a series of FMP cartridges and search one or all for a specific name. Assume that a cartridge contains all the RTE relocatable modules for system generation. Let's further assume that you suspect there might exist duplicate NAM records in more than one file. The following console session will illustrate how this can be done:

```
/RELIA SN,DSTAT   (Search all mounted CRN's for "NAM DSTAT")
 SEARCHING LU #41
 SEARCHING LU #2
 SEARCHING LU #3
 SEARCHING LU #42
  %GPSCM ON LU #42  CR #99
  %FMPC  ON LU #42  CR #99
  %FMPC2 ON LU #42  CR #99
  %DSTAT ON LU #42  CR #99
/RELIA SX,LOGLU   (Search for "EXT LOGLU")
 SEARCHING LU #41
 SEARCHING LU #2
 SEARCHING LU #3
 SEARCHING LU #42
  %DBUGR ON LU #42  CR #99
  %4LDR  ON LU #42  CR #99
  %BAIN1 ON LU #42  CR #99
  %GPSCM ON LU #42  CR #99
  %GCBIM ON LU #42  CR #99
/RELIA SX,REIO
 SEARCHING LU #41
 SEARCHING LU #2
 SEARCHING LU #3
 SEARCHING LU #42
  %FF4.N ON LU #42  CR #99
  %MSAFD ON LU #42  CR #99
  %RLIB1 ON LU #42  CR #99
  %RLIB2 ON LU #42  CR #99
  %4SP01 ON LU #42  CR #99
  %4SP02 ON LU #42  CR #99
  %BMLIB ON LU #42  CR #99
  %4LDR  ON LU #42  CR #99
  %SDLS4 ON LU #42  CR #99
  %SDS4  ON LU #42  CR #99
  %BAIN1 ON LU #42  CR #99
  %BASLB ON LU #42  CR #99
          .
          .
          .
          .
```

This program will soon be available from the Library of Contributed User Software (LOCUS) in the near future. Check with your local Hewlett-Packard Sales Representative.

BLOCKMODE INPUT WITH 264X SERIES TERMINALS

Frank Slootweg/HP Amstelveen, The Netherlands

The HP 264X Series CRT terminals give the user two different methods of performing blockmode input when using drivers DVR05 and DVA05. These two modes of operation are LINE mode and PAGE mode.

Line Mode

In line mode, the terminal transmits one line at a time to the computer. The advantage of this method is that only small buffers are required in System Available Memory (SAM) when performing input via re-entrant I/O (REIO) or via Class I/O. However, this method has the following disadvantage: for each additional line of input data there is an overhead of two characters (ESCd) for triggering the transmission of these lines to the computer.

This situation becomes even less optimal when multiple protected and unprotected fields are used per line because then only one unprotected field (e.g. part of a line) at a time is transmitted to the computer.

Page Mode

In page mode, the terminal transmits all the data on the screen from the current cursor position to the end of memory or to the next Record Separator, whichever occurs first. If the screen contains both protected and unprotected fields (e.g. format mode is on), only the data in the unprotected fields will be transmitted to the computer. The advantage of this method is that the transmission overhead is much less than it is for line mode. The disadvantages of page mode are:

- The subroutine REIO is limited to a maximum (input) buffer size of 260 characters. In many instances, the user's screen will contain much more than 260 characters. If REIO is used with an input buffer of more than 260 characters, it will use the "usual" method of performing input (calling EXEC 1), which means that the calling program cannot be swapped out of its partition during the input operation. Since it will take several minutes for an operator to input the data into the screen, this non-swappable condition is normally unacceptable.

- If Class I/O is used to overcome the 260 character limitation of REIO, another problem occurs. In this case during the time it takes the operator to fill in the screen, a block of SAM will be allocated to the terminal. For example, if each screen contains 1000 characters of unprotected fields and the system has eight terminals, then 8000 bytes of SAM will be allocated to the terminals nearly all of the time. Since SAM is a precious resource in RTE, this situation is also quite undesirable.

Alternate Methods

Unquestionably, it would be highly desirable to use only the advantages of each of the methods and to eliminate the disadvantages. The alternate methods which will be described below are intended to realize these goals.

1. ALTERNATE METHOD FOR 2645A AND SIMILAR TERMINALS

   This method is intended for 2645A and similar terminals (e.g. 2648A). A sample program A2645 appears in Figure 1. The procedure is described below:

   - Retrieve the LU number of the user's terminal (lines 38,39).

   - Issue an escape sequence to the terminal to put it in line mode and issues an update terminal configuration control request to the driver in order to inform the driver of this change (lines 43,44).

- Display the message "/A2645: BLOCKMODE WILL NOW BE SWITCHED ON" on the terminal for five seconds and then sends an escape sequence to the terminal to turn on block mode (lines 48 through 51).

- Send escape sequences to the terminal to home-up the cursor, to clear the display, and to disable the keyboard (lines 55 through 57).

- Write the form to the screen. In this example, the form consists of unprotected fields #1 and #2 on the first line, and fields #3 and #4 on the third and fifth lines respectively. All unprotected fields are thirty characters wide. The second and fourth lines are blank. (lines 61 through 64).

- Send escape sequences to home-up the cursor, to turn on format mode, and to enable the keyboard (lines 69 through 71).

- Issue a dummy read request of one word to wait until the operator has pressed the ENTER key. This read request requires only a few words of SAM and if necessary, the program can be swapped out of its partition during the time it takes the operator to fill in the screen.

- After the program continues (after the ENTER key has been pressed), disable the keyboard, home the cursor, switch the terminal to page mode, and inform the driver of this change of status (lines 77 through 80).

- Perform a "program enabled block read" by first sending the escape sequence ESCd to the terminal followed by a read operation with bits 9 and 10 of ICNWD set (lines 81 to 83). Note that for program enabled block read, the operator does not have to press the ENTER key again since this function is performed by the ESCd sequence.

- For purposes of demonstration, write the transmission log and the contents of the input fields to the display (lines 88 through 95).

- Perform "clean-up" operations by switching the terminal back to line mode, informing the driver of this change, enabling the keyboard, displaying a message, and switching off block mode (lines 100,108).

The described method has accomplished the following goals:

a) Only a few words of SAM are required during the operator input period (line 72).
b) The program is swappable during the operator input session.
c) The transmission overhead is as low as possible.

The only "disadvantage" is that the program is not swappable during the actual data transmission time if the user's input buffer is more than 260 characters. This is not critical since actual data transmission will typically take less than one second.

2. ALTERNATE METHOD FOR 2640A/B TERMINALS

This method is intended for 2640A/B terminals which cannot be switched from line to page mode by program control. However, this method is transportable to the 2645A and similar terminals. An example program B2640 appears in Figure 2. The major differences between A2645 and B2640 are highlighted below.

- The terminal is hardware-strapped for page mode (switch D on the Keyboard Interface Card is open). If this strap is never changed during on-line operation, the update terminal configuration control request in line 37 is not necessary. Switch E on the Keyboard Interface is in the open position (pressing CNTL is not required when using the function keys).

- The operator is asked to press down the block mode switch (lines 42 through 44).

- The program waits/loops until the F1 function softkey has been pressed. In this case, this key performs the equivalent of the normal "ENTER" function.
- The operator must manually release the block mode switch.

