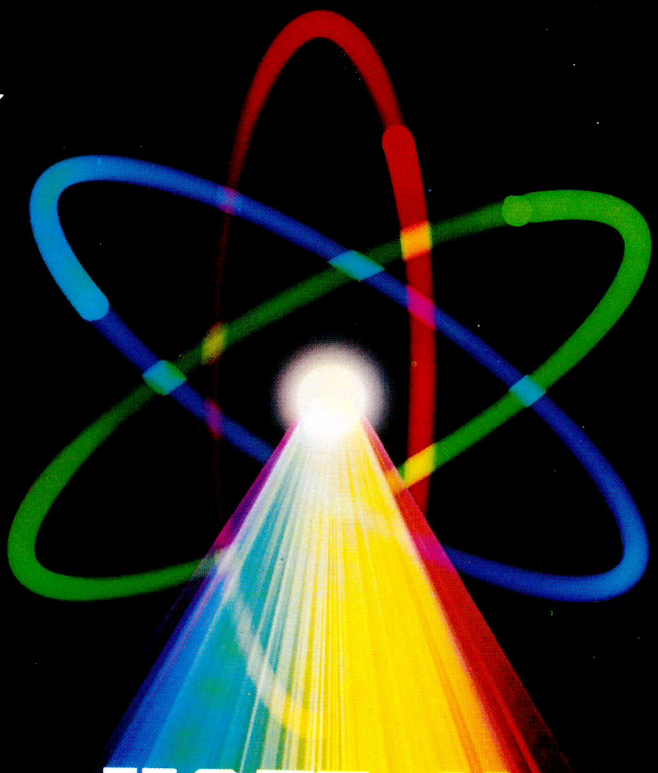
RSTS PROFESSIONAL

Volume 5, Number 3

June 1983

$10.00/issue, $35.00/year

NOW AVAILABLE FOR VAX



USER-11: POWERFULLY PRODUCTIVE.

People productivity. It's more important than ever. And a good database system can mean *real* productivity.

USER-11 is a high-performance database system.

It is a fact: Software designed with USER-11 is built more quickly, operates more reliably, and performs better than other software techniques.

USER-11 is unique. It's easy to install. Easy to learn. And easy to apply. Adaptive tools and a standard approach ensure that maintenance is easier than ever.

A key to USER-11's success is its powerful, dictionary-based modules. Software developers simply describe and assemble these modules to create custom business packages—at an unprecedented rate.

Naturally USER-11 is supported with excellent documentation and a variety of training options for beginner to expert. Our commitment is to your complete satisfaction.

Whether you are a software provider or a software user, we guarantee you will be delighted.

Ask us about USER-11 and our family of business software products, or better yet, ask a *productive* USER!

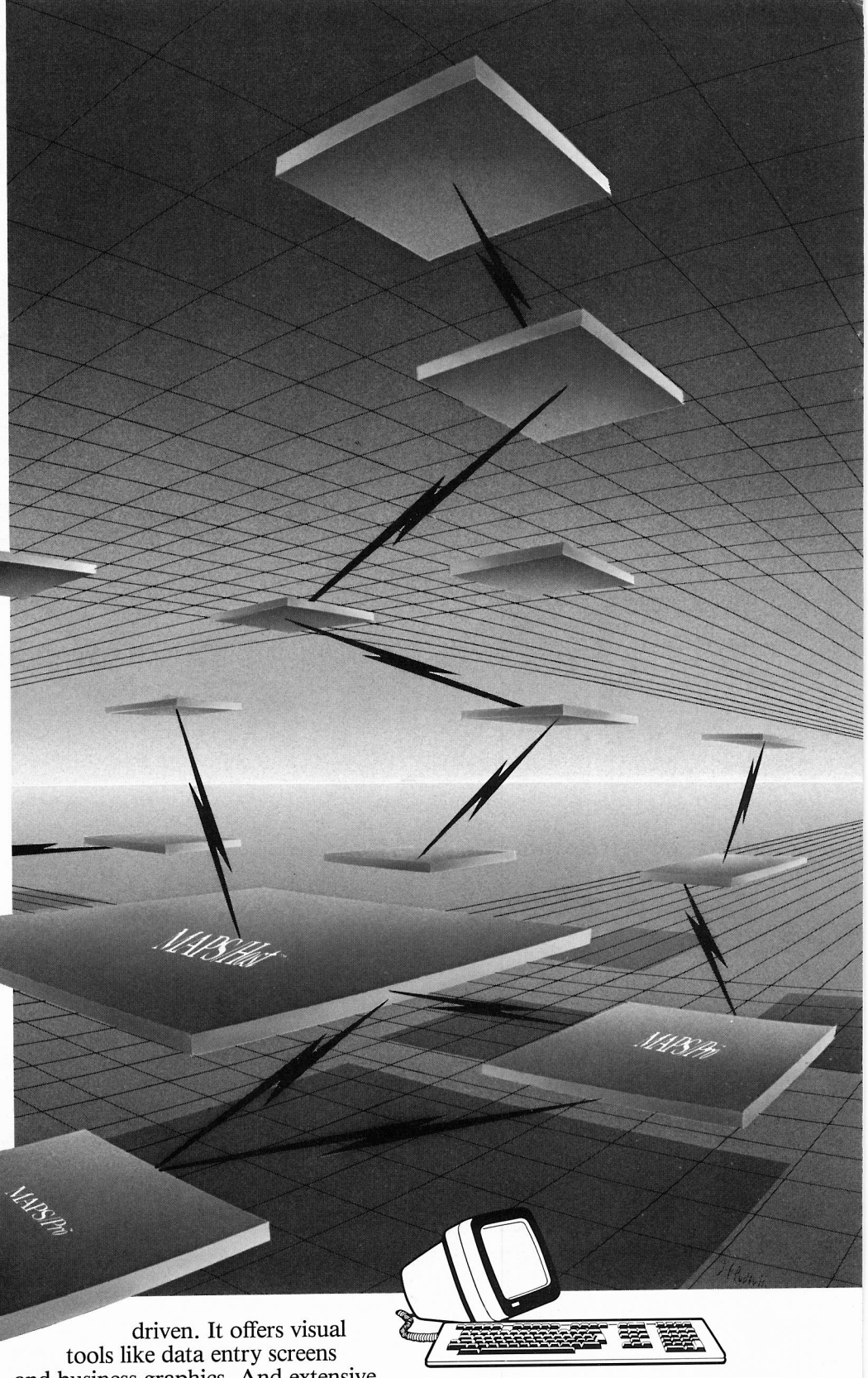
NCCS USER 11

North County Computer Services, Inc.
2235 Meyers Ave.,
Escondido, California 92025
(619) 745-6006, Telex: 182773

***USER-11 is currently available for DEC computers using the RSTS and VMS operating systems.**

CIRCLE 30 ON READER CARD

Ross Systems has perfected the Art of Distributed Financial Modeling



MAPS/*PRO*™ *is the most effective method for distributing financial modeling solutions throughout the levels of your company.*

Designed exclusively for the DEC* Professional 350 microcomputer, MAPS/PRO, together with MAPS/Host™ provides company-wide solutions to a wide range of common financial planning problems. Like inter-departmental budgeting. Divisional roll-ups. Even pro forma forecasting.

It's the first financial modeling language that uses fully-compatible mini and microcomputer data and model formats. Equally important, MAPS/PRO offers you instant availability. So you don't have to wait in line for the corporate mainframe.

MAPS common financial planning language and sharable financial data base lets you work independently at your own micro. So you can develop models and data on your Pro 350 computer. Easily transfer them to your VAX* or PDP-11* minicomputer. And run them under MAPS/Host. Without costly and time-consuming changes. You are productive . . . right away.

MAPS/PRO is easy to use, too. It's completely interactive and menu-driven. It offers visual tools like data entry screens and business graphics. And extensive on-line HELP is available whenever you need it.

Isn't it time you took a look at the state-of-the-art in distributed financial planning software? It's easy. Just call Ross Systems toll free at (800) 547-1000 (in California, call (415) 856-1100).

**MAPS, MAPS/PRO and MAPS/Host are registered trademarks of Ross Systems, Incorporated.*

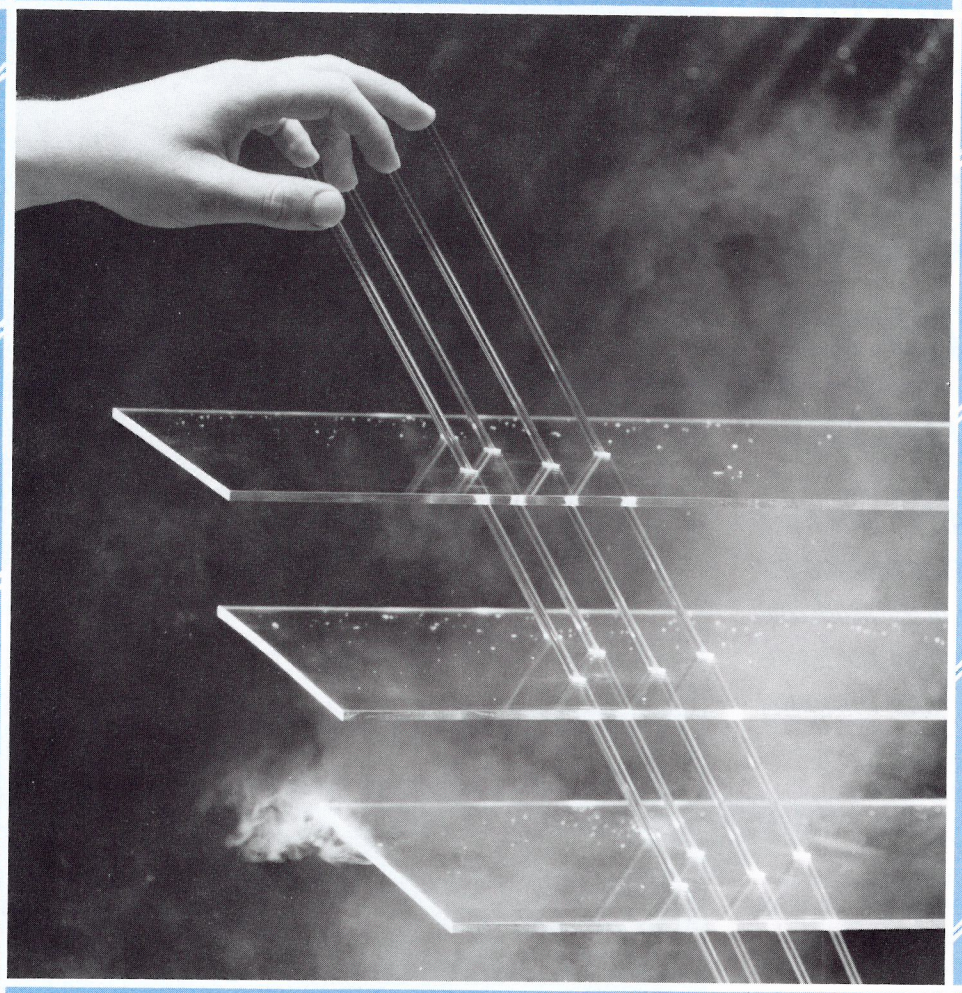
**DEC, VAX, PDP, Pro, and Professional are registered trademarks of Digital Equipment Corporation.*

ROSS SYSTEMS

1860 Embarcadero Road
Palo Alto, CA 94303
Regional Offices in San Francisco, New York, Dallas, Los Angeles.

CIRCLE 1 ON READER CARD

Summoning Solutions DIGICALC v.1.2 T.M.



- GOAL SEEKING
- ARCHITECTURE ALLOWS DIVISION CONSOLIDATION
- TEXT OR NUMBER SORTING
- MULTIPLE USER CAPABILITY
- ADVANCED CLASSROOM TRAINING

DIGICALC 1.2 brings new dimensions to the DEC system user. This truly advanced version has immense capacity for spreadsheets that generate budgeting, financial modeling, project management and many types of analyses. This most flexible software is the perfect combination of procedural and non-procedural language. Once learned, the scientific concept of pattern recognition that has been designed into the keypad aids recall for professionals and executives with little computer experience. No other program for DEC equipment has such extensive HELP built in. The user is in command and can define functions, consolidate multiple spreadsheets . . . even integrate with existing systems on a multi-user level.

With over 400 copies installed and 8000 users world wide, a simple call to WHY Systems will give you finger tip access to the more perfect power of DIGICALC™ 1.2. Runs on VAX/VMS,* PDP-11*and RSTS/E.*

WHY

Seeking Solutions . . . Mankind Asks WHY

16902 Redmond Way, Redmond, WA 98052 U.S.A. (206) 881-2331

*Registered trademarks of Digital Equipment Corporation

- BUILT IN TRAINING
- EXTERNAL FILE INTERFACE
- UP TO 17 DIGITS OF NUMERICAL PRECISION
- BOARDROOM QUALITY REPORTS
- THREE-DIMENSIONAL CONSOLIDATION

CIRCLE 72 ON READER CARD

Contents

Coming

- Make your PDP/34 Work Harder
- CASTAT, An Optimization Tool
- Act 4 — A 4th Generation System Development
- Tape Copy
- ISCAN.BAS
- A Macro-Sub Program for Determining Time Between Two Dates
- On-2-Off/TU-58 Drives
- Patching PIP For Protection
- The Effective Use of the VT-100 Printer Port
- Basic Plus Techniques for Table Organization & Look up
- Multilingual Applications Development
- DYNPRI
- More . . .