Conclusion

With a little special programming, the user can take full advantage of the versatile features of both the 264X series of terminals and the RTE operating system to accomplish a very efficient method of performing block mode input.

```
0001  FTN4,L
0002        PROGRAM A2645
0003  C
0004  C PROGRAM TO DO BLOCK/PAGE MODE INPUT WITH A 2645A TERMINAL.
0005  C
0006  C WRITTEN BY FRANK SLOOTWEG , HP/AMSTELVEEN, THE NETHERLANDS , 11DEC78
0007  C
0008        DIMENSION IBUF1(100),LU(5),ILOG(5)
0009        INTEGER HOME(2),CLEAR(2),KBDOF(2),KBDON(2),FMTON(2),FMTOF(2)
0010        INTEGER ESCD(2),PAGE(3),LINE(3),BLKON(3),BLKOF(3),STRTUN,ENDUN
0011        DATA HOME/15550B,57440B/,CLEAR/15512B,57440B/,KBDOF/15543B,57440B/
0012        DATAKBDON/15542B,57440B/,FMTON/15527B,57440B/,FMTOF/15530B,57440B/
0013        DATA ESCD/15544B,57440B/,PAGE/15446B,71461B,42137B/
0014        DATA LINE/15446B,71460B,42137B/,BLKON/15446B,65461B,41137B/
0015        DATA BLKOF/15446B,65460B,41137B/,STRTUN/15533B/,ENDUN/15535B/
0016  C
0017  C     THE FOLLOWING TERMINAL/DISPLAY CONTROL FUNCTIONS ARE AVAILABLE
0018  C     IN ARRAYS WITH THE NAMES :
0019  C
0020  C     HOME   IS  ESC h     (LITTLE H) CURSOR HOME
0021  C     CLEAR  IS  ESC J                CLEAR DISPLAY
0022  C     KBDOF  IS  ESC c     (LITTLE C) DISABLE KEYBOARD
0023  C     KBDON  IS  ESC b     (LITTLE B) ENABLE  KEYBOARD
0024  C     FMTON  IS  ESC W                FORMAT MODE ON
0025  C     FMTOF  IS  ESC X                FORMAT MODE OFF
0026  C     ESCD   IS  ESC d     (LITTLE D) BLOCK TRANSFER ENABLE FROM COMPUTER
0027  C     PAGE   IS  ESC &s1D  (LITTLE S) SET KEYBOARD INTERFACE SWITCH D TO
0028  C                                     PAGE MODE
0029  C     LINE   IS  ESC &s0D  (LITTLE S) SET KEYBOARD INTERFACE SWITCH D TO
0030  C                                     LINE MODE
0031  C     BLKON  IS  ESC &k1B  (LITTLE K) SET BLOCK MODE ON
0032  C     BLKOF  IS  ESC &k0B  (LITTLE K) SET BLOCK MODE OFF
0033  C     STRTUN IS  ESC [                START UNPROTECTED FIELD
0034  C     ENDUN  IS  ESC ]                END UNPROTECTED FIELD
0035  C
0036  C     GET LU NUMBER OF TERMINAL
0037  C
0038        CALL RMPAR(LU)
0039        IF((LU.LT.1).OR.(LU.GT.63)) LU=1
0040  C
0041  C     SWITCH TERMINAL TO LINE MODE AND UPDATE TERMINAL CONFIGURATION
0042  C
0043        CALL REIO(2,LU,LINE,-6)
0044        CALL EXEC(3,LU+2500B)
0045  C
0046  C     DISPLAY MESSAGE FOR FIVE SECONDS , THEN TURN ON BLOCK MODE
```

```
0047  C
0048        WRITE(LU,1000)
0049  1000  FORMAT("/A2645: BLOCKMODE WILL NOW BE SWITCHED ON !")
0050        CALL EXEC(12,0,2,0,-5)
0051        CALL REIO(2,LU,BLKON,-6)
0052  C
0053  C     HOME CURSOR , CLEAR DISPLAY AND DISABLE KEYBOARD
0054  C
0055        CALL REIO(2,LU,HOME,-3)
0056        CALL REIO(2,LU,CLEAR,-3)
0057        CALL REIO(2,LU,KBDOF,-3)
0058  C
0059  C     WRITE THE FORM TO THE SCREEN
0060  C
0061        WRITE(LU,1010)STRTUN,ENDUN,STRTUN,ENDUN
0062  1010  FORMAT("FIELD 1 : ",A2,30X,A2,"FIELD 2 : ",A2,30X,A2)
0063        WRITE(LU,1020)STRTUN,ENDUN,STRTUN,ENDUN
0064  1020  FORMAT("FIELD 3 : ",A2,30X,A2,//,"FIELD 4 : ",A2,30X,A2)
0065  C
0066  C     HOME CURSOR , TURN FORMAT MODE ON , ENABLE KEYBOARD AND WAIT FOR
0067  C     THE ENTER KEY TO BE PRESSED
0068  C
0069        CALL REIO(2,LU,HOME,-3)
0070        CALL REIO(2,LU,FMTON,-3)
0071        CALL REIO(2,LU,KBDON,-3)
0072        CALL REIO(1,LU,IDUMY,-2)
0073  C
0074  C     DISABLE KEYBOARD , HOME CURSOR , SWITCH TERMINAL TO PAGE MODE ,
0075  C     UPDATE TERMINAL CONFIGURATION AND DO A PROGRAM ENABLED BLOCK READ
0076  C
0077        CALL REIO(2,LU,KBDOF,-3)
0078        CALL REIO(2,LU,HOME,-3)
0079        CALL REIO(2,LU,PAGE,-6)
0080        CALL EXEC(3,LU+2500B)
0081        CALL REIO(2,LU,ESCD,-3)
0082        CALL REIO(1,3000B+LU,IBUF1,-200)
0083        CALL ABREG(IA,IB)
0084  C
0085  C     TURN FORMAT MODE OFF , WRITE TRANSMISSION LOG AND INPUT FIELDS TO
0086  C     DISPLAY
0087  C
0088        CALL REIO(2,LU,FMTOF,-3)
0089        WRITE(LU,1030)IB
0090  1030  FORMAT(/,"/A2645: LOG = ",I6)
0091        WRITE(LU,1040)
0092  1040  FORMAT("FIELDS 1 THROUGH 4 ARE :",/)
0093        DO 45 I=1,4
0094        CALL REIO(2,LU,IBUF1((I-1)*15+1),-30)
0095  45    CONTINUE
0096  C
0097  C     SWITCH TERMINAL TO LINE MODE , UPDATE TERMINAL CONFIGURATION ,
0098  C     ENABLE KEYBOARD , TURN BLOCK MODE OFF AND FINISH
```

```
0099  C
0100        CALL REIO(2,LU,LINE,-6)
0101        CALL EXEC(3,LU+2500B)
0102        CALL REIO(2,LU,KBDON,-3)
0103        WRITE(LU,1050)
0104  1050  FORMAT("/A2645: BLOCKMODE WILL NOW BE SWITCHED OFF AGAIN !")
0105        CALL REIO(2,LU,BLKOF,-6)
0106        WRITE(LU,1060)
0107  1060  FORMAT(/,"/A2645: END",//)
0108        END
```

Figure 1

```
0001  FTN4,L
0002        PROGRAM B2640
0003  C
0004  C PROGRAM TO DO BLOCK/PAGE MODE INPUT WITH A 2640B TERMINAL.
0005  C
0006  C WRITTEN BY FRANK SLOOTWEG , HP/AMSTELVEEN, THE NETHERLANDS , 11DEC78
0007  C
0008        DIMENSION IBUF1(100),LU(5),ILOG(5)
0009        INTEGER HOME(2),CLEAR(2),KBDOF(2),KBDON(2),FMTON(2),FMTOF(2)
0010        INTEGER ESCD(2),STRTUN,ENDUN
0011        DATA HOME/15550B,57440B/,CLEAR/15512B,57440B/,KBDOF/15543B,57440B/
0012        DATAKBDON/15542B,57440B/,FMTON/15527B,57440B/,FMTOF/15530B,57440B/
0013        DATA ESCD/15544B,57440B/,STRTUN/15533B/,ENDUN/15535B/
0014  C
0015  C     THE FOLLOWING TERMINAL/DISPLAY CONTROL FUNCTIONS ARE AVAILABLE
0016  C     IN ARRAYS WITH THE NAMES :
0017  C
0018  C     HOME   IS  ESC h     (LITTLE H) CURSOR HOME
0019  C     CLEAR  IS  ESC J                CLEAR DISPLAY
0020  C     KBDOF  IS  ESC c     (LITTLE C) DISABLE KEYBOARD
0021  C     KBDON  IS  ESC b     (LITTLE B) ENABLE  KEYBOARD
0022  C     FMTON  IS  ESC W                FORMAT MODE ON
0023  C     FMTOF  IS  ESC X                FORMAT MODE OFF
0024  C     ESCD   IS  ESC d     (LITTLE D) BLOCK TRANSFER ENABLE FROM COMPUTER
0025  C     STRTUN IS  ESC [                START UNPROTECTED FIELD
0026  C     ENDUN  IS  ESC ]                END UNPROTECTED FIELD
0027  C
0028  C     GET LU NUMBER OF TERMINAL
0029  C
0030        CALL RMPAR(LU)
0031        IF((LU.LT.1).OR.(LU.GT.63)) LU=1
0032  C
0033  C     UPDATE TERMINAL CONFIGURATION TO BE SURE THAT THE DRIVER KNOWS
0034  C     THAT THE TERMINAL IS STRAPPED FOR PAGE MODE (SWITCH/STRAP D
0035  C     ON KEYBOARD INTERFACE IS OPEN/NOT-INSTALLED)
0036  C
0037        CALL EXEC(3,LU+2500B)
0038  C
0039  C     DISPLAY MESSAGE FOR FIVE SECONDS TO ALLOW OPERATOR TO PRESS DOWN
0040  C     THE BLOCK MODE SWITCH
0041  C
0042        WRITE(LU,1000)
0043  1000  FORMAT("/B2640: PLEASE PRESS DOWN BLOCK MODE SWITCH!")
0044        CALL EXEC(12,0,2,0,-5)
```

```
0045  C
0046  C     HOME CURSOR , CLEAR DISPLAY AND DISABLE KEYBOARD
0047  C
0048        CALL REIO(2,LU,HOME,-3)
0049        CALL REIO(2,LU,CLEAR,-3)
0050        CALL REIO(2,LU,KBDOF,-3)
0051  C
0052  C     WRITE THE FORM TO THE SCREEN
0053  C
0054        WRITE(LU,1010)STRTUN,ENDUN,STRTUN,ENDUN
0055  1010  FORMAT("FIELD 1 : ",A2,30X,A2,"FIELD 2 : ",A2,30X,A2)
0056        WRITE(LU,1020)STRTUN,ENDUN,STRTUN,ENDUN
0057  1020  FORMAT("FIELD 3 : ",A2,30X,A2,//,"FIELD 4 : ",A2,30X,A2)
0058  C
0059  C     HOME CURSOR , TURN FORMAT MODE ON , ENABLE KEYBOARD AND WAIT FOR
0060  C     THE "F1" FUNCTION KEY TO BE PRESSED
0061  C
0062        CALL REIO(2,LU,HOME,-3)
0063        CALL REIO(2,LU,FMTON,-3)
0064        CALL REIO(2,LU,KBDON,-3)
0065  10    CALL REIO(1,LU,IDUMY,-2)
0066  C
0067  C     CHECK FOR RIGHT FUNCTION KEY , IF WRONG KEY THEN IGNORE INPUT
0068  C
0069        IF(IDUMY.NE.15560B) GOTO 10
0070  C
0071  C     DISABLE KEYBOARD , HOME CURSOR AND DO A PROGRAM ENABLED BLOCK READ
0072  C
0073        CALL REIO(2,LU,KBDOF,-3)
0074        CALL REIO(2,LU,HOME,-3)
0075        CALL REIO(2,LU,ESCD,-3)
0076        CALL REIO(1,3000B+LU,IBUF1,-200)
0077        CALL ABREG(IA,IB)
0078  C
0079  C     TURN FORMAT MODE OFF , WRITE TRANSMISSION LOG AND INPUT FIELDS TO
0080  C     DISPLAY
0081  C
0082        CALL REIO(2,LU,FMTOF,-3)
0083        WRITE(LU,1030)IB
0084  1030  FORMAT(/,"/B2640: LOG = ",I6)
0085        WRITE(LU,1040)
0086  1040  FORMAT("FIELDS 1 THROUGH 4 ARE :",/)
0087        DO 45 I=1,4
0088        CALL REIO(2,LU,IBUF1((I-1)*15+1),-30)
0089  45    CONTINUE
0090  C
0091  C     ENABLE KEYBOARD  AND FINISH
0092  C
0093        CALL REIO(2,LU,KBDON,-3)
0094        WRITE(LU,1050)
0095  1050  FORMAT("/B2640: PUT BLOCK MODE SWITCH BACK IN UP-POSITION !")
0096        WRITE(LU,1060)
0097  1060  FORMAT(/,"/B2640: END",//)
0098        END
```

Figure 2

"TWEAKING" INTERNALS

Harvey Bernard/HP Rockville

There are two types of HP 1000 programmers: those who make use of the operating system through standard methods, and those who are content only when they can "tweak" the internals of the system. I suspect many of our "standard" users would secretly delight in a bit of tweaking and are envious of their counterparts, the "systems types." This article is for those of us in the former class, who, Walter Mitty-like, dream about "going privileged" or "doing our own mapping" and other fantastic things.

We will reference the code in Figure 1 by number throughout this article.