The RSTS Professional Magazine, June 1, 1983, Vol.5, No.3. ISSN 0745-2888 is published bi-monthly by M Systems, Inc., 161 E. Hunting Park Avenue, Philadelphia, PA 19124. Subscriptions: single copy price 10^{00}, 35^{00} per year; $50 Canada and 1st class. All other countries, air mail, $60 US. Second Class postage paid at Philadelphia, PA. POSTMASTER: Send all correspondence and address changes to: RSTS Professional, P.O. Box 361, Ft. Washington, Pa. 19034-0361, telephone (215) 542-7008.

From the publishers . . .

RSTS/VAX PROFESSIONAL??

Carl B. Marbach & Dave Mallery

As of the next issue, we have decided to change our flagship publication's identity. The August issue will be officially titled "RSTS/VAX PROFESSIONAL." It will have more articles of interest to VAX people and feature several new authors who will continue to improve our editorial content.

We want to take this opportunity to explain our choice to you, our loyal RSTS crew. Like any business decision, this change of name is based on a change of realities. What realities?

Reality #1

The population is shifting. Growing, but shifting. Not exactly lemmings marching to the sea, but if you go to DEC with $200,000 and tell them you want a computer, you get a VAX 11/750 every time.

Reality #2

Your PDP-11 is getting older. There is no 11/70 replacement (yet). You can't wait forever. Service and electricity are only going up. The J-11(11/70 chip) seems always just over the horizon and probably will emerge as a board-level machine with very different characteristics, more in keeping with DEC's overall marketing strategy. In other words, don't expect any great new exciting PDP-11s in the near future; They will slowly evolve to be smaller, faster and cheaper, but there will not be the PDP-11/90.

Reality #3

More and more of you, our readers, are moving on to the VAX/VMS and frankly, we hate to see you go. Maybe even worse, there are VAX people coming into the world without ever having known the RSTS PROFESSIONAL.

Reality #4

We thought of calling ourselves THE INTERACTIVE PROFESSIONAL, the magazine for Digital technical computer users. A technical journal is what we set out to be; and we want to be a technical journal for all of us who know and love our DEC computers. We realize that a large number of new DEC users are now VAX oriented and we need to include them in not only our readership, but to count the among our contributors.

This is not a magazine for a small clique of gurus. It can and will serve a much larger audience. The more of us there are, the more we can share, and the greater our collective clout will be.

Five years from now, RSTS will be thriving — in numbers that are unimaginable right now. However, it will be the mainstream O/S of the 'new' PDP-11 line:

The PRO 3xx
The MICRO-11
The (MACRO-11??)
(sorry but your 44 is obsolete . . .)

Most of the 'hosts' in this brave new, distributed world will be VAX. RSTS will live happily in the box under your desk.

Most of us, willing or not, will be faced with VAX in our everyday work. Now is a great time to start; sharing if you already know, learning if you are new, and growing as we all must to keep up with the exciting world we work and live in.

So come with us. Start to give VAX a heart! ♥

VAX RSTS PROFESSIONAL

RSTS PROFESSIONAL

Publishers: R.D. Mallery, Carl B. Marbach

Managing Editor: James L. Trichon

Director of Advertising: Helen B. Marbach

Business Manager: Peg Leiby

Assistant to the Editor: Hope Makransky

Editorial Assistant: Linda DiBiasio

Production Manager: Georgia Conrad

Subscription Fulfillment:
Kathi B. Campione, Claire Hollister
Steven Barsh, Margie Pitrone
Connie Mahon

United Kingdom Representative:
Pauline Noakes
RTZ Computer Services Ltd., P.O. Box 19
1 Redcliff Street, Bristol, BS99-7JS
Phone: Bristol 24181

Mid-Atlantic Representative:
Ed Shaud
P.O. Box 187, Radnor, PA 19087
Phone: 215/688-7233

N.Y. & New England Representative:
Jack Garland
P.O. Box 314 S.H.S., Duxbury, MA 02332
Phone: 617/934-6464

Contributors:
R. David Broom, William R. Davy,
Peter Dick, Steven L. Edwards,
Alexander Ehrlich, Mark Gilmore,
Philip Hunt, Ken Isaacs,
Michael H. Koplitz, Bob Meyer,
Maury Pepper, Terry G. Ridgers,
Greg Wenzel.

Cartoons: Frank Baginski, Emily Dahlstrom, Brian Hansen

Design, Typesetting & Layout: Grossman Graphics

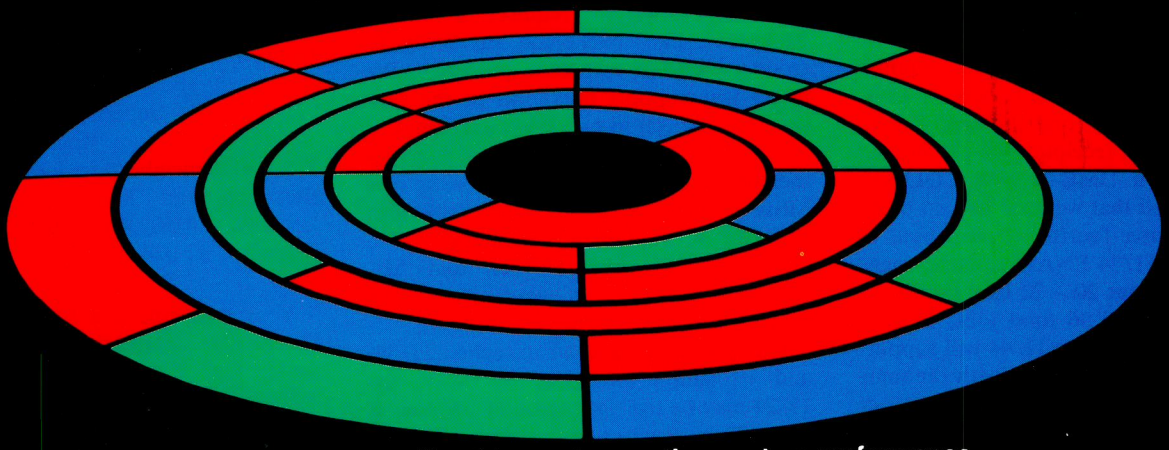
Printing & Binding: Schneider Litho Co., Inc.

Cover Photo: David Sheppard
Cover Model: Margie Pitrone

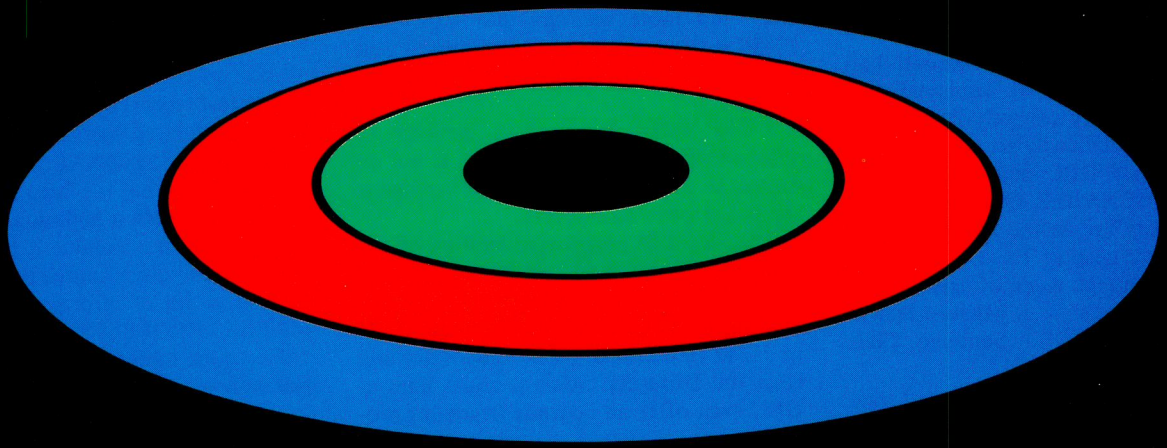
Editorial Information: We will consider for publication all submitted manuscripts and photographs, and welcome your articles, photographs and suggestions. All material will be treated with care, although we cannot be responsible for loss or damage. (Any payment for use of material will be made only upon publication.)

OW FOR V8.0

WHAT YOU DON'T KNOW ABOUT YOUR DISKS IS COSTING YOU MONEY



If your disk looks like this, you're wasting system performance.



If your disk looks like this, you're using DISKIT.

When the job you're running requires reading the "red" file, it naturally happens faster on a well-ordered disk. Disks become "fragmented" as you use your computer. The system slows down. And that costs you money.

Now, you can restructure your disks and get back that lost performance (up to 50%) without spending a dime on new hardware. DISKIT is the original software system that makes this possible.

But don't confuse DISKIT with other system utilities, DISKIT is a complete "software tool kit" that optimizes your RSTS/E system.

DISKIT is:

- DSU — The utility which restructures the information on your disk, making data fast and easy to access.
- DIR — The incredible directory tool that finds files at the rate of 400 per second.
- RDR — Reorders disk directories 30 times faster than ever before possible.
- OPEN — Displays complete job statistics and file activity so you can see what your system is doing.
- DUS — The set of CALLable subroutines which pre-extend file directories, reducing fragmentation.

In today's tight economy, it's more important than ever to get the most out of your hardware investment. Call or write today and start getting your money's worth from your computer.

Software Techniques Incorporated

5242 Katella Avenue
Los Alamitos, CA 90720
United States
Phone: [714] 995-0533

287 London Road
Newbury, Berkshire RG13 2QJ
United Kingdom
Phone: 44 [0] 635-30840

CIRCLE 65 ON READER CARD

LETTERS to the RSTS Pro ...

Send letters to: Letters to the RSTS Pro, P.O. Box 361, Ft. Washington, PA 19034-0361.

I can't let the bad words on ABLE computer and ENABLE pass without responding. We installed ENABLE here in Chile last year, with only one problem due to ABLE — the first receipt of the patches did not support our DMC-11 with COLINK. Once they fixed that we have had no problems. With three-fourths of megabyte of memory, our 11/34 ENABLEd with cache will support about 20 - 22 user terminals, some printers, and 36 total jobs. In contrast, our one megabyte 11/44 will support about 34 - 36 user terminals with the same job mix. More memory probably will not help the 34 since it appears CPU bound.

In addition, our ABLE DH/DMs have been faultless, and we are delighted. We have nothing but good things to say about ABLE, and their relations with their customers, and with their products.

However, I can not say enough bad things about the National Semiconductor NS-11 boards — they are both unreliable in themselves, and obviously are not true DEC substitutes, since DEC boards work perfectly well with ENABLE. One of our NS-11 boards can only live on the 18-bit bus, and the other won't keep its DC-DC converter working long enough to be of use.

J. Michael Hewitt
Santiago, Chile

* * *

Thank you for the fine magazine; I look forward to every issue. I find I use my RSTS PROFESSIONAL "library" just as much as I use my DEC RSTS/E manuals. I particularly enjoyed Scott Banks' articles on disk directories and I have applied his ideas to create a program that may be of interest to other RSTS DIBOL users.

RSTS is alive and well in SKI TOWN, USA!!!

David S. Williams Jr.
Director of Computer Services
Yampa Valley Electric Assoc.
Steamboat Springs, CO

Ed. Note: See Dave's article, ISCAN.BAS in the next issue.

* * *

The February 1983 issue of the *RSTS Professional* carried both an article and a letter from Alan Woloshin discussing Digital's single-board 11/44. Both are very wrong.

Here are the facts:

- Digital currently markets two Unibus PDP-11 processors: The 11/24 and the 11/44. The 11/24 is a single hex-board processor and is pinned somewhat like the 11/04 and 11/34. This is probably where the confusion arose. The 11/44 is a multi-board processor, consisting of between 4 1/3 and 8-hex modules, depending on options.
- Each processor requires a unique backplane. Each backplane was newly introduced with its respective processor. We in no way support operation of the processor module(s) in previous backplanes.
- The 11/24 uses existing MS11-L parity memory (albeit in a new mode) or the new MS11-P 1Mb ECC memory. The 11/44 uses MS11-M or MS11-P ECC memory.

From these you can see that Mr. Woloshin's suggestion won't work. We suggest instead that:

- If you now have an uncached 11/34, and are simply swapping too much, the 11/24 may be the right solution for you. It offers 22-bit (4Mb) addressing to solve your swapping problems but uses a slightly slower CPU.
- If you now have an uncached 11/34, and need more memory and CPU, the 11/44 is the right solution for you. It also offers 22-bit (4Mb) addressing and a faster CPU.
- The speed comparison between a chached 11/34A and the 11/44 is not so dramatic. Here, the principal advantage is the larger physical memory.