Map driver partition into ADDVR's address space:

```
[1] ------ LDX $DLTH          Get driver partition size

[2] ------ LDA D32
           ADA $DVPT          Compute 1st ADDVR driver partition page

[3] ------ LDB $DVMP          Get 1st physical page of driver
           ADB EQTN             partition to be written into
           LDB 1,I            B-Reg = Physical page

[4] ------ SSB                Skip if driver in driver partition
           JMP N1             Driver resides in SDA, no special mapping

           JSB $LIBR          Turn memory protect off
[5] ------ NOP
           XMS                Map driver partition into user's address

[6] ------ LDA SBUF           Save user map into temp buffer
           USA                Save map

           LDA D33            Map Reg(1) = Map Reg(0) to access hidden BP
           LDB LBUF,I         Get Map Reg(0)
[7] ------ LDX B1             Change 1 Reg
           .
           .
           XMS

           LDA LBUF           Move temp buffer to system DMS buffer
[8] ------ LDB B3740             in hidden base page
           MVW D32

[9] ------ LDA LBUF           Restore user map from temporary buffer
           USA

           JSB $LIBX          Turn memory protect back on
[10] ----- DEF *+1
           DEF *+1

     N1    NOP
           .
           .
           .
     B1    OCT 1
     D32   DEC 32
     D33   DEC 33
     B3740 OCT 3740
     EQTN  NOP
[11] LBUF  DEF BUFF           Set up to load from BUFF with USA
[12] SBUF  DEF BUFF,I         Set up to store to BUFF with USA
     BUFF  BSS 32
```

Figure 1

This tricky little specimen was written by David Hoffman of Data Systems Division to supplement a program used in the "Driver Writing" course. Putting ourselves in Dave's place, we will try to develop the same code from scratch.

The problem is as follows: A program called ADDVR, written for RTE-III (originally written by Gary McCarney, HP Rockville), allows a user to overwrite an RTE driver with the absolute code of a user written driver on-line for the purpose of testing it. However, unlike RTE-III, RTE-IV maps in the driver a program needs only when it needs it. Hence, to convert ADDVR for RTE-IV, we need to map the driver to be overwritten into ADDVR's 32 page address space. More precisely, we need to put the page numbers of the appropriate driver partition into the right mapping registers. Figure 2 illustrates our objectives.

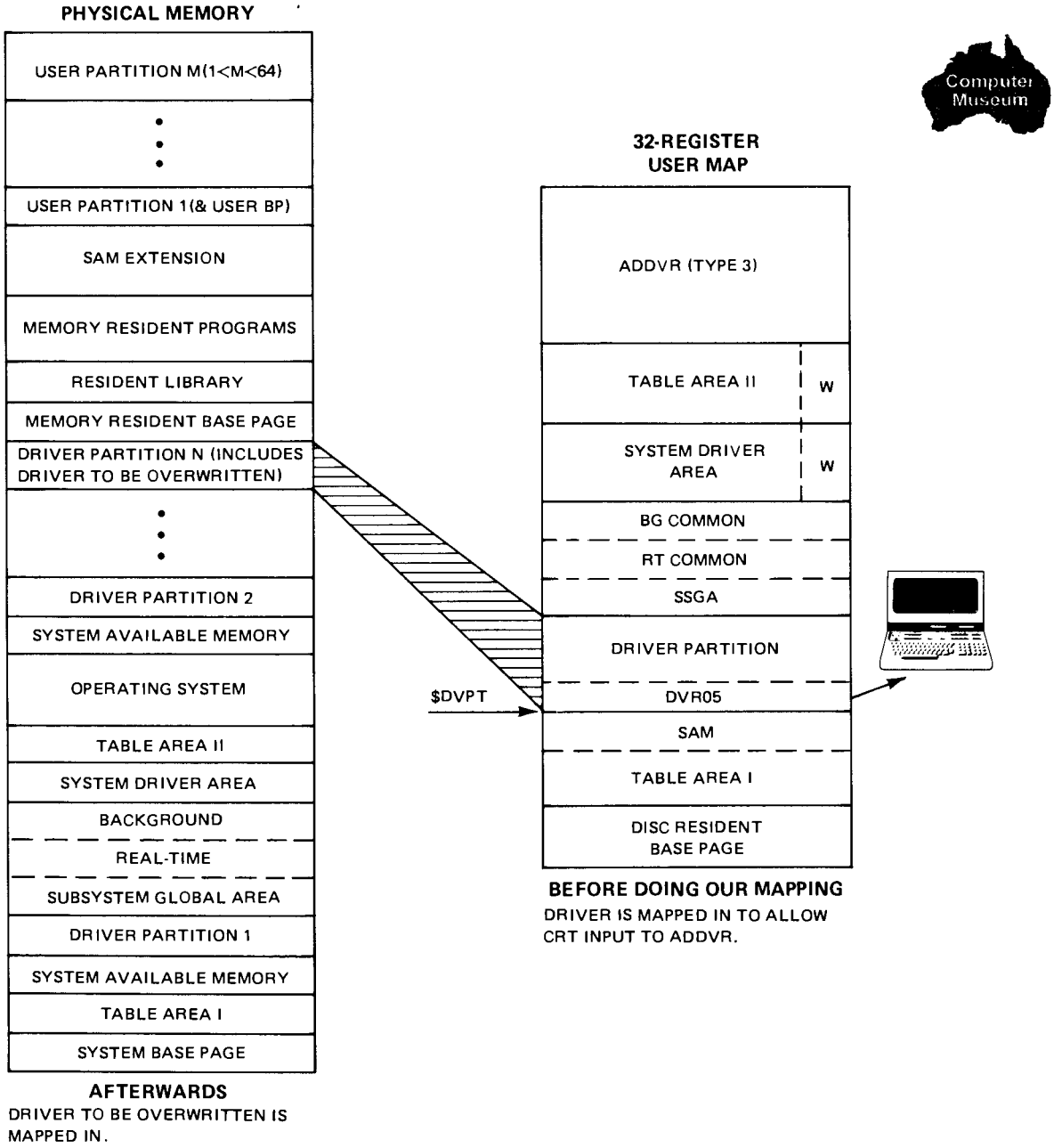
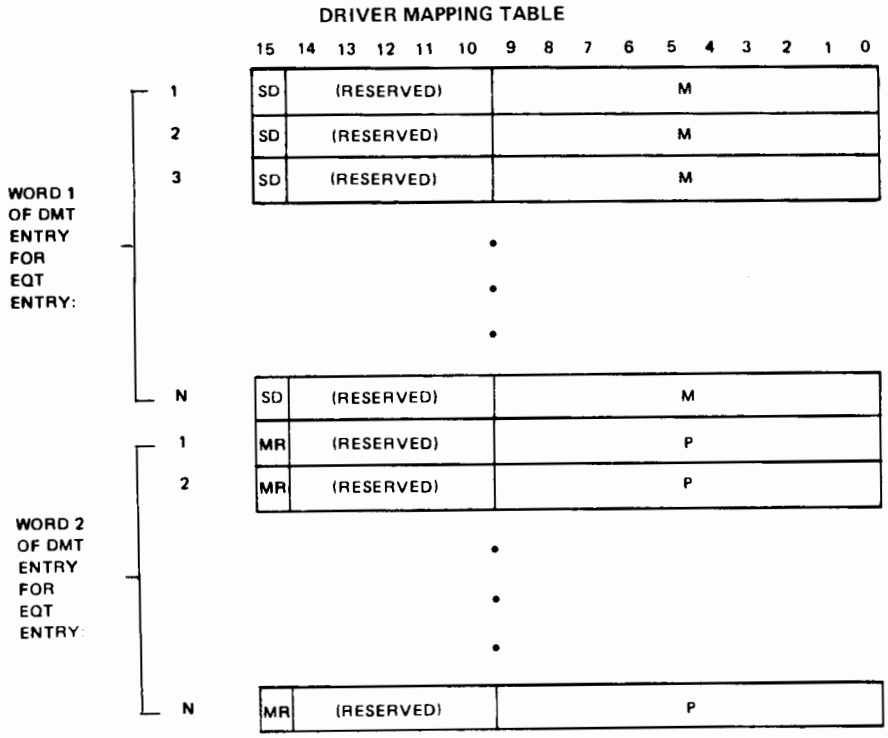


Figure 2

Having defined our objectives, we will need the following manuals: "RTE-IV Programmer's Reference Manual" (Rev. 1840), to show us how driver partitions and the base page are structured (I will refer to this manual as "RTE-IV"); "RTE-IV Student Workbook" for the Advanced RTE course, to describe various system entry points (this will be referenced as the "Workbook"); and finally, the "21MX-E (or F) Series Computer Operating and Reference Manual" (to be referred to as "21MX-E"), and used to explain dynamic mapping.

Now we can plan our attack. When the user runs ADDVR, he passes the program the LU of the driver to be overwritten, so that we can obtain the corresponding EQT from the Device Reference Table (see page V-7 of "RTE-IV"). Then we can find the page number of the correct driver partition from the Driver Mapping Table (DMT). Refer to Figure 3.



WHERE:

SD = 0 IMPLIES DRIVER RESIDES IN A DRIVER PARTITION, AND
M = STARTING PAGE NUMBER OF PARTITION IN BITS 0-9

SD = 1 IMPLIES DRIVER RESIDES IN SYSTEM DRIVER AREA, AND
M = 0 IMPLIES DRIVER NOT DOING ITS OWN MAPPING
M = 1 IMPLIES DRIVER DOING ITS OWN MAPPING

MR = 1 IMPLIES THAT THE I/O REQUEST BUFFER IS LOCATED IN A MEMORY RESIDENT PROGRAM.
(P VALUE NOT SIGNIFICANT – RESERVED FOR FUTURE USE)

MR = 0 IMPLIES THAT THE I/O REQUEST BUFFER IS NOT LOCATED IN A MEMORY RESIDENT PROGRAM. BUFFER LOCATION IS INDICATED BY THE VALUE OF P, AS FOLLOWS:
P = 0 IMPLIES BUFFER IS IN THE SYSTEM AREA
P NOT ZERO IMPLIES BUFFER IS LOCATED IN A DISC RESIDENT PROGRAM. P IS THE PHYSICAL PAGE NUMBER OF THE PROGRAM'S BASE PAGE

N = NUMBER OF EQT ENTRIES IN SYSTEM (BP 1651)

$DVMP = ADDRESS OF DRIVER MAPPING TABLE
$DVPT = LOGICAL START PAGE OF DRIVER PARTITION
$DLTH = $ PAGES PER DRIVER PARTITION

Figure 3

The entry in the DMT tells us whether the driver is in SDA (System Driver Area) or in a driver partition. If in SDA, our job is done, because ADDVR is a Type 3 program and therefore has SDA already mapped in. Otherwise, the DMT entry gives us the starting page number of the driver partition.

"How do we find the DMT?", you ask. On page 3-8 of our "Workbook" we find the system entry point $DVMP pointing to it. Voila! Below is the code to get the DMT entry and check for SDA.

```
                LDB $DVMP
(No. 3, Fig. 1) ADB EQTN        Get right entry
                LDB 1,I

(No. 4)         SSB             Check for SDA.
                JMP N1
```

Okay, that was easy. Now let's track down the appropriate mapping registers for our driver partition. Figure 4 demonstrates the mapping registers numbering scheme.

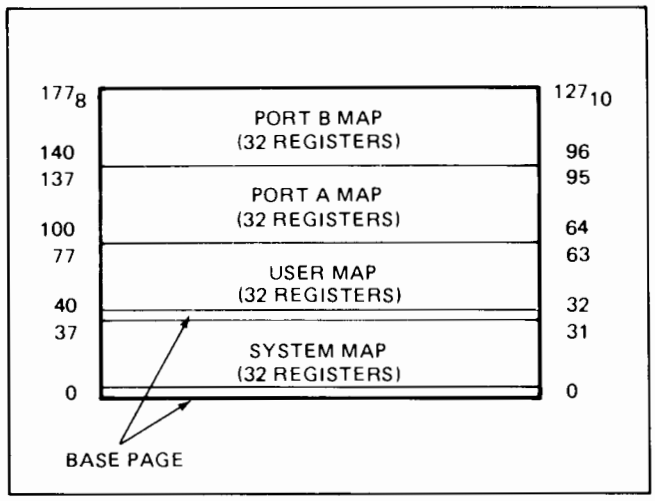


Figure 4

The number of registers needed to map the entire driver partition corresponds to the number of pages in the partition, which according to our "Workbook" is contained in $DLTH. Typically, this length will be two, which is the default size set at system generation.

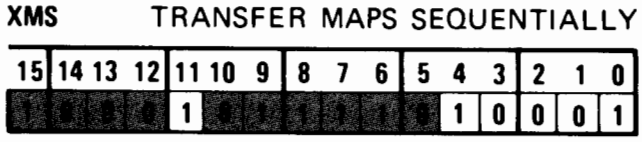
```
(No. 1)         LDX $DLTH
```

Slowly, the pieces are beginning to fit together.

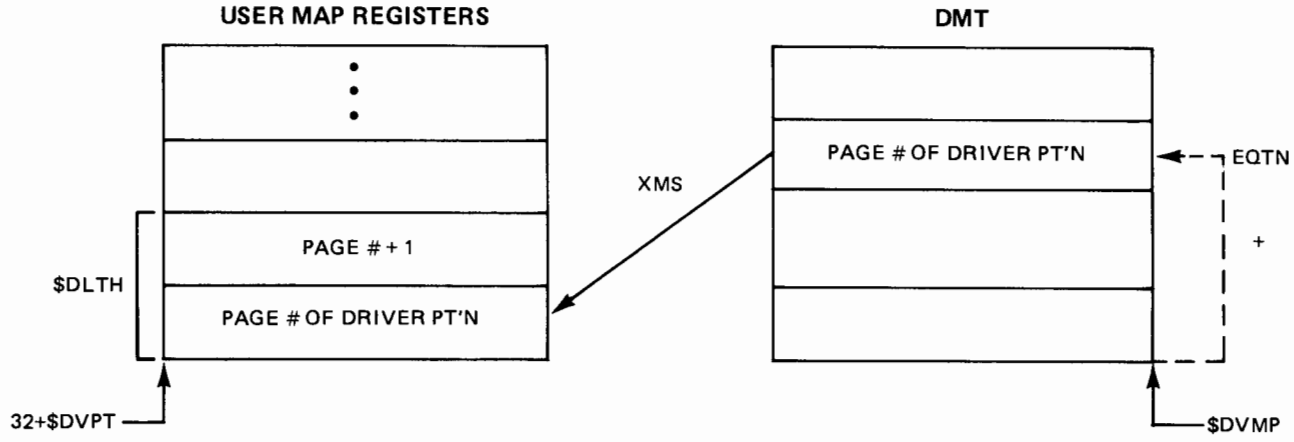
According to our "Workbook" on page 3-8, $DVPT contains the relative page number of the driver partition in the user map (see Figure 2). Hence, 32+$DVPT should give us the right register number. Here is the code:

```
(No. 2, Fig. 1) LDA D32
                ADA $DVPT
```

All we need is one super instruction to put the right numbers in the right places. It just so happens . . . .



The XMS instruction works as follows:



All we have to do to make the instruction XMS perform is to load the A, B, and X registers as in Numbers 1, 2, and 3 in Figure 1.

There is one hitch. As "21MX-E" indicates, XMS is a "privileged instruction"; that is, it will generate a DM (dynamic mapping) error in normal mode. To suppress the DM violation interrupt we need to turn off the interrupt system. On page X-3 of "RTE-IV" we find a powerful system library routine, $LIBR, which will do this for us. Below is the correct format:

(No. 5)

```
JSB $LIBR
NOP
XMS
```

We would be finished tweaking but for one small detail. RTE saves a copy of the user map in what has become known as "the hidden base page," which is the upper 32 words of the user or local base page (see page V-15 of "RTE-IV"). In this way, the system does not have to rebuild the user map after each external interrupt. To be consistent with this procedure we must store our recently created user map in our partition's base page. Unfortunately, the 32 words in question are truly "hidden." As Figure 5 illustrates, the logical base page consists of a portion of the system base page (see A) and part of the user's base page (see B).

No matter how we try to access the upper 32 words of the physical base page (C of Figure 5), instead we get the system communication area. As explained in "21MX-E," the culprit is the "base page fence," preventing our entrance to the upper part of the base page. In that case, let us enter through the back door. Hypothesize page 51 to be the user's base page and consider the following diagram:

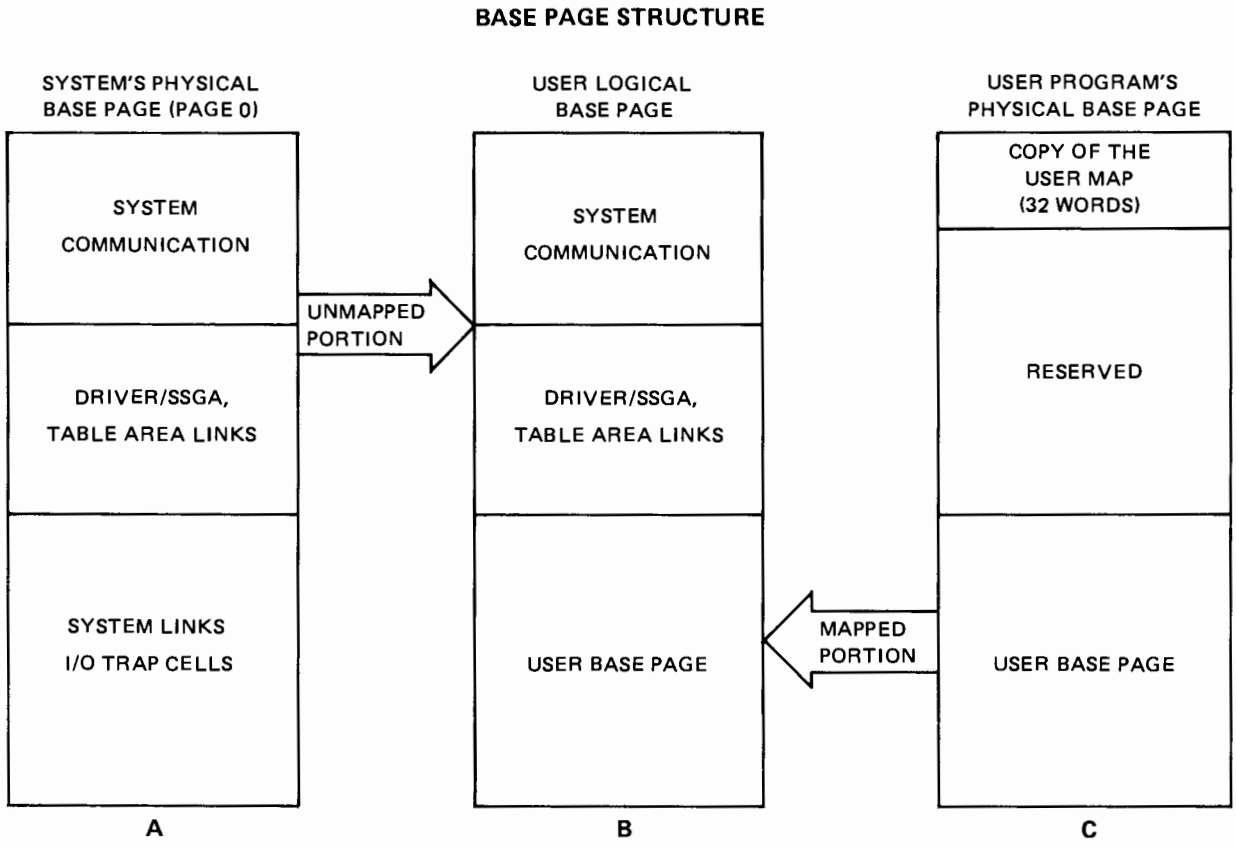
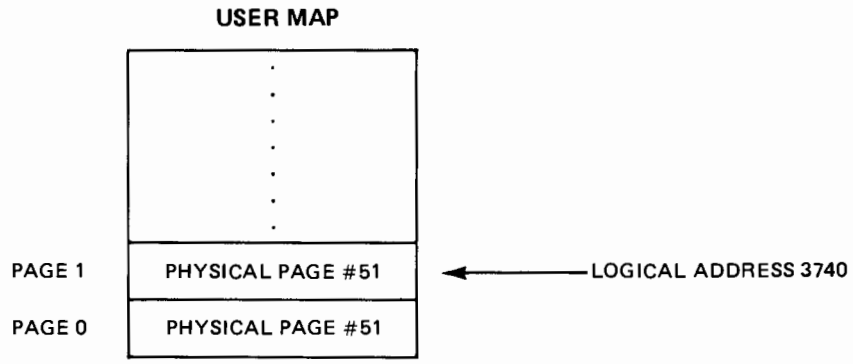


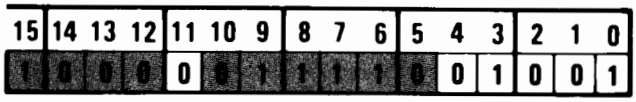
Figure 5

By moving our new 32-word map into logical page 1 (see previous diagram), we will not be hindered by the base page fence. Furthermore, moving it to location 3740 octal (as illustrated) will move it, in reality, to the upper 32 words of the local base page.



STEP ONE to accomplish our task is to write our user map into a memory buffer. We use the instruction . . . .

USA LOAD/STORE USER MAP PER A



The code is:

(No. 6)

```
LDA SBUF
USA
```

where

(No. 12)

```
SBUF DEF BUFF,I
BUFF BSS 32
```

STEP TWO is to move the contents of register 32 (user map--base page) to register 33 (page 1) using the XMS instruction as shown below:

(No. 7 and 11)

```
LDA D33
LDB LBUF,I
LDX B1
XMS
```

STEP THREE is to move the 32-word memory buffer to address 3740.