The only DEC-supported method of upgrade is (CPU) box replacement. RSTS users can just move all your I/O from your existing 11/34 CPU box to your new 11/24 or 11/44 CPU box or cabinet, boot up, and reset the DEfaults. Some product lines at DEC even offer an upgrade (trade-in) program.

Atlant G. Schmidt, Principal Engineer
PDP-11 Systems Engineering
Maynard, MA

* * *

I applaud your editoral, *The Right Tool* (April, 1983), in which you point out how RSTS continues to lag behind other DEC operating systems. DEC, although pledging their support of the RSTS community, continues to ignore our needs or place them behind the needs of the VAX and RSX communities. Is it that we don't have enought clout?

The RSTS community must be able to present its needs and problems to DEC and expect some action on their part. I have heard estimates which claim that there are 10,000 RSTS sites across the U.S.A. (and this may be conservative). It would seem to me that 10,000 sites should be enough clout to convince DEC that RSTS is a product which must remain state-of-the-art and have their full support. What we need is a united voice.

The Independent RSTS Users Society (IRUS) can be effective in using clout to influence DEC. In addition, it can assist member sites in overcoming some of the deficiencies of RSTS by providing a means of exchanging information and experiences with other RSTS sites.

If indeed there are 10,000 sites out there, then there should be 10,000 members of IRUS. I invite each of your readers to join.

Carlos Flores, IRUS Chairman
Superior Computer Systems
Rockville Center, NY

P.S. IRUS membership dues are $75.00 per site, per year. To join, contact Joyce Leonard, IRUS, Inc., 3657 Post Road, Warwick, RI 02886.

* * *

Your magazine is a fine instrument for the transfer of expertise, and we thank you for it.

Do you, your associates, or any of your readers know of a DIBOL program generator? I would be grateful for any informed response, yea or nea.

Richard L. Logan, President
Access Information Systems, Inc.
Ashland, MA

* * *

I've just received your inquiry as to why I haven't renewed my subscription to the RSTS PROFESSIONAL.

Let me say that it's been my pleasure to subscribe to such a high-quality and *useful* publication since Volume 1, Issue 1. I don't know of any other source that can give me such a wealth of information of RSTS as the RSTS PROFESSIONAL.

However, things do change, and my office now runs 4 VAX 11/780's in addition to one PDP-11/70 (and RSTS). Furthermore, the PDP will be phased out by late summer, so in a nutshell, no more RSTS. While I find *The VAX Scene* to be an informative section, I hope that you might find it possible in the very near future to expand it into a VAX PROFESSIONAL that will rival the RSTS PROFESSIONAL in scope and quality.

Les Hino
University of Hawaii
Honolulu, HI

Ed. Note: Please see our editorial on page 4. Welcome back!

* * *

I am writing in response to the fellow who is looking for an escape from XQWIK. There was a contribution to the Spring 1982 RSTS symposium tape called CSORT which may be of help.

CSORT is a high performance replacement for XQWIK. It supports the same data formats and creates the same key files. However, it is ten times faster.

Ken Harris, software consultant
Greendale, WI

♥

EMULEX TALKS DEC 4

IS YOUR VAX SUFFERING FROM A TERMINAL CONDITION???

How would you like to double (repeat, double) the throughput of your terminal-bound VAX-11/750 or 780—for less than $6,000? DEC's DMF-32 modules have already slashed the data-communications overhead of "small" VAX-11/730's. Now Emulex is extending DMF concepts to the whole VAX family. With bonus benefits. Emulex's single-board CS21/F multiplexer gives you 16 asynchronous lines (instead of 8), with modem control for remote terminals on all 16 lines (instead of just 2). Emulex CS11/F expands this to 48 lines—again on a single board.

THE BEST OF DH AND DZ...

Dynamic selection of the most efficient mode of operation, DMA or programmed-I/O, by the DMF driver significantly reduces DZ11-type interrupts.

Emulex's DMF emulations provide DMA (DH11) operation when queued output data is greater than 64 characters in length, as in word processing, screen editing and graphics.

Because there is CPU overhead involved in setting up a DMA transfer, interrupt (DZ11) operations are more efficient for lesser amounts of data, such as character echo and data entry. The DMF driver can transfer "bursts" of up to 32 characters to the 32 character output silo on each line, under a single interrupt operation. Result: Emulex DMF emulations provide the most cost effective VAX performance enhancement currently available!!!

SOON (STAY TUNED)...

And that's just for starters. Emulex will soon offer two new DMF-type multiplexers that will allow you to expand your system, in DMF mode, to include statistical concentration (an Emulex exclusive). Result: Single-board modules that can support up to 128 or 256 local or remote terminals. Become a Believer. Write for a more extensive explanation of Emulex data communications and the new DMF emulations.

SYSTEMS 10 AND 20, TOO...

The word is out. Emulex communications technology has been extended to DEC System 10 and System 20 mainframes. How? Users are plugging Emulex modules into the PDP-11/34's or 40's that serve as front-ends for nearly every large DEC installation. No change in front-end or mainframe software required.

FROM THE EMULEX FILE...

Read about all these new Emulex products and a lot more in the new Emulex Controller Handbook now coming off the press. Write or call for your free copy.

3545 Harbor Blvd., P.O. Box 6725,
Costa Mesa, California 92626,
Toll Free (800) 854-7112, In Calif. (714) 662-5600.

CIRCLE 58 ON READER CARD

RSTS/E DISK OPTIMIZATION IN A MULTI-USER ENVIRONMENT

By William R. Davy, System Performance House, Inc.

ABSTRACT

A great deal of existing literature addresses the important matter of RSTS/E disk optimization. This article extends beyond the conventional wisdom to describe previously unpublished optimizations available for multi-user RSTS/E systems. Included are a review of common disk optimization practices, some observations about the multi-user RSTS/E environment, and how these two interact.

Most RSTS/E users are painfully aware that their systems tend to be disk bound. They perform far more disk accesses than are needed for mere data retrieval, program loading and swapping. Furthermore, disk seek time for these operations and others is longer than necessary. To minimize these problems, users are generally limited to the following methods.

- Center and pre-extend the UFDs. Some shops also pre-extend the MFD contiguously starting at device cluster one.
- Map files contiguously and give them the contiguous attribute where appropriate.
- Increase file clustersize so that the file is mapped in seven clusters or less, up to the maximum clustersize of 256.
- Center the swap files, run-time system files, etc. Tedious manual procedures generally limit these efforts to the few files which are perceived to be most used.
- Increase the pack cluster size to decrease FIP overhead.
- Run REORDR frequently.
- Use the "new files first" attribute on the disks.
- Allocate some free space near the center of the disk.

Two major focuses of this article will be to correct the misconceptions associated with the above steps and to describe other significant optimizations. It is assumed that the user is familiar with the RSTS/E file structure. There have been excellent descriptions by Mike Mayfield, Scott Banks, et al.

THE RSTS/E ENVIRONMENT

— Software Characteristics

RSTS/E systems are by definition multi-user systems: their performance problems arise under multi-user conditions. Consequently, our optimization efforts will focus on these.

The fundamental consideration of RSTS/E disk optimization is that consecutive disk accesses to the same file, UFD or MFD are statistically rare in a multi-user RSTS/E system.

For example, take a system whose file clustersize has been optimized and whose directories have been recently REORDRed. Consider what would happen if there were a number of jobs performing heavy I/O to disk files and a few jobs excercising FIP through directory accesses, etc. (i.e., a typical RSTS/E system). We would find enough idle time so that each job would receive the CPU time to queue its next disk access well before the system could process the other pending requests. For most of the time, each job would have a pending request. No given job would be able to receive two consecutive disk accesses, the RSTS/E "fairness" algorithm notwithstanding. If each job accessed a different set of files, no file would ever receive two consecutive accesses.

What about accesses to MFDs and UFDs? It is fairly well known that FIP is single-threaded. That is, it will process any operation to completion before starting another. This guarantees that two jobs will never perform FIP operations simultaneously. Although some FIP operations require multiple UFD accesses (e.g., a file lookup in a large directory), there are other jobs which do file accesses without FIP. There will be file accesses in between the UFD accesses.

Furthermore, FIP always accesses exactly one MFD/UFD block at a time. Monitor statistics show that the number of directory accesses always equals the number of directory blocks transferred. The value of this observation will become apparent when we discuss UFD optimizations.

— Hardware Characteristics

DEC and third-party vendors offer a number of disk drives with different sizes and speeds, but they all have some common characteristics.

First, total time for completion of a disk read or write operation is equal to: SEEK TIME + ROTATIONAL LATENCY TIME + TRANSFER TIME. Seek time across just one track is significant when compared to either rotational latency or transfer time. Seek time increases less than linearly as the number of tracks increases.

Rotational latency time is the time it takes for the disk to rotate the transferable data under the head(s) after the seek has been performed. The fundamental consideration discussed above shows us that it is nearly impossible to predetermine the rotational state of the disk before any given operation. Therefore, except where special circumstances warrant otherwise, it is accurate to assume an average latency time equal to one-half the maximum latency time.

From a statistical standpoint, transfer time is small compared to the other two components. Consequently, our major hope of speeding up accesses lies in reducing seek time. This is accomplished by accessing blocks on the disk which are "close" together. For now, it will suffice to say that blocks which are close together on the disk have block numbers which are near each other, and vice versa.

THE CONVENTIONAL WISDOM RE-EXAMINED

Now we are ready to re-examine all of the popular disk optimization methods, paying particular attention to their effect on multi-user systems.

DOPTER

The Time & Money Saver

1. **PERFORMANCE**

DOPTER is a RSTS/E disk optimization program which provides dramatic improvements in response time and up to 50% more throughput. DOPTER produces a better organized disk than any other product!

2. **EASE OF USE**

The Disk Analyzer and other automatic functions require almost no intervention by the user. Yet features not offered by anyone else are standard with DOPTER.

3. **PRICE**

At $750 for the first CPU, DOPTER is priced so you can't afford to have fragmented disks and a slow system. And with each license we include a 30 day unconditional money-back guarantee.

If you would like more information on how you can increase the performance of your RSTS/E system with DOPTER, phone or write today.

SPH

System Performance House, Inc.

51 North High Street · Columbus, Ohio 43215 · 614-464-4411

A) Determining the disk center

In many disk optimizing schemes, the UFDs, free space and certain files are centered. Some programs calculate the center of the disk to be the median numbered block. Others recognize that the entire disk may not be allocated, so that the center is better considered to be closer to the beginning of the disk, say perhaps one-third of the way from the beginning to the end.

These schemes are feeble attempts to guess the optimal "center" of the disk. A much better position can be calculated. Simply stated, the center would be the block number equal to half of the space needed for all of the files on the disk, plus some free space and the space needed for the UFDs and the MFD. An improvement upon this algorithm would be to subtract the size of all of the files not used during time sharing. (They would be placed at the end of the disk where they would not get in the way.) The other files, the UFDs, the free space and the MFD would be placed on the disk, starting at the beginning. The calculated center would then be at the center of the active files. In this article, the term "center" refers to this.

B) Pre-extending the MFD

Some installations pre-extend the MFD. Presumably, this is done to make it contiguous. In a single-user system, this can speed up MFD searches, but in a multi-user system, it practically guarantees that MFD seeks will be as slow as possible. The MFD would be entirely contained at the edge of the disk. As the fundamental consideration shows, we are unlikely to get two consecutive accesses to the MFD. A much better strategy for optimizing the MFD is described later.

C) Placing and pre-extending the UFDs.

The fundamental consideration shows that the main reason to pre-extend a UFD is not to make it contiguous, but to control its general position. The proper position for UFDs is near the center. The strategy of placing UFDs near their associated files may make some sense in a single-user system, but it is folly in a multi-user system. It will guarantee that UFDs — the most heavily accessed blocks on the disk — will be scattered as far apart as possible. It will also guarantee that if the most used files are in different accounts, they will also be scattered as far apart as possible, maximizing seek time.

The second question concerning UFDs is how much should they be extended. One strategy is to extend them to their maximum size, 112 blocks. On a system with 100 accounts, this requires 11,200 blocks of prime space on the disk, most of which will never be used.

Another strategy is to use the minimum amount necessary to hold the current directory information. This strategy is poor on two accounts. First, many systems add new files to their accounts without first deleting others. If the UFD is full, it will be extended, not necessarily in a central location. Second, when files are deleted and recreated as they are by editors, compilers, etc. they are often recreated with smaller clustersize which requires more UFD space. If the UFD was created just large enough to hold its previous contents, it will have to be extended to hold the expanded mapping information. So it is clear that some scratch space should be left in the UFD. The UFD optimization section will have more on this subject.