(No. 8 and 11)

```
LDA LBUF
LDB B3740
MVW D32
```

and we have saved the user map on the "hidden base page!"

All that remains is to restore our former user map . . .

(No. 9)

```
LDA LBUF
USA
```

and turn the interrupt system back on

(No. 10)

```
JSB $LIBX
DEF *+1
DEF *+1
```

(See page X-3 of "RTE-IV")

and ADDVR can overwrite the driver of its choice to its heart's content.

I hope this article has been enlightening and perhaps enjoyable. Looking back at Figure 1 you can see that everything we have discussed happened in thirty lines of code. The entire matter is quite straightforward, isn't it?

References:

1. "RTE-IV Programmer's Reference Manual", 92067-90001, Rev. 1840.

2. "RTE-IV Student Workbook" for the Advanced RTE Course, 22999-90200, Print Date August 1978.

3. "21MX-E Series Computer Operating and Reference Manual", 02109-90014, Print Date August 1977.

A METHOD FOR SMOOTH CURVE FITTING

Larry W. Smith/HP Fullerton

A number of HP 1000 users have expressed the need for a good point-to-point curve fitting routine. Since there is not an HP designed routine nor a contributed one in LOCUS, I have done considerable research and found a method that appears both simple to understand and flexible to use.

The contents of this article presents a mathematical method of fitting a smooth curve to a set of given points in a two-dimensional plane. The FORTRAN subroutine which implements this method appears at the end of this article.

Introduction

In order to determine the relationship between two variables (i.e. two entities that have the ability to change from one value or state to another), we usually perform computations or make measurements to determine the nature of the relative change. The resulting data could be represented as a set of discrete (known) points in a plane. If we know that the relationship between these points can be mathematically described and visually represented by a smooth curve; then our next step is to try to fit a smooth curve to this set of points. The result desired is a curve which passes through all the given points. This process is best known as Interpolation. Before the advent of computers, these curves were manually drawn by well-trained scientists and engineers and usually resulted in a reasonably good looking graph. In order to let the computer draw a curve, we must provide the points themselves and a complete set of detailed instructions.

Existing Curve Fitting Methods

There are several well known mathematical methods for interpolating the value of a function expressed by a given set of values. Some of these can be found in F. B. Hildebrand's Introduction to Numerical Analysis (1956) in chapters 2, 3, 4 and 9; in W. E. Milne's Numerical Calculus (1949) in chapter 3; and, in A. Ralston and H. S. Wilf's Mathematical Methods for Digital Computers, Volume II (1967), in chapter 8. The application of any of the above methods sometimes results in a curve that is visually quite different than one drawn manually. That is to say, the resulting curve sometimes appears strange and unnatural. The technique presented in this article will describe a method of interpolation that produces a smooth and natural looking curve.

A Brief Discussion of Existing Methods

Let's assume that the values of X and Y at 11 points taken from a sideband distortion study are as shown in Figure 1.

X	0	1	2	3	4	5	6	7	8	9	10
Y = Y (X)	10	10	10	10	10	10	10.5	15	50	60	85

Figure 1

Assuming that we know the physical phenomena (i.e., variations in sideband frequency distortion) can be represented by Y(X), a single-valued smooth function of X, we must fit a smooth curve through all the given points by interpolating the value of Y(X). If we use the method of interpolation based on polynomials (Milen, Hildebrand) or several other variations by Newton-Cotes, Lagrange, Aitken, or Neville, each with its own advantages and disadvantages, we must realize that each is based on the common assumption that Y(X) can be closely approximated by a polynomial of X of order N-1, where N is the number of points. This might lead one to assume that they should all give the same result since the uniqueness of a polynomial of N-1 with given values of Y(X) at N points has been proven by Hildebrand (page 44). If we apply a 10th order Lagrangian polynomial to the set of 11 points in Figure 1 by collocating each, the result is shown in Figure 2.

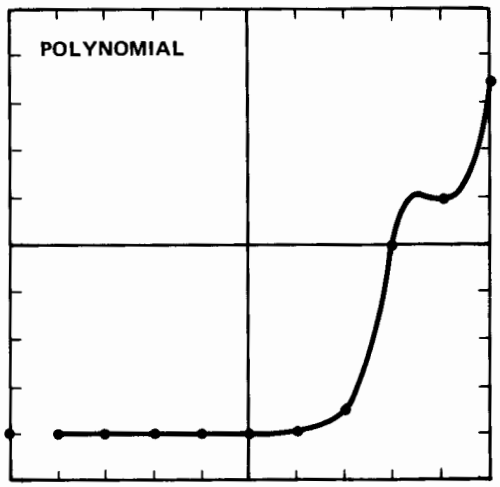


Figure 2

The next method is based on a ratio of two polynomials (called a "Rational Function") by Hildebrand (sections 9.9-9.12). Although this method produces a better looking and closer approximated curve at each point, its function does not always exist; and, non-singularity of the function cannot be guaranteed. If we were to omit the first point (0,10), then the function would exist and appear as shown in Figure 3.

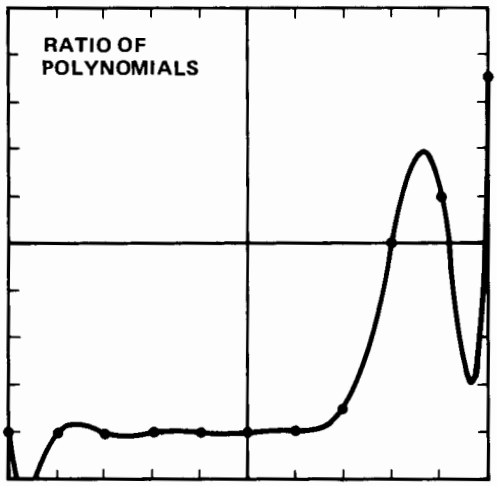


Figure 3

The problem is that if we make any assumptions concerning the functional form for a whole set of points other than continuity and smoothness of the curve, it is inevitable that the curve will behave strangely. On the other hand, when we try to fit a smooth curve manually, we do not make any assumptions about the functional form for the whole curve. We draw a portion of the curve based on a small amount of the whole curve without using the whole set of points. This local procedure is the basis for the method described in this article. As a further illustration, you might note that although the Spline function employs a piecewise function composed of a set of polynomials, all polynomials are evaluated simultaneously on the basis that the function and its derivitives are in the whole range, and that no individual polynomials can be determined locally.

The New Curve Fitting Method

To best illustrate how this method works, you might visualize creating a computerized computational procedure that emulates the skill of the best human curve-fitter in the world. This method is devised such that it can handle a single-valued or a multiple-valued function depending upon whether we know in advance that the given data points represent one case or the other.

The method is based on a piecewise function given by a third order polynomial for a single-valued function, and by a pair of third order polynomials for a multiple-valued function. For a single-valued function, continuity of the function and of its first-order derivative (i.e. the direction of the tangent to the curve or the slope of the curve) is assumed. For the second order derivative, we determine the direction of the tangent locally under certain assumptions. By doing this we can fit a curve piecewise to the given set of data points without having discontinuties in the curve and its slope.

The portion of the curve between any pair of points is assumed to be determined only by its coordinates and slope. However, since the slope of the curve should be determined at the end points as well, estimation of two more points at each end point is necessary.

Direction of the Tangent

Let's assume that the direction of the tangent to the curve (i.e. the slope) at a given point p_i is determined by the coordinates of five points, p_{i-2}, p_{i-1}, p_i, p_{i+1}, and p_{i+2}. In other words, the points more than two intervals away are assumed not to effect the determination of the slope.

Consider five points, 1,2,3,4, and 5 as shown in Figure 8 below:

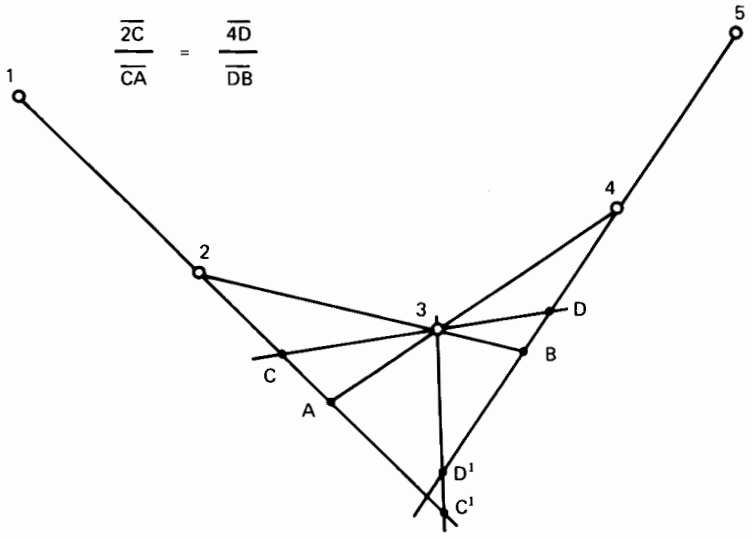


Figure 8

Let the point of intersection of the two straight lines extended from line segments 12 and 34 be denoted by A and a similar point corresponding to line segments 23 and 45 be denoted by B. Our task is to seek a reasonable condition for determining the direction of the tangent CD at point 3.

It seems plausible that the direction of CD should approach that of 23 when the direction of 12 approaches that of 23, and that the angle 23C (i.e. the angle between 32 and 3C) should be equal to D34 when I23 is equal to 345. With these reasonable assumptions as a guideline, the condition of determining the direction of CD is still not unique. For simplicity, we assume that the tangent CD is determined by the condition:

$$\frac{2C}{CA} = \frac{4D}{DB}$$

However, this condition does not exist for certain configurations of five points as illustrated in Figure 9.