D) Using the optimum file clustersize

Optimum file clustersize is generally considered to be the smallest clustersize which will map the file in seven clusters or less. If the file is larger than 256*7 blocks, the optimum clustersize would be 256, the maximum clustersize. The reason for this is that the "retrieval blockettes" in the UFDs each hold pointers to seven clusters. If the file has seven clusters or less, then the minimum number of blockettes is needed to map the file.

Optimum clustersize helps performance two ways. First, since only one retrieval blockette is kept by FIP at a time, fewer "window turns" will be performed to access the file. Second, by having fewer blockettes in the UFD, it will be more compact and cached more efficiently.

There are some disadvantages to using the optimum clustersize. The space allocated to the file is a multiple of the clustersize, regardless of what is actually used. This can increase disk usage up to seven percent in a typical case, and up to 14 percent in certain pathological cases when compared to using the minimum clustersize.

Raising clustersize also makes it more difficult to place a file exactly where you want it. FIP always places files on cluster boundaries. In cases where it is desirable to pack files of different clustersize close together with no free space in between, using optimum clustersize often prohibits placing a file exactly where you want it.

On certain files, it may actually be advantageous to lower clustersize below the "optimum" value so that the files may be placed on an otherwise unattainable boundary. It may also be worthwhile to save the allocated but unused space that is wasted when using large clustersize. Run-time system and swap files are good examples.

E) Positioning files

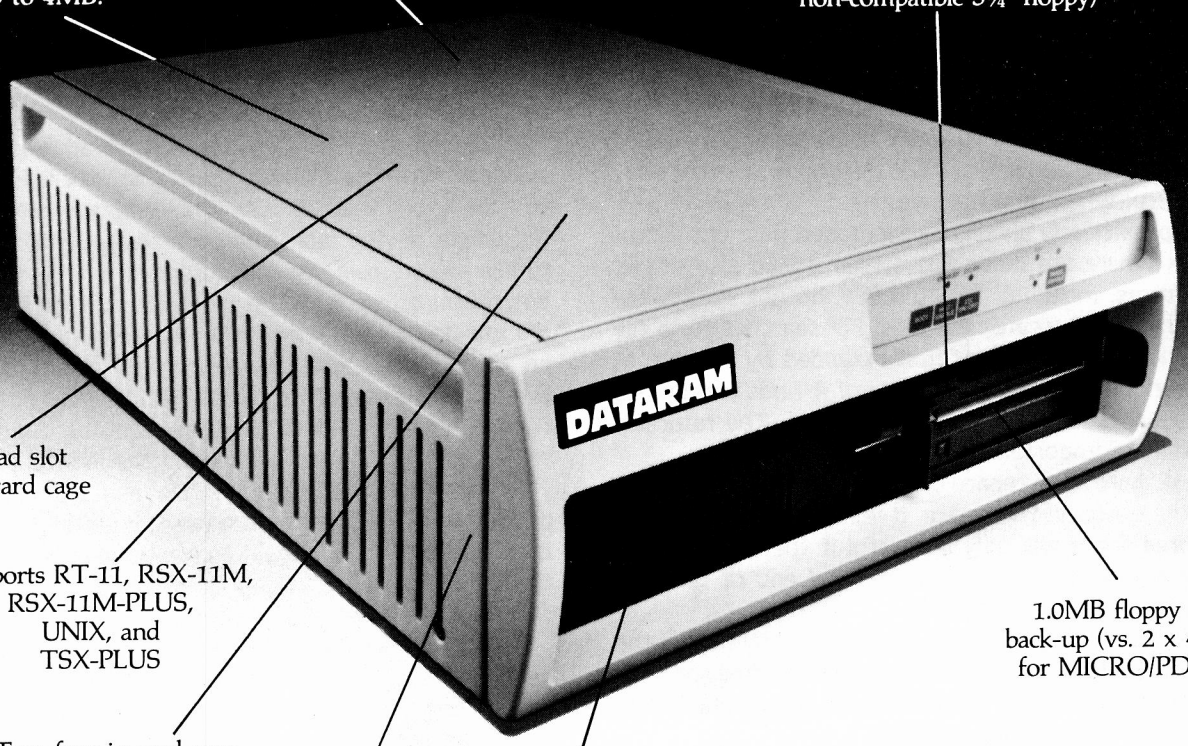
Positioning files can increase system throughput by decreasing seek time. One of the popular optimizations is to position files accessed by a particular program as close to each other as possible, if you cannot place them on separate drives. Here is another case where a single-user optimization causes degradation of performance. As we saw when discussing the fundamental consideration, a particular job is unlikely to get two consecutive accesses to the same disk. So what is the purpose of positioning the files? The only value comes from positioning them near the files being accessed by other jobs, which will also cause them to be positioned near each other.

From the above example, we begin to see that we must position files with consideration for all of the files on the disk. The most used files will be placed closest to the center; the least used files will be placed nearest the edges. This algorithm will be expanded into a powerful file positioning strategy.

F) Making files contiguous

At this point, it is necessary to distinguish between the two types of contiguousness. Files whose block numbers are

Like DEC's.



$9,800 system price*

256KB minimum... up to 4MB!

Media and software compatibility with DEC's RX02 8" floppy (vs. MICRO/PDP-11's non-compatible 5¼" floppy)

8-quad slot Q-bus card cage

Supports RT-11, RSX-11M, RSX-11M-PLUS, UNIX, and TSX-PLUS

Two fans in card cage area (vs. one in MICRO/PDP-11)

RL02-compatible 10MB 5¼" Winchester disk standard; 20MB optional

Space for future 40MB cartridge tape drive.

1.0MB floppy disk back-up (vs. 2 x 400KB for MICRO/PDP-11)

See us at

DEXPO™/Europe 83

West Centre Hotel, London
29, 30 June & 1 July 1983

Only better.

By paying a little more than you would for a MICRO/PDP-11, you can get a lot more! Like an 8" RX02-compatible floppy. 10MB or 20MB 5¼" Winchester and space for a 5¼" cartridge tape. Two fans provide push-pull air flow in the card cage area.

The A22 with LSI-11/23, 256KB, 10MB Winchester, and 8" floppy is only $9,800. 30-day delivery.

For more information, forward this coupon to us, or, for faster response, call (609) 799-0071.

☐ Send information. ☐ Contact me immediately.

Name _____

Company _____

Address _____

City _____ State _____ Zip _____ Phone _____

Return to:
Dataram Corporation, Princeton Road, Cranbury, NJ 08512

*$9,800 is single-quantity domestic price for A22 with LSI-11/23, 256KB, 10MB Winchester and RX02-compatible 8" floppy.

DATARAM

CIRCLE 55 ON READER CARD

Dataram Corporation ☐ Princeton Road ☐ Cranbury, New Jersey 08512 ☐ Tel: 609-799-0071 ☐ TWX: 510-685-2542

contiguous will be referred to as being mapped contiguously. In addition, there is a RSTS/E file attribute known as the contiguous attribute, which tells RSTS/E that the file is contiguous without FIP examining the mapping. This attribute is what causes PIP and DIR to list a file as being contiguous.

There are three main reasons for mapping files contiguously. First, the file is compactly placed on the disk so that large transfers can be made with one access. Second, the entire file can be placed where desired. Finally, in cases where the file need not be extended, it can be given the contiguous attribute to help reduce window turns (FIP overhead) on files larger than 256*7 blocks. Only on single-user systems is it necessarily true that head movement is minimized by making files contiguous.

G) Putting some free space near the center

Assuming that most systems create new or temporary files, free space becomes very active file space and should be positioned near the active files. Two questions arise concerning free space: where and how much?

Free space clearly should be positioned near the center of the disk where the most active files should also reside. Most programs, including DEC utilities, do not specify file position when they create new files. FIP searches for a file location when it is created or first extended by starting at the low numbered end of the disk until it finds contiguous space (if possible) for the specified clusters. The ramifications of this are important.

First, if there is free space between the beginning of the disk and the centered free space, it will be used. Therefore, centered free space will only be useful if there is no other free space below it — the low numbered end of the disk must be packed tightly with files.

Second, FIP has considerable overhead searching for the free space. If we have a choice between placing free space on the high or the low side of the center, we should place it on the low side so FIP will find it faster.

H) Increasing pack clustersize

Pack clustersize can be considered the minimum file clustersize. FIP maintains SATT.SYS, which is a bitmap for the pack clusters on the disk. It tells which pack clusters are in use. If those clusters are large, there are fewer of them and FIP can search the bitmap easier. Also, SATT.SYS will be smaller and there will be fewer accesses to it. FIP only keeps one block of SATT.SYS in memory at a time.

Increasing pack clustersize also results in less variation in clustersize between files. The disk becomes less fragmented and it is easier to pack files tightly, eliminating free space in front of the centered free space mentioned in the previous section.

The disadvantage of increasing pack clustersize is that it tends to waste space. Note that on a pack with pack clustersize 16, even a one block file uses 16 blocks. All storage allocations are rounded up to a multiple of 16.

I) Running REORDR frequently

Using REORDR makes systems run faster. The UFD structure can be considered a tree, and REORDR allows FIP to traverse that tree more quickly. However, while there are many ways to organize MFD/UFDs, the one created by REORDR is almost never optimal.

POSITIONING FILES ON A MULTI-USER SYSTEM

Earlier, we defined the center of the disk. If we could somehow determine which blocks of which files/UFDs were accessed the most during a time sharing session, we could then position those files in their optimum static position by placing the most accessed blocks nearest the center.

There is no practical way to know exactly which files are used the most, but one can make some reasonable guesses by examining the nature of the system. What follows is a description of common files in what is likely to be close to their optimum order. That is, the ones mentioned first should be placed closest to the center.

A) Ordering files around the center

The swap files are frequently accessed on most systems. With the following exception, there is little reason to have more than one swap file on any given disk. The reason for having more than one swap file would be to optimize a system in which interactive jobs swap frequently with event driven jobs, the usual job count is much less than job maximum, and there is good reason to do all swapping on the same disk. Under these conditions, having SWAP3.SYS just in front of SWAP.SYS (and no other swap files) can decrease lost time by requiring slightly shorter seeks between the out-swaps and the in-swaps.

The files OVR.SYS, ERR.SYS, and BUFF.SYS are frequently accessed, if they are used. If OVR.SYS is not used, then the current SIL should be positioned near the center of the disk. Otherwise, it can go to the back edge with the other unused files.

The non-permanently resident run-time systems and resident libraries are frequently accessed. Note that in addition to being accessed at program load time, they are accessed on certain in-swaps. Under certain conditions, lost time can be decreased by positioning these files as near as possible to the swap files. Permanently resident run-time systems are not accessed once a time sharing session has started, so those files should be placed at the back edge of the disk.

MFDs and UFDs should be placed near the center, as will be discussed later.

SATT.SYS is accessed frequently on disks when files are being created or deleted. It should also be near the center.

Free space should be near the center as mentioned earlier.

Next should be frequently used files. Presumably, someone knows which files those are.

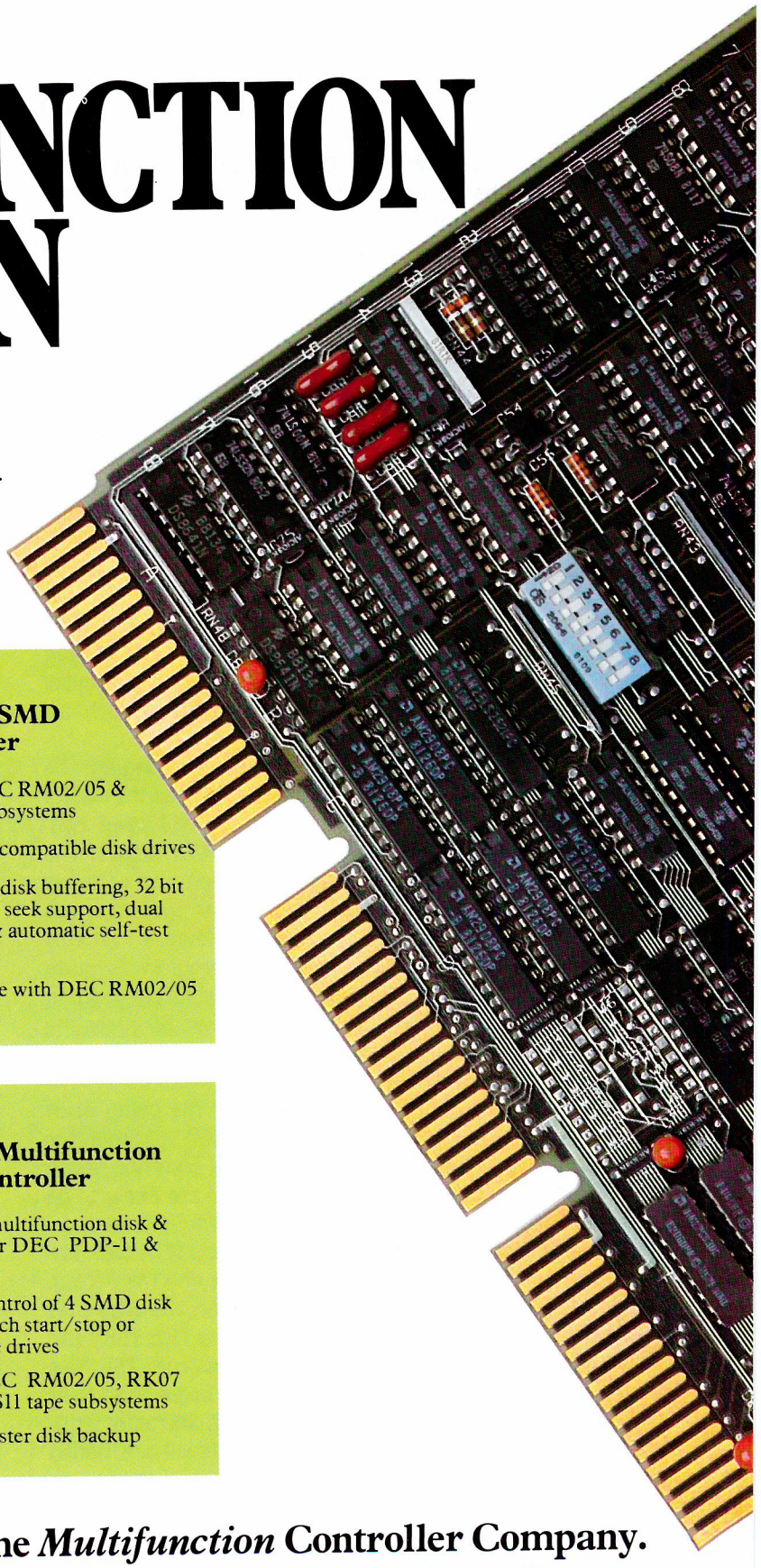
The remaining files should be positioned around the others in decreasing order of use, keeping in mind that the free space in the center will not be used until the free space toward the front of the disk is used.

B) Improvements over ordering around the center

The above rules for positioning files are effective until the system is actually used. When files are created and deleted (e.g., when they are edited or rebuilt), the new versions end up where FIP puts them, and free space is created

MULTIFUNCTION BY DESIGN

Spectra Logic delivers single and multifunction disk/tape controllers for DEC users.



Now you can improve system performance and reduce system cost with our popular family of DEC compatible disk and tape controllers.

We're Spectra Logic. The company that introduced the multifunction concept back in 1979. And we've been revolutionizing the controller market ever since.

Our multifunction disk/tape controllers provide the high-performance and added value you need to stay competitive. (Compare our $2,900 SPECTRA 21 OEM price with separate disk and tape controllers.)

They eliminate the cost and need for separate disk and tape controllers, saving CPU slots, increasing reliability, and reducing power and spares requirements.

All of our controllers are smart, firmware-intensive, single board units. They support a wide range of independent disk and tape drives while emulating standard DEC subsystems. And they're backed by the industry's most comprehensive one year warranty.

Interested in single or multifunction controllers? Then contact the company that set the standard in the DEC market. Spectra Logic.

Call or write us today.

SPECTRA 12 PDP-11/VAX SMD Disk Controller

- Emulation of DEC RM02/05 & RK06/07 disk subsystems
- Attaches 4 SMD compatible disk drives
- Features 3 sector disk buffering, 32 bit ECC, overlapped seek support, dual port disk drives & automatic self-test microdiagnostics
- Media compatible with DEC RM02/05

SPECTRA 21 PDP-11/VAX Multifunction Disk/Tape Controller

- Industry's first multifunction disk & tape controller for DEC PDP-11 & VAX computers
- Simultaneous control of 4 SMD disk drives & 8 ½-inch start/stop or "streaming" tape drives
- Emulation of DEC RM02/05, RK07 disk & TM11, TS11 tape subsystems
- Ideal for Winchester disk backup

The *Multifunction* Controller Company.

SPECTRA LOGIC

CIRCLE 183 ON READER CARD

1227 Innsbruck Drive
Sunnyvale, CA 94086
Telephone (408) 744-0930
TWX 910 339-9566
TELEX 172524 SPL SUVL

Western Regional Sales Office:
(214) 934-9294
Eastern Regional Sales Office:
(216) 826-3137
New England District Office:
(914) 623-0502

elsewhere on the disk. These new creations are not likely to be well placed. Consequently, several additional steps should be taken.

First, as a general rule, files which are likely to be deleted should be placed near the center. This will force the resulting free space to be in a good location. Furthermore, if there is sufficient free space, the files likely to be deleted without being otherwise accessed, can be placed near the high numbered end of the disk. The free space generated when they are deleted will be used last; the well-positioned free space will be used first.

It may seem difficult to guess which files are likely to be deleted, but in reality it is quite easy. With few exceptions, the most recently created files are the most likely to be deleted soon. Think about it, keeping in mind that most editors, etc. do not modify old versions of files, but create new ones, deleting the old ones when done.

C) Determining the most accessed files

With the large file processor, it is relatively easy to scan the list of open files. One might conclude that monitoring this list for the most commonly opened files would reveal the most heavily accessed files. Unfortunately, such attempts will produce poor results.

Consider a typical system which runs error logging, the spooling package with batch processors, and uses the CUSPs frequently. Error logging leaves the error file open continuously and only accesses it in the event of errors. The spooling leaves a number of work files open and accesses them only occasionally. To load CUSPs, the system must open them, read them, and close them. However, all this happens so quickly that the monitoring program is quite likely to never see it happen (unless it is absolutely hogging the system). So any likely monitor program would guess the relative frequency of accesses to those files exactly backwards.

There is a large difference between a file being open and being accessed. Accesses are difficult to monitor without large overhead. Fortunately, some common sense observations can help determine which files are likely to be accessed most often.

Compiled BASIC programs and RT11 and RSX task images, can be loaded into memory with just one file access, provided that they are not overlaid. However, the ones that are overlaid (including PIP.SAV, EDT.TSK, and TKB.TSK), can be accessed many times in the course of one run. TSK files larger than 115 blocks and SAV files much larger than their core images are also overlaid. There is considerable sentiment that suggests that LOGIN should be well positioned because of its frequent use, but consider that in the course of one task build, more accesses are made to TKB.TSK than to LOGIN all day long. The same phenomenon is seen with PIP, EDT, ED2, and LBR.

Files which are accessed by EDT or compilers are typically accessed one block at a time. This implies that even if the file is only opened once, it will be accessed many times. Note that any file with a source file extension likely falls into this category. Data files, especially randomly accessed ones,are typically accessed a few blocks at a time. Object libraries, if they are used, are accessed many times per use. LOG files are not likely to be accessed more than once, sequentially, and possibly many blocks at a time (as PIP would access it).

Programs which access particular files each time they are run are most likely accessed less often than the files they access, especially if they do many accesses to those files.

D) Positioning across cylinder boundaries

Discussions about disk cylinder boundaries are rare in PDP-11 operating systems. On RSTS/E systems, the system takes care of mapping virtual blocks onto physical blocks on the disk without help from the programmer. For the most part, little is gained by positioning files across the minimum number of cylinder boundaries.

Files on most systems are accessed one or a few blocks at a time. The blocks transferred on any given access are unlikely to cross a cylinder boundary regardless of how files are placed. The fundamental consideration shows there is little to be gained from positioning such files across minimum cylinder boundaries, since they are unlikely to be accessed twice in succession.

There are disadvantages in trying to avoid crossing cylinder boundaries. Due to FIP's alignment algorithm, cylinder boundaries always straddle clusters for any clustersize greater than one. On any large disk and on any optimized disk, the pack clustersize will be greater than one. An attempt to have all files straddle minimum cylinder boundaries would create free space at many cylinder boundaries. The degradation from scattering free space would more than make up for any gains from avoiding cylinder boundaries. Furthermore, attempts to position files with large clustersize across minimum cylinder boundaries tend to position them far from where they are desired. (Explanation of this effect is beyond the scope of this article; the mathematically inclined are referred to the Chinese Remainder Theorem.)

There are benefits, though, to positioning the most useful files. Non-permanently resident run-time systems are ideal candidates for minimum-cylinder positioning. If possible, the entire file is transferred in one access. Since there are just a few of these files, it is worthwhile to straddle the minimum number of boundaries with these, if they are used often.

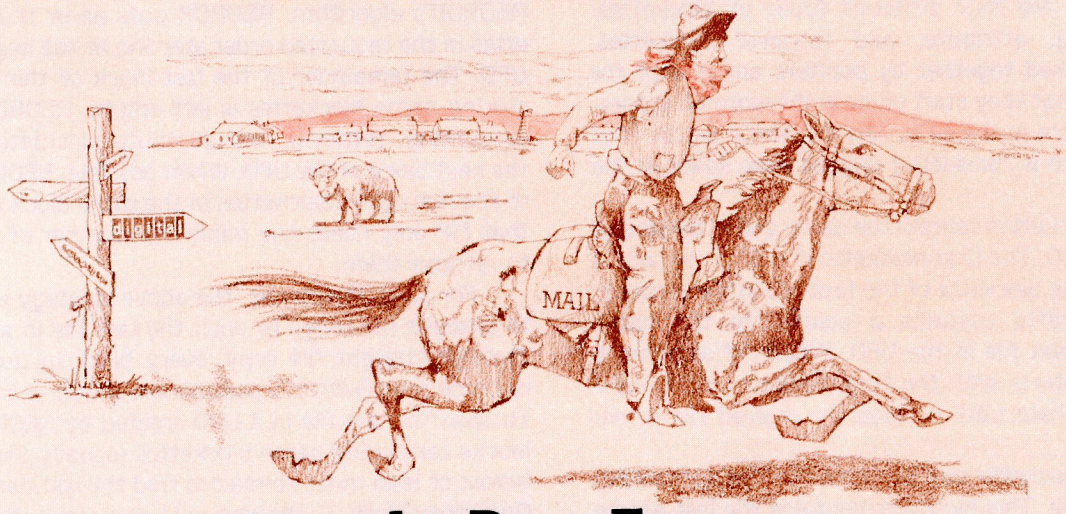
Similarly, frequently accessed BACs are good candidates for intra-cylinder positioning. Notice that on large disks, most of these files will miss cylinder boundaries anyway, regardless of the positioning algorithm. Ditto for unoverlaid TSKs and SAVs.

Overlaid task images are another story. They tend to be larger (harder to position), and they are accessed by a smaller number of blocks at a time, so they tend not to be accessed across cylinder boundaries. The benefit is small compared to the effort and other resulting degradations.

The swap files are large and will likely cross many cylinder boundaries regardless of where they are placed. Optimum placement can at best avoid one cylinder boundary. Once again, the benefit is very small compared to the harm resulting from the likely off-center placement.

The last "files" to be considered are the MFDs and

The most exciting
MESSAGE DELIVERY NEWS



since the Pony Express...

DIGITAL ANNOUNCES DECmail/RSTS VERSION 1.0 ELECTRONIC MAILING SYSTEM AT AN AFFORDABLE PRICE TO SPUR YOU ON!

Once, a letter had to travel tortuous routes to reach its destination. Via the swift Pony Express. Or more recently, by way of the office message box.

Now, the wide open spaces between your office and company headquarters, the 18th and 22nd floor, the New York office and the Toronto branch are bridged easily — electronically — with DECmail/RSTS.

DECmail/RSTS provides a low cost, reliable message handling system for RSTS/E sites. With this new system, you can create, edit and process messages — from simple memos to complicated specifications. You can forward your message for delivery almost instantly to a business associate down the hall, across town, around the country or overseas.

DECmail/RSTS also brings you up-to-date in the morning and holds messages for you while you're out. It will organize your material for you — storing, searching and retrieving messages as needed. If you have a problem, an on-line help facility comes to the rescue.

If this sounds as easy as falling off a horse, it is. DECmail/RSTS was designed by the original RSTS/E development team. They put together a system with easy to learn commands and flexible options that make DECmail/RSTS sophisticated enough for advanced users yet simple and uncomplicated for computer novices.

DECmail runs on RSTS/E Versions 7.2 and 8.0. No update will be needed until after RSTS/E V8.0 is released. Thanks to DECmail update kits, you won't be saddled with an obsolete software product down the line.

What does all this cost? A lot less than you might think. And the increase in productivity in your working environment — not to mention the wonders of electronic communication — is worth every penny.

Find out about this new electronic message system for your office. Complete, clip out and return the coupon — pronto!

REWARD!

Request full information on your DECmail/RSTS message system now. If you order within 90 days, you will receive a RSTS western-style belt buckle FREE.

Mail the coupon today or call 1-800-DEC-INFO any working day between 8:30 a.m. and 5:00 p.m. E.S.T.

MAIL this coupon to: Digital Equipment Corporation
Direct Response Center, POB 279
Maynard, MA 01754

Please send me complete information on DECmail/RSTS and my eligibility for a Western belt buckle.