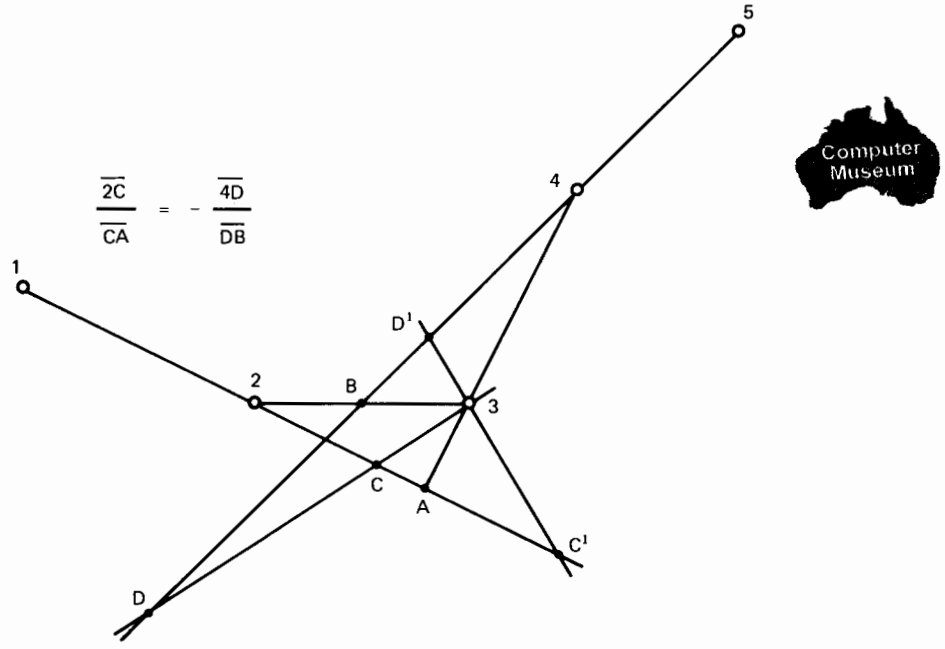


Figure 9

In this case the alternate condition,

$$\frac{2C}{CA} = -\frac{4D}{DB}$$

does exist and we shall use this condition.

Summarizing the above two conditions, we assume that the direction of the tangent CD is determined by the following general condition:

$$\frac{2C}{CA} = \pm \frac{4D}{DB} \qquad (1)$$

The sign (i.e., + or −) to be used depends on the configuration of the five points. The sign for which the condition exists should be selected.

Examples

An example of the application of this method is shown in figure 7 where the curve is very close to the one in figure 6 determined manually:

Figures 10-15 gives some examples of the application of the method in different modes. Two curves drawn for single-valued functions Y=Y(X) (MODE=1) and X=X(Y) (MODE=2) appear in figures 10 and 11, respectively. Two examples for multiple-valued nonclosed curves (MODE=3) are shown in figures 12 and 13. A circle and an ellipse are drawn in figures 14 and 15, respectively, as examples for the case of a closed curve (MODE=4).

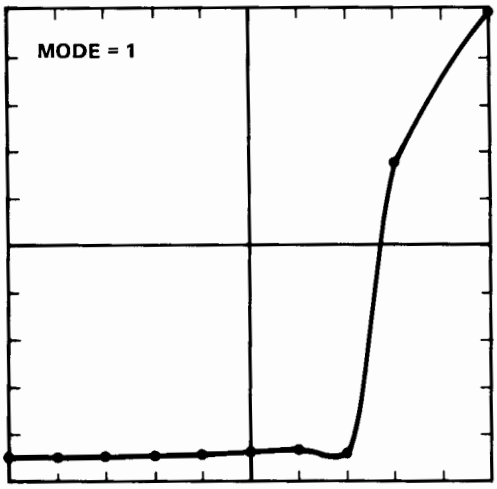


Figure 10

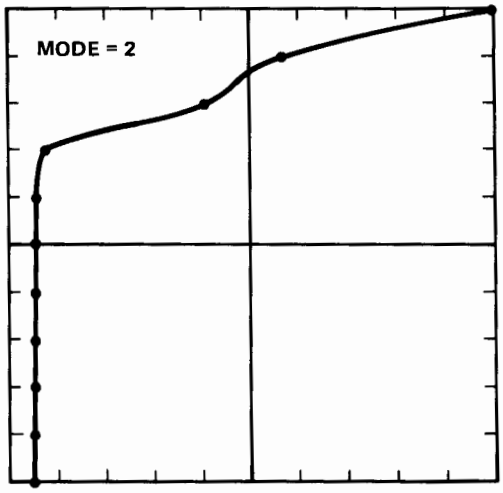


Figure 11

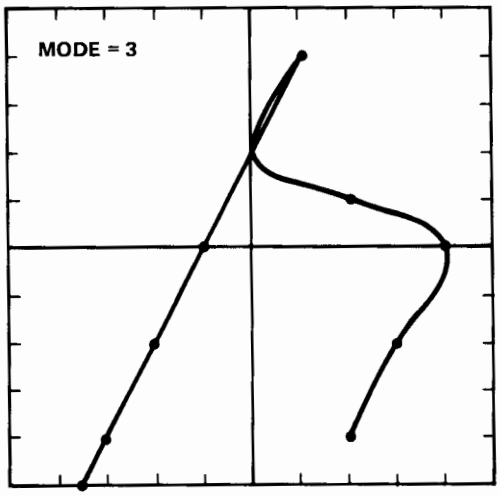


Figure 12

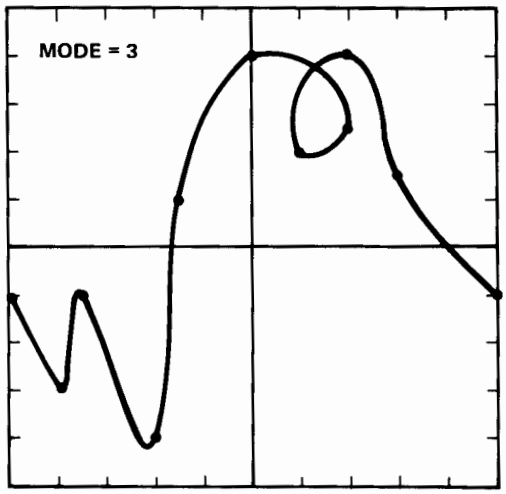


Figure 13

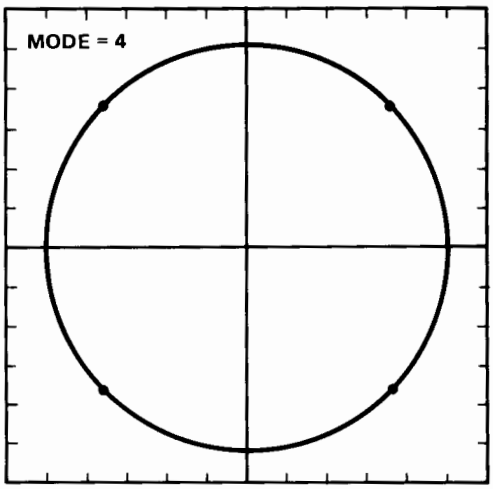


Figure 14

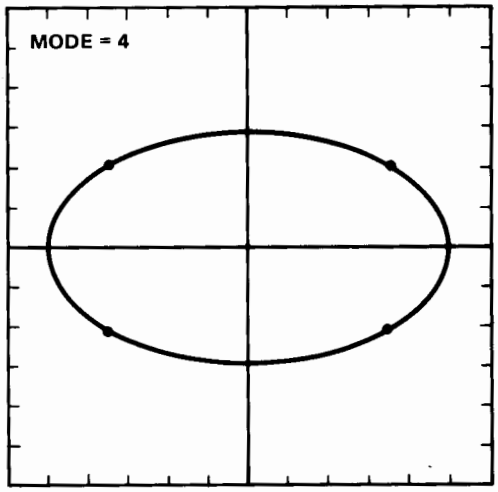


Figure 15

Concluding Remarks

We have described a method of smooth curve fitting. For proper application of this method, the following aspects should be taken into consideration:

1. The curve obtained by this method passes through all the given points; Therefore, the method is applicable only to the case where the piecewise values of the point coordinates are given;

2. Use of this method is not recommended when given data points manifest apparent regularity or when we have a prior knowledge on the regularity of the data;

3. As is true for any method of interpolation, no guarantee can be given of the accuracy of the interpolation, unless the method in question has been checked in advance against precise values or a functional form;

4. The method yields a smooth and natural-looking curve and is therefore most useful in cases where manual, but tedious, curve fitting will do in principle;

5. For a single-valued function, the resultant curve is invariant under a linear-scale transformation of the coordinate system. In other words, different scalings of the coordinates result in a similar (if not the same) looking curve;

6. For a multiple-valued function, the resultant curve is variant under a linear-scale transformation of the coordinate system; the scalings of the coordinates should be coincident with the actual size of the graph.