Name _____

Title _____ Telephone No. ( ) _____

Company _____

Address _____

City _____ State _____ Zip _____

CIRCLE 189 ON READER CARD

UFDs. They are always accessed one block at a time, so no single access crosses a cylinder boundary. The fundamental consideration shows that it is unlikely to access the same MFD/UFD twice in succession, so there is almost nothing to be gained by avoiding cylinder boundaries with MFDs/UFDs.

OPTIMIZING MFDS AND UFDS

UFDs are divided into chunks of eight words called blockettes. There are four different types of blockettes: name, accounting, attribute, and retrieval blockettes. Blockettes are linked together by pointers and except for the requirement that they start on an eight-word boundary, can reside anywhere in the UFD. Any blockette whose first two words are zero is considered unused (free space in the UFD).

The name blockettes contain the name and extension of the file in RADIX-50. The first blockette in a UFD contains a pointer to the name blockette of the first file in the account. Each name blockette contains a pointer to the name blockette of the next file in the UFD. When FIP searches a UFD for a file, it starts with the first blockette of the UFD and follows this chain until it finds the desired file or exhausts the list.

Each name blockette has a pointer to the accounting blockette for its file. This blockette contains the last access date, the number of blocks in the file, the creation date and time, the associated run-time system, and the file cluster-size.

Each name blockette also has a pointer to the first attributes blockette if the file has any attributes. The attribute blockettes contain file attributes (up to seven per blockette), plus a pointer to the next attribute blockette for the file. There is a maximum of two attribute blockettes per file.

Finally, each name blockette contains a pointer to the first retrieval blockette. Each retrieval blockette contains the starting device cluster numbers for up to the next seven clusters of the file, plus a pointer to the next retrieval blockette, if it exists.

In a sense, the internal structure of the UFDs is a tree: the name blockettes form the root of the tree, the retrieval and accounting blockettes form branches, and the attribute blockettes are leaves off of the accounting blockettes. This tree can be searched in the forward direction, that is, from the root to the leaves, but not backwards. Since all of the nodes of the tree are found only by following pointers, the nodes (blockettes) can be located in any order in the UFD. However, we will see that some orders are better than others.

A) Straight file copy

When files are copied into an empty UFD one at a time, their associated UFD blockettes are tightly packed into the UFD, one after another, starting at the first available location in the UFD. To scan the list of name blockettes (i.e., do a file lookup), FIP reads through the UFD sequentially, until it gets to the name blockette it is seeking. Straight file copy guarantees that FIP will only read a block of the UFD once when performing a file lookup.

It is a common belief that this system is efficient. It is much better than the tangled mess that results from creating and deleting files at random. Furthermore, most of the disk structuring utilities leave their UFDs this way. Unfortunately, except for very small directories and a few pathological cases, straight file copy is never best.

B) REORDR

The other well-known UFD optimizing technique is REORDR's algorithm. REORDR puts all of the name blockettes in the requested order starting at the beginning of the UFD. The remainder of the last block of the UFD used for storing name blockettes is left empty. REORDR writes the accounting, retrieval, and attribute blockettes starting in the next block of the UFD. When possible, REORDR writes all of the non-name blockettes in the same block of the UFD so that FIP only needs the minimum number of disk accesses to retrieve them.

For large directories, the above strategy is far superior to straight file copy. To open the last file in a UFD created through straight file copy, every block in use in the UFD must be read (with the possible exception of the last block). To open the last file in a UFD created by REORDR, only the blocks containing name blockettes (usually one-third of the blocks or less) must be read to find the last name blockette. One other block — the one with the accounting, attribute, and the first retrieval blockette — also must be read. So in large directories created by REORDR, approximately one-third the UFD accesses are needed to open files as in directories created by straight file copy.

Unfortunately, REORDRed directories are not always optimal. Consider the case in which there are just a few small files in a UFD. If the UFD was created by a straight file copy, all of the directory information is in the first block of the UFD and can be retrieved with one access. If the UFD is REORDRed, the name blockettes will be in the first block, and the rest of the directory information will be in the second block, requiring two accesses to open any file!

C) DOPTERed UFDs

Up to this point, we have hinted that there are a number of improvements to the UFD structure possible. In developing DOPTER, the System Performance House combined the best attributes of straight file copy and REORDR, and added some new optimizations.

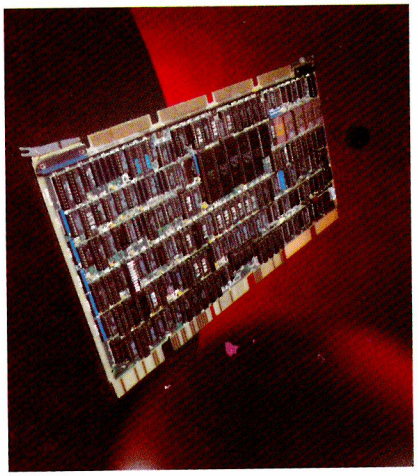
DOPTER has an internal routine somewhat similar to REORDR. If all the UFD information will fit into one block, it is put into the first block of the UFD. If there are more blockettes than will fit into one block, the name blockettes are separated from the others as in REORDR, but with some interesting differences.

The most obvious way to reduce the overhead of file lookups is to have the desired files as near as possible to the beginning of the name blockette list. While REORDR makes a good attempt at this with its reverse order sort on creation or access date, it certainly does not do nearly as well as a good heuristic algorithm which will accept help from the user. (Note that sorting in alphabetical order on file name or extension is basically useless for optimization.)

Since DOPTER is very good at placing the most used files at the front of the UFDs, DOPTER leaves room in the

ACCESS

Access the X.25 world. Today.



A Unibus/DMA system that provides DEC minicomputers access to X.25 networks. Great!

Now available for RSTS/E

Today's X.25 public networks offer an alternative to the high cost of dedicated lines and the slow speed of dial-up connections. ACC's X.25 products help major government agencies and Fortune 500 companies take advantage of this alternative.

Our products plug directly into your DEC host's UNIBUS or LSI-11 Bus. They off-load X.25 protocol processing and transfer data to your host CPU by means of high-speed Direct Memory Access (DMA).

Three DEC/X.25 Interfaces. ACC has three major X.25 products to meet the requirements of your application. All are microprocessor based. All are certified for operation on Telenet and other public packet networks. All comply with CCITT's Recommendation X.25 for levels 1, 2, and 3. And all are available for delivery today.

1. Terminal Networking. With the IF-11/X.25 PLUS, remote X.25 network terminals can access your host as if they were locally connected. The IF-11/X.25 PLUS can be configured to support any combination of up to 32 local and remote terminals. Additionally, local terminal users have the option of connecting to other hosts on the X.25 network. All PAD (Packet Assembly/Disassembly) functions (CCITT X.3, X.28, X.29) are coded into subsystem firmware, without impacting your host CPU.

2. High Speed File Transfer. The IF-11/X.25 connects your host to an X.25 network. It provides up to 32 full-duplex virtual circuit connections to a VAX or PDP-11, at line speeds of 56 Kbps (with even faster line speeds available). The IF-11/X.25 is ideal for file transfer applications to remote network locations or for any application that needs direct access to an X.25 network.

3. LSI-11 Bus Systems. The IF-11Q/X.25 network access system is functionally identical to the IF-11/X.25, but designed for your PDP-11/23.

Access Is Our Business. For over a decade, beginning with ARPANET, ACC personnel have designed and manufactured a variety of systems to access packet-switched networks. ACC's X.25 products are designed to meet your custom applications. For example, we have customized X.25 systems with the following options: (a) 256 byte packet size, (b) ADCCP frame level, (c) Point to Point capability (DCE version).

If you need access to the world of X.25, phone us at (805) 963-9431. Today.

ACC

Accessing the World ... Today.

Associated Computer Consultants
720 Santa Barbara Street
Santa Barbara, CA 93101
TWX 910 334-4907
(805) 963-9431

first block of the UFD for all of the blockettes of the first two files. Thus, the most used files can be opened with only one UFD access, a 50 percent improvement over REORDR. Except for this improvement, all of the name blockettes remain segregated from the other blockettes.

To appreciate DOPTER's most significant attribute, it is necessary to understand FIP's method of allocating free space in UFDs. When FIP needs free space, (i.e., when it is creating a new file or extending an old one), it first searches the current UFD block in memory for free space. If it finds what it needs there, FIP uses it. Otherwise, FIP searches the UFD sequentially from the beginning, examining each blockette to see if it is in use.

When FIP finds its current block full, it will search every allocated blockette before it finds a free one if the UFD was created by straight file copy. If the UFD was created by REORDR, it will search through all of the name blockettes until it comes to the small amount of free space left at the end of the name blockettes (if any). If that space has been used (which it will be after a few new files are created), FIP must search to the end of the allocated blockettes, just as in straight file copy.

The solution is to leave some free space near the beginning of the UFD. Since FIP only accesses one block at a time in the UFDs, there is little to be lost with this strategy. DOPTER leaves blocks two through nine empty. Thus, FIP finds free space with a minimum number of reads. In a large UFD, this strategy can cut file open overhead by 80 percent.

There is another important benefit from leaving free space near the start of the UFD. In a typical UFD, files are both created and deleted in somewhat random order. If an otherwise tightly packed UFD has a few files deleted, there will likely be a few free blockettes scattered throughout the UFD. When FIP reallocates the space, it scatters the blockettes throughout the UFD, causing FIP to perform many disk accesses to do file lookups, creations, etc. FIP generates what is commonly referred to as a tangled directory.

When free space is left at the front of the UFD, files still are deleted wherever they are. However, when new files are created, they are built as they would be by straight file copy. As we have seen, straight file copy is not best, but it is far better than the alternative.

D) DOPTERed MFDs

MFDs are similar in structure to UFDs, except that the name blockette entries may be for UFDs as well as files. However, MFDs provide special opportunities for optimization.

The first cluster of an MFD must be at device cluster one. MFDs are normally extended contiguously. However, this is a poor strategy because it guarantees that some of the most frequently accessed blocks on the disk are at the very edge. There are several ways to improve the situation.

An MFD may be pre-extended anywhere. That is, clusters two through seven may be placed on any cluster boundary, preferably near the center. Not only does DOPTER place the other clusters near the center, but it only uses the first block of the first cluster. Thus, instead of building the entire MFD at the edge of the disk, it only uses one block there. All the others are where they belong. Once again, the fundamental consideration shows us that we have lost very little by removing the contiguity of the MFD because consecutive accesses are seldom made to an MFD. (Note: all blocks of the MFD are available if they are needed, but that is seldom necessary.)

E) Additional MFD/UFD optimizations

Many files have associated attributes which are unused. The prime example is task images created by the task builder. The task builder uses RMS I/O to create its task images, but the attributes are never used. EDT Version 1 put attributes on its output files, but in the case of most source files, they were unneeded by the compilers or editors. The unneeded attributes take up space in the UFDs and create extra overhead for FIP. DOPTER effectively eliminates supernumerary attributes by recognizing source file and task image extensions. The same can be done with PIP.

Another UFD optimization can be made with files with the contiguous attribute. Once such a file is opened, RSTS/E knows the mapping for the entire file from having read the first retrieval blockette. UFD accesses can be minimized by putting the rest of the retrieval blockettes for large contiguous files at the very end of the UFD, away from the useful information in the UFD. In this manner, they will never be accessed, not even when searching for free space. This strategy assumes that there is sufficient free space in the UFD, which is almost never a problem.

OTHER TOPICS

A) DLA versus DLW

When initializing a RSTS/E volume, RSTS/E can be made to keep either the date of last access (DLA) or the date of last write (DLW). In addition to the arbitrary needs of the installation, there are several trade-offs when considering one against the other.

The date of last access/write is stored in the accounting blockette. If the DLW option is chosen, this blockette only needs to be rewritten when a file is modified. In the case of read-only files, of which executable programs are a major example, a physical UFD access can be saved by using DLW instead of DLA. Note that all writes to UFDs generate physical accesses — they are not cached.

On the other hand, if REORDR is used to sort the files in the UFD in reverse order by access date, a better order is likely to occur if DLA is used instead of DLW. If DLW is used, some frequently accessed programs/files can have very old last access dates and thus be put at the end of the UFD. With DLA, they would be put near the front.

On balance, DLW is better with all small directories because only a block or two will ever be read to search the name blockette list for any file. The savings from not writing the last access date on read-only files will more than compensate for the slightly longer lookup path after running REORDR. DLA is better only on large accounts where heavily used read-only programs/files will be placed at the front of the UFD by REORDR.