An HP 1000 subroutine written in FORTRAN IV, named CRVFT, has been programmed to implement the method reported in this article. This subroutine as follows:

```
0001  FTN4,L
0002  C
0003        SUBROUTINE CRVFT(MODE,L0,X,Y,M0,N0,U,V,IERR)
0004  C
0005  C  SUBROUTINE DESCRIPTION
0006  C  ----------------------
0007  C
0008  C  THIS IS A CURVE FITTING SUBROUTINE ILLUSTRATING THE TECHNIQUE AS
0009  C  DESCRIBED IN THIS ARTICLE.
0010  C
0011  C  CALLING SEQUENCE
0012  C  ----------------
0013  C
0014  C   CALL CRVFT(MODE,L0,X,Y,M0,N0,U,V,IERR)
0015  C
0016  C    MODE ---> MODE OF THE CURVE:
0017  C
0018  C               1 - SINGLE-VALUED FUNCTION Y=Y(X)
0019  C               2 - SINGLE-VALUED FUNCTION X=X(Y)
0020  C               3 - MULTIPLE-VALUED FUNCTION, NONCLOSED CURVE
0021  C               4 - MULTIPLE-VALUED FUNCTION, CLOSED CURVE
0022  C
0023  C    L0 -----> NUMBER OF INPUT POINTS CONTAINED IN X & Y ARRAYS
0024  C
0025  C    X,Y ----> ARRAYS CONTAINING THE ABSCISSAS AND ORDINATES OF L INPUT
0026  C              POINTS
0027  C
0028  C    M0 -----> NUMBER OF DIVISIONS BETWEEN EACH PAIR OF INPUT POINTS
0029  C
0030  C    N0 -----> NUMBER OF OUTPUT POINTS FOR U & V ARRAYS
0031  C
0032  C    U,V ----> ARRAYS WHERE THE ABSCISSAS AND ORDINATES OF N OUTPUT
0033  C              POINTS ARE TO BE PLACED
0034  C
0035  C    IERR ---> ERROR RETURN: 0 - NUMBER OF INPUT POINTS AND NUMBER OF
0036  C                               DIVISIONS BETWEEN POINTS WITHIN RANGE.
0037  C                           -1 - NUMBER OF INPUT POINTS AND/OR NUMBER OF
0038  C                                DIVISIONS BETWEEN POINTS LESS THAN OR
0039  C                                EQUAL TO ZERO.
0040  C
0041  C
0042  C  EXTERNAL FUNCTIONS
0043  C  ------------------
0044  C
0045  C   SCR(SIJ,CIJ) = ABS(SIJ)-ABS(CIJ)*1.08E-8
0046  C
0047  C
0048        DIMENSION A(1),Y(1),U(1),V(1),A0(2),B0(2)
0049        EQUIVALENCE (A,A0(1)),(B,A0(2)),(C,B0(2)),(P0,X2),(Q0,Y2),(DX,A2),
0050       -            (DY,B2),(FLM,TS,Z),(JP,JS),(DU,DA,D,X1),(DV,DB,R,Y1),
0051       -            (S2,S20,A1),(S3,S03,B1),(P1,S12),(P2,C12),(P3,R12),
0052       -            (Q1,S13),(S2,C13),(Q3,R13),(B,B0(1))
0053  C
```

```
0054  C... PRELIMINARY PROCESSING ...
0055  C
0056        MD=MODE
0057        L=L0
0058        LPS=L0
0059        M=M0
0060        IF(L.LE.0.OR.M.LE.0) GO TO 900
0061        KP1=L*M+1
0062        IP=L+1
0063  C
0064        DO 10 JP=1,L
0065        KP1=KP1-M
0066        IP=IP-1
0067        U(KP1)=X(IP)
0068     10 V(KP1)=Y(IP)
0069  C
0070        KP2=1
0071        KP3=1
0072  C
0073        DO 20 I=2,L
0074        KP2=KP2+M
0075        IF(U(KP2).EQ.U(KP3).AND.V(KP2).EQ.V(KP3)) GO TO 20
0076        KP3=KP3+M
0077        U(KP3)=U(KP2)
0078     20 V(KP3)=V(KP2)
0079  C
0080        L=KP3/M+1
0081        N=KP3
0082        IF(N.EQ.1) GO TO 890
0083        IF(MD.NE.2) GO TO 50
0084  C
0085     30 DO 40 KP4=1,N,M
0086        TS=U(KP4)
0087        U(KP4)=V(KP4)
0088     40 V(KP4)=TS
0089  C
0090     50 MM1=M-1
0091        FLM=M
0092        DZ=1.0/FLM
0093        IF(L.EQ.2) GO TO 100
0094        LM1=L-1
0095        GO TO 200
0096  C
0097  C... SMOOTH CURVE FITTING FOR L=2 ...
0098  C
0099    100 DU=(U(N)-U(1))*DZ
0100        DV=(V(N)-V(1))*DZ
0101  C
0102        DO 110 KS=1,MM1
0103        U(KS+1)=U(KS)+DU
0104    110 V(KS+1)=V(KS)+DV
0105  C
0106        GO TO 800
0107  C
```

```
0229          B4=B3+DB
0230          A5=A4+DA
0231          B5=B4+DB
0232          GO TO 280
0233      480 A5=A3
0234          A4=A3
0235          B5=B3
0236          B4=B3
0237          GO TO 280
0238    C
0239    C... DETERMINATION OF THE DIRECTION ...
0240    C
0241      500 SGN=1.0
0242          IF(R23.LE.0.0) GO TO 550
0243          IF(R12.LE.0.0.AND.R34.LE.0.0) GO TO 580
0244          IF(R13.LE.0.0.AND.R24.LE.0.0) GO TO 580
0245          IF(R12.LE.0.0.OR.R24.LE.0.0) GO TO 560
0246          IF(R13.LE.0.0.OR.R34.LE.0.0) GO TO 570
0247          S2=S12*S24
0248          S3=S13*S34
0249          IF(S2*S3.LT.0.0) S3=-S3
0250          A=S2*A3*A3-S3*A2*A2
0251          B=S2*A3*B3-S3*A2*B2
0252          C=S2*B3*B3-S3*B2*B2
0253          D=S23*SQRT(S2*S3)
0254          IF(B*D.LT.0.0) D=-D
0255          B=B+D
0256          S20=A2*B0(1)-A0(1)*B2
0257          S03=A0(1)*B3-A3*B0(1)
0258          IF(S20*S03.LE.0.0) GO TO 510
0259          COS3=A0(1)
0260          SIN3=B0(1)
0261          GO TO 520
0262      510 S20=A2*B0(2)-A0(2)*B2
0263          COS3=A0(2)
0264          SIN3=B0(2)
0265      520 IF(S20*S23.GT.0.0) GO TO 590
0266          COS3=-COS3
0267          SIN3=-SIN3
0268          GO TO 590
0269      550 IF(C23.LT.0.0) SGN=-1.0
0270      560 COS3=A2
0271          SIN3=B2
0272          GO TO 590
0273      570 COS3=A3
0274          SIN3=B3
0275          GO TO 590
0276      580 COS3=A2+A3
0277          SIN3=B2+B3
0278      590 IF(MD.LE.2) GO TO LBL
0279          R=SQRT(COS3*COS3+SIN3*SIN3)
0280          COS3=COS3/R
0281          SIN3=SIN3/R
0282          GO TO LBL
0283    C
```

```
0284  C... INTERPOLATION IN A SECTION ...
0285  C
0286    600 KS0=(I-2)*M+1
0287        Z=0.0
0288        IF(MD.GT.2) GO TO 660
0289    610 KS1=KS0
0290        P1=DX
0291        Q1=P1*SIN2/COS2
0292        Q2=3.0*DY-2.0*Q1-P1*SIN3/COS3
0293        Q3=DY-Q1-Q2
0294  C
0295        DO 620 JS=1,MM1
0296        KS1=KS1+1
0297        Z=Z+DZ
0298        U(KS1)=P0+Z*P1
0299    620 V(KS1)=Q0+Z*(Q1+Z*(Q2+Z*Q3))
0300        GO TO 290
0301  C
0302    660 KS2=KS0
0303        R=SQRT(DX*DX+DY*DY)
0304        P1=R*COS2
0305        P2=3.0*DX-R*(2.0*COS2+COS3)
0306        P3=DX-P1-P2
0307        Q1=R*SIN2
0308        Q2=3.0*DY-R*(2.0*SIN2+SIN3)
0309        Q3=DY-Q1-Q2
0310  C
0311        DO 670 JS=1,MM1
0312        KS2=KS2+1
0313        Z=Z+DZ
0314        U(KS2)=P0+Z*(P1+Z*(P2+Z*P3))
0315    670 V(KS2)=Q0+Z*(Q1+Z*(Q2+Z*Q3))
0316        GO TO 290
0317  C
0318  C... NORMAL RETURN ...
0319  C
0320    800 IF(MD.NE.2) GO TO 890
0321  C
0322        DO 810 KR=1,N
0323        TS=U(KR)
0324        U(KR)=V(KR)
0325    810 V(KR)=TS
0326    890 L0=LPS
0327  C
0328        N0=N
0329        IERR=0
0330        RETURN
0331  C
0332  C... ERROR EXIT ...
0333  C
0334    900 ERROR=-1
0335  C
0336        END
0337        FUNCTION SCR(SIJ,CIJ)
0338  C
0339        SCR=ABS(SIJ)-ABS(CIJ)*1.08E-8
0340  C
0341        END
0342        END$
```

References

1. Hildebrand, F. B. (1956), Introduction to Numerical Analysis, ch. 2,3,4, and 9 (Mc Graw-Hill Co., New York, N.Y.).

2. Milne, W. E. (1949), Numerical Calculus, ch. III (Princeton University Press, Princeton, N.J.).

3. Ralston, A., and H. S. Wilf (1967), Mathematical Methods for Digital Computers, Vol. II, ch. 8 (John Wiley & Sons, New York, N.Y.).

MICROPROGRAMMING BASE SET SECRET

Joel Dubois/HP Grenoble

Have you ever thought, when you looked at the execution time for the 1000 E-Series computer instructions, "Why does a LDX instruction take longer than a LDA instruction"? If you examine the microcode of the base set you will certainly find the answer. However, you can fall into a trap when you follow the base set listing and then it is not easy to exit because some hardware information has been omitted from the Microprogramming Reference Manual (P/N 02109-90004). This article will illustrate.

Before I expose this situation let me give you some basic definitions of microprogramming.

As you know, to access the contents of a memory location, the address of this location must be put into the M-Register. A read operation is started, and at the end of the read cycle the data (memory location contents) is stored in the T-Register (data register). You also know that the A- and B- Registers can be accessed as Address 0 and 1 respectively. In a program, if you execute a "LDB 0" instruction, the computer must know that it is not the contents of memory location 0 but the contents of the A-Register to be loaded into the B-Register.

How does the processor "know" this?

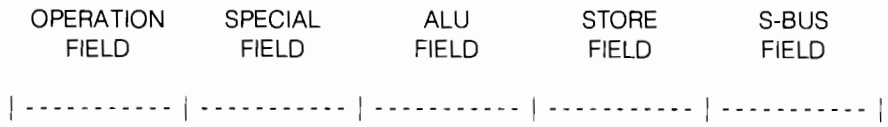
Every time you store an address into the M-Register, a check is made to see if this address is a "0" or a "1". If it is "0", a flag will be set which is called the "A-Addressable Flip-Flop" (AAF). If it is "1", another flag is set, the "B-Addressable Flip-Flop" (BAF). As you can see, only one of the two flip-flops can be set at any given time.

When a microprogrammer writes microcode for a LDB instruction, he does not know if later the instruction will be a LDB 100B or a LDB 0. He must take into consideration every case, and for this reason he will use a micro-order called "TAB". This is the abbreviation of the T-Register (data register) or A-Register or B-Register. According to the state of the A- and B- Addressable Flip-Flops, the computer knows which of these three registers it has to use. The decision is made using the following table:

ADDRESS STORED IN THE M-REGISTER	FLIP-FLOP STATES		REGISTER REFERENCED BY T A B
	AAF	BAF	
0	1	0	A
1	0	1	B
ANY OTHER VALUE	0	0	T

Now that you know how the computer interprets the TAB micro-order, you have to know where the programmer can use it.

One of the main differences between a routine written in assembler code and a microprogram is the fact that in assembler you can code only one instruction per line of code, but in a microprogram you have five fields. This allows execution of several operations, such as arithmetic, logical, shift-of-result, and start-of-read operation in the same microinstruction. The function of these fields is described below.



From the right, the fields are as follows:

THE S-BUS FIELD:

In the computer the main data path is called the S-Bus. When you want to put the contents of a register on this bus you just specify the name of this register in the S-Bus field.

THE STORE FIELD:

In this field you specify the register which is the destination for the data. The data can come from one of three sources, (1) the S-Bus, (2) the Arithmetic/Logic Unit, or (3) the rotate shifter. Which source is used depends on the destination specified.

THE ALU (ARITHMETIC/LOGIC UNIT) FIELD:

In this field you will specify which operation to perform in the ALU (e.g. ADD, SUB, XOR). This operation will be executed on the data on the S-Bus and the contents of another register called the L-Register (Latch Register).

SPECIAL FIELD:

In this field you may specify operations like COV (Clear OVerflow), L1 (shift left one place on the data leaving the ALU), RTN (return from a subroutine or to Control Store Address 0).

OPERATION FIELD:

In this field you will specify operations such as READ, WRITE, ARS (arithmetic shift), CRS (rotation).

The above description is specific to one type of microinstruction. The E-Series has four types. The format for a JUMP microinstruction is not the same. However, let us return to the discussion of the TAB micro-order.

The TAB can be used in two of these fields.

- In the S-Bus field, to get the result of a READ operation (you get the contents of the A- or the B-Register depending on whether the M-Register contains a zero or one).

- In the STORE field, when you want to store a value into the T-, the A-, or the B-Register, depending on the address in the M-Register.

Now that TAB is no longer mysterious, let us see how an instruction is executed in the computer.

The first step for the computer is to bring into the instruction register the binary code of the instruction which is in memory and then update the program counter. This sequence of events is called the FETCH routine. Microcode at Control Store Addresses 0 and 1 is the FETCH routine. This part of microcode will be executed for each assembler instruction, and at the end of this routine a branch will be performed to the appropriate microcode according to the contents of the instruction register. This is the execution routine which is specific to each assembler instruction.

Let us now review the processing of the instruction "LDA 100B". To add some challenge, suppose we have put this instruction in the A-Register itself. Thus,

ADDRESS	INSTRUCTION	OCTAL CODE
0=A REG	LDA 100B	060100

We will use the front panel to store this instruction in the A-Register, and then set the program counter "P" equal to zero. When PRESET and the RUN is pressed, the computer will execute an initialization routine in the base set. This routine is present at Control Store Address 325B, and stores the contents of the P-Register in the M-Register. A READ operation is initiated, the P-Register is incremented and control is transferred to the FETCH routine. When the FETCH routine commences, a READ operation is performed on the address which is in the M-Register and the P-Register is set to M+1. In this particular case, M=0 and P=1.

For the purists, when you press the RUN button, the FETCH routine is not started at Control Store Address 0, but rather at Control Store Address 1, because the operations performed at Address 0 have already been performed at Address 332B in the initialization routine. However, in this discussion, let us assume that the execution starts at Address 0, which is the common case in instruction execution.

The following microcode will be executed:

CONTROL STORE ADDRESS	OPERATION FIELD	SPECIAL FIELD	ALU FIELD	STORE FIELD	S-BUS FIELD	
0	READ	FTCH	PASS	IRCM	TAB	FETCH
1		JTAB	INC	PNM	P	ROUTINE
.						
.						
.						
47	READ		PASS	CAB	TAB	EXECUTION
50	RTN					ROUTINE FOR LDA

The explanation for this microcode is as follows:

ADDRESS 0:

- The contents of the A-Register (060100B) is put on the S-Bus. Since the initialization routine has stored a zero in the M-Register, the AAF (A-Addressable Flip-Flop) has been set.
- The value which is present on the S-Bus is stored in the instruction register because "IRCM" is present on the STORE field. This micro-order also means store the lower ten bits of the S-Bus in the M-Register (M=100B). At this point, the AAF and the BAF are cleared.
- The "FTCH" in the special field will execute some initializing operations.
- The "READ" in the Operation Field will start a READ operation on memory location 100B.

ADDRESS 1:

- The content of the P-Register is put on the S-Bus (P=1).
- "PNM" in the Store Field and "INC" in the ALU Field have the following meaning: store the contents of the S-Bus in the M-Register, increment the value through the ALU and store the result in the P-Register. In our example, a "1" is put on the S-Bus, and stored in the M-Register. The BAF will be set. A "2" will be stored in the P-Register.

- The "JTAB" in the Special Field will force a branch to Control Store Address 47 (the destination address of the jump is found according to the contents of Bit 15 through Bit 8 of the Instruction Register).

ADDRESS 47:

- On the S-Bus Field is a TAB micro-order. If we apply the rule of a "TAB" micro-order (see above table), the content of the B-Register should be put on the S-Bus because the BAF is set. You then store the S-Bus in the A-Register ("CAB" means conditional A or B according to Bit 11 of the Instruction Register). After this, a READ is started in Memory Location M=2. This read can be considered the beginning of the FETCH routine for the next instruction. Control is then transferred to Control Store Address 0 to execute the next instruction.

Now if you consider what happened in the computer, you have put the contents of the B-Register into the A-Register, but the instruction to execute was a "LDA 100B". Is it a problem in the firmware?

If you go into your computer room and try this example, it will work (you will find in the A-Register the contents of Memory Location 100B). Therefore, what is wrong with the preceding discussion? Actually, nothing in the philosophy, but I just failed to specify that at Control Store Address 1, when "1" is stored in the M-Register, the BAF is not set because the JTAB inhibits the clock for this flip-flop. This information appears to have been left out of the Microprogramming Reference Manual. In conclusion, at Control Store Address 47, it is the content of the T-Register (Data Register, which contains the contents of Memory Location 100B due to the read which had been started at Control Store Address 0) that is put on the S-Bus.

I took this example to show you a special situation. Normally when you go through a microprogram, you do not run into this kind of problem. Now you should be able to follow the execution for a expected results, look at the function of "JTAB", and your problems will be resolved by this very special micro-order.

Good Microprogramming!

Since all domestic training information is contained in a separate publication, we will no longer duplicate those schedules in the Communicator. The Computer Systems North American Customer Training Schedules (5953-0841) is published quarterly — June, September, December and March. This booklet is automatically sent to all Communicator subscribers on the Software Subscription Service and all HP training centers worldwide. It is our intention to continue to increase the usefulness of the CSG Schedule by including more information about prerequisites and the classes themselves in an attempt to make it a stand-alone document. International schedules, in-so-far as we receive them, will continue to appear in this publication.

INTERNATIONAL TRAINING CENTER ADDRESSES

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

AUSTRALIA
(Blackburn) B
CUSTOMER TRAINING CENTER
31-41 Joseph Street
Blackburn, Victoria, Australia
(Pymble) P
CUSTOMER TRAINING CENTER
31 Bridge Street
Pymble, New South Wales, Australia

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

ENGLAND
(Altrincham) A
Navigation Road
Altrincham
Cheshire WA14 1NU
(Winnersh) W
King Street Lane
Winnersh, Workingham
Berkshire RG11 5 AR
Tel: Workingham 784774
Cable: Hewpie London
Telex: 8471789

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

FRANCE
(Grenoble) G
5, avenue Raymond-Chanas
38320 Eybens
Tel: (76) 25-81-41
Telex: 980124
(Orsay) O
Quartier de Courtaboeuf
Boite Postale No. 6
F-91401-Orsay
Tel: (01) 907 7825

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

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

JAPAN
(Osaka) O
Chuo Building
5-4-20 Nishinakajima
Yodogawa-Ku, Osaki-shi
Osaka, 532 Japan
Tel: 06-304-6021
Telex: 523-3624 YHP OSA
(Tokyo) T
2205 Takaido Higashi 3-chome
Suginami-Ku, Tokyo 168
Tel: 03-33-8111
Telex: 232-2024 YHP MARKET TOK

NETHERLANDS
(Amsterdam)
Van Heuven Goedhartlaan 121
Amstelveen 1134
Netherlands
Tel: 02 672 22 40

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

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

For course prerequisites and registration information contact one of the HP training centers listed above.

INTERNATIONAL TRAINING CENTER SCHEDULE

	AUSTRIA	AUSTRALIA	BELGIUM	ENGLAND	FINLAND	FRANCE	GERMANY	ITALY	JAPAN	NETHERLANDS	SPAIN	SWEDEN
22941A 21MX/XE Maint. 5 days/$500						Jul 02 (G) Oct 22 (G)						
22943A 7970B/E Maint. 5 days/$500						Jul 23 (G)						
22945A 7905/06 Maint. 5 days/$500						Jul 09 (G) Oct 29 (G)						
22951B Intro to HP Mini's 4 days/$400	Sep 03						Jul 16 Sep 24			Aug 27		Oct 08
22952B 1000 ASMB 5 days/$500	Sep 24	Jun 11 (P) Jul 30 (B) Oct 08 (P) Nov 12 (B)					Jun 25 Sep 03		Jun 11 (T)	Jun 11 Oct 01	Oct 22	Sep 03 Nov 12
22961B DS/1000 Theory of Operation 4 days/$500							Oct 01			Sep 24		
22962B DS/1000 to HP 3000 Theory of Operation 1 day/$100							Oct 05			Sep 28		
22965B RTE-II/III 10 days/$1000						Jun 11 (O)	Jun 18					
22965B-H01 FORTRAN IV 5 days							Jul 23 Oct 08				Oct 15	
22977 IMAGE 5 days/$500		Jul 09 (P) Aug 20 (B) Nov 12 (P) Dec 03 (B)				Jun 25 (O) Sep 24 (O)	Jul 09		Jun 25 (T)	Jul 16 Oct 08	Nov 12	
22980C HP-IB Minicomputer Environment 4 days/$400							Jul 23			Aug 20		
22984A 7920 Maint. 5 days/$5000						Jul 16 (G)						
22985A RTE-M 5 days/$500							Jun 18 Sep 10					
22987A DS/1000 User's Course 5 days/$500		Oct 22 (P)					Sep 17		Jun 18 (T)	Sep 17		
22990A RTE Driver Writing 3 days/$300		Jun 18 (P) Aug 06 (B) Oct 01 (P) Nov 19 (B)										
22991A HP 1000 DISC RTE-IVA 10 days/$1000	Sep 10	Jul 09 (B) Sep 17 (P) Oct 22 (B)	Oct 01				Jul 09 Jul 30 Aug 20 Sep 10 Oct 01 Oct 15		Jun 04 (O)	Jun 25 Sep 03 Oct 15	Oct 29	Sep 10 Oct 15 Nov 19
22992A HP 1000 Memory RTE 10 days/$1000												
40270A Intro to HP Computers 5 days						Jul 02 (G)						
91302A 2645 Maint. 3 days/$300												

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				_____

☐ **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					_____

TOTAL ORDER DOLLAR AMOUNT _____

☐ **SERVICE CONTRACT CUSTOMERS**

You will receive one copy of the COMMUNICATOR 1000, 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 _____

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	_____	$48.00	_____	
	(if quantity is greater than 1 discount is 40%)			_____	
	TOTAL DOLLARS for 5951-6111				_____

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 dllars 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	_____	_____	10.00	_____	
		_____	_____	10.00	_____	
		_____	_____	10.00	_____	
	TOTAL DOLLARS					_____

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

   HEWLETT-PACKARD
   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.

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. 5/79

Part No. 5951-6111