B) New Files First

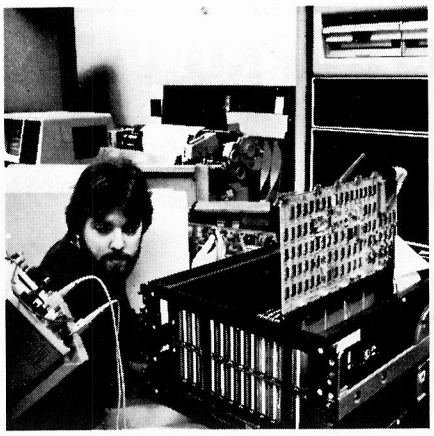
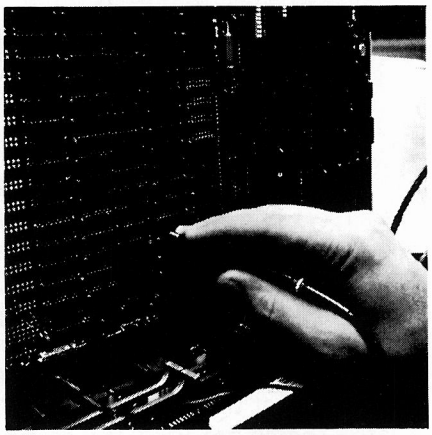
It is easy to overrate the advantages of using New Files First (NFF). The arguments for using NFF are as follows.

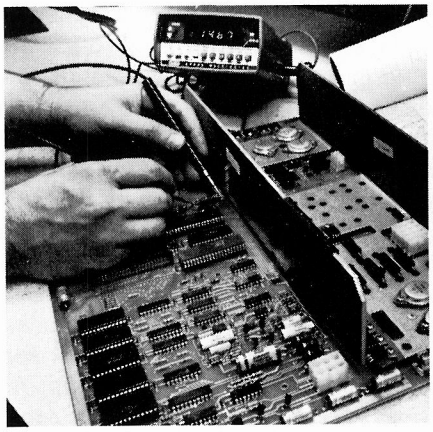
1). It is convenient to have the most recently created files at the front of the directory listing. (This has nothing to do with performance.)

. . . continued on page 30

The Single Source for Digital Products Service

DEC Module Repair, Exchange or Replacement


Photos show how we're fully equipped to repair and test DEC,* PDP-11* and TERMINAL MODULES. All modules we repair are 100% inspected.

Expert, Fast and Reliable Work

Serving the computer systems market since 1973, we are now troubleshooting, repairing and testing hundreds of modules weekly.

We can turn around a critical repair job in just 24 hours! Ten-day service is normal.

The modules we return to you are guaranteed to meet or exceed the manufacturer's specifications. And we prove it by submitting an inspection and test report.

For emergency service, 7 days a week, 24 hours a day, call

614/890-0939

DIGITAL PRODUCTS REPAIR CENTER
939 Eastwind Drive
Westerville, Ohio 43081
614/890-0939

*DEC and PDP are registered trademarks of Digital Equipment Corporation.

H-1580

CIRCLE 140 ON READER CARD

RELIANCE ELECTRIC

THE RSTS PRO VISITS THE NCC

By Carl B. Marbach

Los Angeles is busy. Even at night (or early morning) the airport is busy and choked with traffic. There are no rental cars and the taxi ride to Anaheim requires the national debt to be raised substantially. From the East coast it takes twelve hours of travel to make the trip and when you finally roll into bed in the plastic motel it seems only marginally worthwhile. Morning comes early in California and arriving at the Anaheim Convention Center it looks like a large conference with the routine exhibitors, coffee in styrofoam cups, sore feet and lots of exciting equipment. The press rooms are small, but there are two of them. After the obligatory cup of coffee in the first, I wander into the seccond. There are tables of typewriters for the reporters and . . . five tables of DECmate II's! Ten in all, complete with two letter quality printers and two DECpersons to help all the unfortunate ones who do not know how to use this terrific machine. So, the National Computer Conference is off to (for me) a great start; I feel at home.

The DECmate II might be the most hidden secret DEC has. It is a really fine machine. With its PDP-8 based word processing software (among the best in the industry), it also has CP/M capability which means it can run a large number of microprocessor based software packages. These packages include professional time and billing, spreadsheet planning, many financial planning languages, BASIC compilers and many, many others. The price is getting more competitive all the time and if you haven't tried one yet just find your nearest DIGITAL computer store and take a good look. Note that the DECmates are being sold by the DEC computer stores; they also will be selling MICRO-11s and Vax 11/730s. They will NOT be selling Rainbow 100s and Professional 300's for the time being, so that while supplies are short the DEC stores will not be competing with the independent DEC dealers. But, when they are finally making more than they can sell . . .

The National Computer Conference is a massive event, taking up the whole Anaheim Convention Center, the Disneyland Hotel and several large (and hot) tents in the parking lot. It used to be known as the Fall Joint Computer Conference (The Fall Joint for short, but that means something else now) and the Spring Joint Computer Conference, each in its own season. The show was combined into one event when it became too expensive for exhibitors to display at both shows, and when travel to two shows was too much for people. The price we pay for one show is the crowd. Exhibitors love it; attendees put up with it. When you attend, be prepared.

For most of us it is a Howard Johnsons for computers only there are more than 27 flavors; there are hundreds. All the majors are there and it is very informative to look and learn about the computer systems we don't use. When you run out of systems, you can start with the peripherals. All the old ones you knew about and many new ones you haven't thought about yet. A new letter quality printer for $995 list (20 CPS), A VT100 emulating terminal, a page size (66 lines) CRT, 100 MB disks in 5½ inch size, 400+ MB disks in slightly larger configurations, streaming tape drives, cartridge tape drives, laser printers, supplies, desks, and, of course, software.

It is an impressive show. I mentioned that in general and was told that "without the micros, it wouldn't be nearly as exciting." Right. VisiCorp (the VisiCalc people) introduced VisiWord and VisiSpell on the xxx personal computer (it will be on the Rainbow 100 in the future). While the word processing isn't that impressive, the Speller is dynamite. DECword, WORD-11 and DECmate people take note: it is much better. It will even make a very good guess at what you were TRYING to spell.

DEC was not outdone by anyone. The DECbooth was filled with new and old DEC products. The Personal computer line was well represented, as was the MICRO-11 running all PDP-11 operating systems (even RSTS) and the VAX 11/730 with its see through panels showing the inside of this well packaged VAX.

The personal computers were well represented with their color graphics really making the booth look good. The telephone management system was alive and well on the Pro 350 and will be an impressive product; this is due out in the fall and will be the subject of a complete article in **Personal and Professional** (THE magazine for DEC personal computers). DEC also introduced IVIS, Integrated Video Information System which is a Pro 350 connected to a laser disc video recorder. Between the color graphics, the TV picture and the Pro 350 controlling the video disk it is an amazing array of gadgetry that will tie together two pieces of electronic equipment which DEC hopes will make a training tool par excellence. Can you imagine having the computer control the video disk bringing up random sections of a training manual depending on the user's response to the prior video section?

VAX users were not left out. Enter the new VAXstation 100. This gigantic (18"??) display is connected to a MICRO box that houses a Motorola 68000 microprocessor. It emulates up to four VT-100 "windows," has bit-mapped graphics with a resolution of 1000 x 800, and can put a graph in three quarters of the picture and still give you complete VT-100 functionality in the remaining quarter. Oh yes, it has an optional mouse. Want to see a VAX that looks like a LISA?

The NCC is a dizzying display of equipment and services; I have not even touched on the academic sessions which appear to be of high quality. There is more to do than you can in one week. Whatever time you can spend is not only informative but also good for the soul. We all need to be rejuvenated once in a while. This could be the place.

♥

COMPILE.BAS: COMPILATION AID FOR BASIC PLUS TWO

By Alexander Ehrlich, Arel Software Ltd., Yavneh, Israel

1. INTRODUCTION

While working on a large data processing project, I found that a lot of my nice BASIC PLUS programs were growing into not so nice BASIC PLUS 2 programs. Gradually my account filled up with .ODL, .OBJ and .TKB files. Also, whenever I ran a compilation, my terminal was tied up for at least several minutes. COMPIL.BAS was written to solve these and other problems.

COMPIL.BAS helps the BASIC PLUS 2 programmer by doing the following:

1) Generates the auxiliary files that are necessary for a compilation.
2) After the compilation, deletes any auxiliary files that are no longer needed.
3) Performs all the commands needed to run a compilation.
4) Runs the compilation in "background" so as not to tie up the user's terminal. See DETACH option below.

2. USING COMPIL.BAS

The programmer uses COMPIL by typing the following:

RUN COMPIL

(The program responds with "PROGRAM NAME?")

program name / option list

The extend part of the program name will be assumed to be .B2S unless explicitly stated.

The following can be in the option list, separated by commas:

1) CHAIN — The program will be compiled with the chain option
2) NOCHAIN — The program will be compiled with the no chain option
3) LINE — The program will be compiled with the line option
4) NOLINE — The program will be compiled with the no line option
5) DEBUG — The program will be compiled with the debug option
6) XFILE — The program can use file channels 13 and 14
7) DETACH — The compilation will run in "background" on a pseudo keyboard. This will enable the programmer to log on again and continue working (or at least play PAC-MAN) until his or her compilation is finished. The programmer will receive a message when the compilation is finished and any error messages will be stored in the .ERR file.

   If DETACH is not specified, then the compiler commands will be forced into the job stream automatically.

3. NOTES TO THE SYSTEM MANAGER

Although COMPIL is an aid for BASIC PLUS 2 programming, it is written in BASIC PLUS. Place the .BAC version of COMPIL into a convenient account (such as {1,2}) with a protection code of <232>. COMPIL uses privileged commands.

The default option specified at line number 65 is CHAIN. Substitute any options that you prefer, separated by commas.

Examine carefully lines 210 through 250. These lines generate the command and overlay files for our installation and might have to be modified.

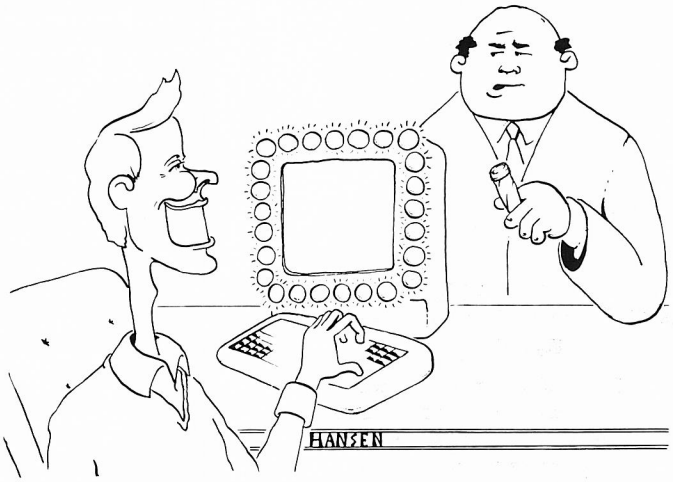
```
10      EXTEND
20  !                                                                   &
!       PROGRAM         - COMPIL.BAS<232>                               &
!                                                                       &
!       DESCRIPTION     - BASIC PLUS 2 COMPILATION AID                  &
!                                                                       &
!       PROGRAMMER      - Alexander Ehrlich                             &
!                         Arel Control Systems Ltd.                     &
!                         Yavneh, Israel   70600                        &
!                                                                       &
!       DATE            - 1-October-82                                  &
!                                                                       &
!-----------------------------------------------------------------------
50  !                                                                   &
!                                                                       &
!                                                                       &
!                       MAIN PROGRAM                                    &
!                                                                       &
!                                                                       &

60      CR$ = CHR$(13%)                                                 &
\       DIM OPTION.LIST$(7%), OPTIONS%(7%), SYS1%(30%)                  &
\       OPTION.LIST$(1%) = "DETACH"             !OPTION LIST            &
\       OPTION.LIST$(2%) = "CHAIN"                                      &
\       OPTION.LIST$(3%) = "NOCHAIN"                                    &
\       OPTION.LIST$(4%) = "LINE"                                       &
\       OPTION.LIST$(5%) = "NOLINE"                                     &
\       OPTION.LIST$(6%) = "DEBUG"                                      &
\       OPTION.LIST$(7%) = "XFILE"                                      &
\       DUMMY$ = SYS(CHR$(6%) + CHR$(9%) + CHR$(10%))                   &
\       KEYBOARD% = ASCII(MID(DUMMY$, 2%, 1%)) * .5                     &
                                        ! OBTAIN KEYBOARD NUMBER        &

65      DEFAULT.OPTION$ = "CHAIN"                                       &

67      GOSUB 100                       ! OBTAIN PROGRAM NAME           &
                                        !  OPTIONS                      &

70      GOSUB 200                       ! CREATE COMMAND AND OVERLAY    &
                                        !  FILES                        &

80  !                                                                   &
!                   NON DETACH - RUN COMPILATION ON USER'S KEYBOARD     &
!                                                                       &
        IF NOT OPTIONS%(1%)                                             &
        THEN    GOSUB 300               ! CREATE INSTRUCTION TEXT       &
                                                                        &
\               GOSUB 400               ! FORCE TEXT TO KEYBOARD        &
                                                                        &
\               GOTO 99                                                 &

90  !                                                                   &
!               DETACH - RUN COMPILATION IN BACKGROUND                  &
!                                                                       &
        PRINT " . . . DETACH"                                           &
\       DUMMY$ = SYS(CHR$(6%) + CHR$(7%))                               &
                                                                        &
\       GOSUB 500                       ! OBTAIN PASSWORD               &
                                                                        &
\       GOSUB 600                       ! RUN PSEUDO JOB                &
                                                                        &
\       GOSUB 700                       ! SEND MESSAGE TO USER'S        &
                                        !  TERMINAL                     &
                                                                        &
\       DUMMY$ = SYS(CHR$(6%) + CHR$(5%))               ! LOGOUT       &

99      GOSUB 32000                                                     &
                                                                        &
!-----------------------------------------------------------------------

100 !                                                   G100            &
!                                                                       &
!               OBTAIN PROGRAM NAME AND OPTIONS                         &
!                                                                       &
!                                                                       &
!                                                                       &
```

. . . to page 22

```
105  !
!                READ PROGRAM NAME AND OPTIONS                          &
!                                                                       &
        PRINT                                                           &
\       PRINT "PROGRAM NAME " ;                                         &
\       INPUT LINE PROGRAMA$                                            &
\       PROGRAMA$ = CVT$$(PROGRAMA$, 132%)                              &
\       GOTO 32000 IF PROGRAMA$=""                                      &
\       OPTIONS%(I%) = 0% FOR I% = 1% TO 7%                             &
\       IF INSTR(1%, PROGRAMA$, "/")=0%                                 &
        THEN    PROGRAM$ = PROGRAMA$                                    &
\               OPTIONS$ = DEFAULT.OPTION$+","  ! DEFAULT = "CHAIN"     &
\                       GOTO 120                                        &
110     PROGRAM$ = LEFT(PROGRAMA$, INSTR(1%, PROGRAMA$, " ")-1%)        &
\       OPTIONS$ = RIGHT(PROGRAMA$, INSTR(1%, PROGRAMA$, "/")+1%)+","   &

120 !                                                                   &
!               CHECK OPTIONS                                           &
!                                                                       &
        COMMA.LOC% = INSTR(1%,OPTIONS$, ",")                            &
\       GOTO 140 IF COMMA.LOC%=0%                                       &
\       OPTION$ = LEFT(OPTIONS$, COMMA.LOC%-1%)                         &
\       GOTO 130 IF LEFT(OPTION.LIST$(I%), 3%)=LEFT(OPTION$,3%)         &
                FOR I% = 1% TO 7%                                       &
\       PRINT "ILLEGAL OPTION: " ; OPTION$                              &
\       GOTO 105                                                        &

130     OPTIONS%(I%) = -1%                                              &
\       OPTIONS$ = RIGHT(OPTIONS$, COMMA.LOC%+1%)                       &
\       GOTO 120                                                        &

140     IF (OPTIONS%(2%) AND OPTIONS%(3%)) OR                           &
                (OPTIONS%(4%) AND OPTIONS%(5%))                         &
        THEN    PRINT "BOTH NO AND YES ON SAME OPTION"                  &
\       GOTO 105                                                        &

145     OPTION$ = ""                                                    &
\       OPTION$ = OPTION$ + OPTION.LIST$(I%) + ","                      &
                IF OPTIONS%(I%) FOR I% = 2% TO 6%                       &
\       OPTION$ = "/" + LEFT(OPTION$, LEN(OPTION$)-1%)                  &
                IF OPTION$<>""                                          &

150  !                                                                  &
!               CHECK THAT PROGRAM SOURCE FILE EXISTS                   &
!                                                                       &
        PROGRAM$ = PROGRAM$ + ".B2S" IF INSTR(1%, PROGRAM$, ".")=0%     &
                                        ! DEFAULT EXTEND IS .B2S       &

160     ON ERROR GOTO 170                                               &
\       OPEN PROGRAM$ FOR INPUT AS FILE 1%                              &
\       CLOSE 1%                                                        &
\       ON ERROR GOTO 0                                                 &
\       PGM$ = LEFT(PROGRAM$, INSTR(1%, PROGRAM$, ".")-1%)              &
\       GOTO 190                                                        &

170     RESUME 180                                                      &
180     PRINT                                                           &
\       PRINT "UNABLE TO FIND THE PROGRAM FILE -" ; PROGRAM$            &
\       ON ERROR GOTO 0                                                 &
\       GOTO 105                                                        &

190     RETURN                                                          &

!-----------------------------------------------------------------------&

200  !                                          G200                   &
!                                                                       &
!                                                                       &
!               CREATE COMMAND AND OVERLAY FILES                        &
!                                                                       &
!                                                                       &
!                                                                       &

210     OPEN PGM$+".ODL" AS FILE 1%             ! CREATE OVERLAY FILE   &
\       PRINT #1%, CHR$(9%) ; ".ROOT USER"                              &
\       PRINT #1%, "USER:" ; CHR$(9%) ; "FCTR SY:" ; PGM$ ; "-LIBR"     &
\       PRINT #1%, "LIBR:" ; CHR$(9%) ; ".FCTR LB:BP2COM/LB"            &
\       PRINT #1%, CHR$(9%) ; ".END"                                    &
\       CLOSE 1%                                                        &

220     OPEN PGM$+".CMD" AS FILE 1%             ! CREATE COMMAND FILE   &
\       PRINT #1%, PGM$ ; "=" ; PGM$ ; ".ODL/MP"                        &
\       IF NOT OPTIONS%(7%)                                             &
        THEN    PRINT #1%, "UNITS=12"                                   &
\               PRINT #1%, "ASG=SY:5:6:7:8:9:10:11:12"                  &
\               GOTO 250                                                &

230             PRINT #1%, "UNITS=14"           ! OPTION=XFILE          &
\               PRINT #1%, "ASG=SY:5:6:7:8:9"                           &
\               PRINT #1%, "ASG=SY:10:11:12:13:14"                      &

250     PRINT #1%, "RESLIB=LB:BASICS/RO"                                &
\       PRINT #1%, "//"                                                 &
\       CLOSE 1%                                                        &

290     RETURN                                                          &

!-----------------------------------------------------------------------&

300 !                                           G300                    &
!                                                                       &
!                                                                       &
!               CREATE INSTRUCTION TEXT                                 &
!                                                                       &
!                                                                       &
!                                                                       &

310     TEXT$ = "BP2" + CR$                                             &
\       TEXT$ = TEXT$ + "OLD " + PROGRAM$ + CR$                         &
\       TEXT$ = TEXT$ + "COM " + OPTION$ + CR$                          &
\       TEXT$ = TEXT$ + "TKB @" + PGM$ + ".CMD" + CR$                   &
\       TEXT$ = TEXT$ + "UNSAVE " + PGM$ + ".CMD" + CR$                 &
\       TEXT$ = TEXT$ + "UNSAVE " + PGM$ + ".ODL" + CR$                 &
\       TEXT$ = TEXT$ + "UNSAVE " + PGM$ + ".OBJ" + CR$                 &

390     RETURN                                                          &
                                                                        &
```

```
!-----------------------------------------------------------------------&

400 !                                           G400                    &
!                                                                       &
!                                                                       &
!               FORCE TEXT TO KEYBOARD                                  &
!                                                                       &
!                                                                       &
!                                                                       &

410     T.PNT% = 1%                                                     &

420     DUMMY$ = SYS(CHR$(6%) + CHR$(-4%) + CHR$(KEYBOARD%) +           &
                RIGHT(TEXT$, T.PNT%))                                   &
\       IF RECOUNT<>0                           !ADDITIONAL TEXT        &
        THEN    T.PNT% = (LEN(TEXT$)-RECOUNT) + 1%                      &
\               GOTO 420                                                &

490     RETURN                                                          &

!-----------------------------------------------------------------------&

500 !                                           G500                    &
!                                                                       &
!                                                                       &
!               OBTAIN PASSWORD                                         &
!                                                                       &
!                                                                       &
!                                                                       &

510     DUMMY$ = SYS(CHR$(6%) + CHR$(14%) + STRING$(20%,0%))            &
\       CHANGE DUMMY$ TO SYS1%                                          &
\       PROG% = SYS1%(7%)                                               &
\       PROJ% = SYS1%(8%)                                               &
\       PASS$ = RAD$(SYS1%(9%) + SWAP%(SYS1%(10%)))                     &
                + RAD$(SYS1%(11%) + SWAP%(SYS1%(12%)))                  &

590     RETURN                                                          &

!-----------------------------------------------------------------------&

600 !                                           G600                    &
!                                                                       &
!                                                                       &
!               RUN PSEUDO JOB                                          &
!                                                                       &
!                                                                       &
!                                                                       &

610     ON ERROR GOTO 620                                               &
\       PK.NUM% = 1%                                                    &

615 !                                                                   &
!               FIND AVAILABLE PSEUDO KEYBOARD                          &
!                                                                       &
        OPEN "PK"+NUM1$(PK.NUM%)+":" AS FILE 2%, MODE 0%                &
\       GOTO 640                                                        &

620     IF PK.NUM%=10%                                                  &
        THEN    PRINT "NO AVAILABLE PSEUDO KEYBOARDS FOR THE JOB"       &
\               ERROR.IND%=1%                                           &
\               GOTO 690                                                &

630     PK.NUM% = PK.NUM% + 1%                                          &
\       RESUME 615                                                      &

640     FIELD #2%, 128% AS MSG$                                         &

650 !                                                                   &
!               SEND COMMANDS TO BACKGROUND JOB                         &
!                                                                       &
        DUMMY% = FNSEND%("HELLO "+NUM1$(PROJ%)+","+NUM1$(PROG%)+        &
                ";"+PASS$, "Ready")                                     &
\       DUMMY% = FNSEND%("BP2", "BASIC2")                               &
\       DUMMY% = FNSEND%("OLD "+PROGRAM$, "BASIC2")                     &
\       DUMMY% = FNSEND%("COM "+PGM$+".OBJ"+" "+OPTION$, "BASIC2")      &
                IF NOT ERROR.IND%                                       &
\       DUMMY% = FNSEND%("TKB @"+PGM$+".CMD", "Ready")                  &
                IF NOT ERROR.IND%                                       &
\       DUMMY% = FNSEND%("EXIT", "Ready")                               &
                IF ERROR.IND%                                           &
\       DUMMY% = FNSEND%("UNSAVE "+PGM$+".CMD", "Ready")                &
\       DUMMY% = FNSEND%("UNSAVE "+PGM$+".ODL", "Ready")                &
\       DUMMY% = FNSEND%("UNSAVE "+PGM$+".OBJ", "Ready")                &
                IF NOT ERROR.IND%                                       &

660     CLOSE 2%                                                        &

690     RETURN                                                          &

!-----------------------------------------------------------------------&

700 !                                           G700                    &
!                                                                       &
!                                                                       &
!               SEND MESSAGE TO USER'S TERMINAL                         &
!                                                                       &
!                                                                       &
!                                                                       &

710     IF ERROR.IND%=-1%                                               &
        THEN    MESSAGE$ = "PROGRAM: "+PROGRAM$+" COMPILATION ERRORS "  &
                        +"-CHECK "+PGM$+".ERR"                          &
        ELSE    IF ERROR.IND%=0%                                        &
        THEN    MESSAGE$ = "PROGRAM: "+PROGRAM$+" NO COMPILATION ERRORS "&
                                                                        &
        ELSE    MESSAGE$ = "PROGRAM: "+PROGRAM$+                        &
                        " NO AVAILABLE PSEUDO KEYBOARDS"                &
```

```
720     DUMMY$ = SYS(CHR$(6%) + CHR$(-4%) + CHR$(KEYBOARD%) +
                MESSAGE$+CHR$(13%)+CHR$(10%))

790     RETURN

!-----------------------------------------------------------------------

1000 !                                          FNSEND
!
!
!               SEND AND RECEIVE STRINGS FORM PSEUDO TERMINAL
!
!                                       SEND.STRING$    - STRING TO BE SENT
!
!                                       END.STRING$     - END OF STRING TO BE
!                                                         RECEIVED
!
!

1000    DEF FNSEND%(SEND.STRING$, END.STRING$)
\       ERR.COUNTER% = 0%
\       OUT1$ = ""
\       LSET MSG$ = SEND.STRING$+CHR$(13%)
\       IF INSTR(1%, SEND.STRING$, "HELLO")<>0%
        THEN    REC.NUMBER% = 1%
        ELSE    REC.NUMBER% = 10%

1010 !
!               SEND STRING TO PSEUDO KEYBOARD
!
        ON ERROR GOTO 1030

1020    PUT #2%, RECORD REC.NUMBER%, COUNT LEN(MSG$)
\       GOTO 1050

1030    SLEEP 2
\       ERR.COUNTER% = ERR.COUNTER% + 1%
\       RESUME 1020 IF ERR.COUNTER%<10%
\       RESUME 1050                     ! SMALL BUFFERS ARE FULL

1050 !
!               RECEIVE STRING FROM PSEUDO KEYBOARD
!
        ON ERROR GOTO 1070

1060    INPUT LINE #2%, OUT$

1062    OUT3$ = LEFT(OUT$, RECOUNT)
\       OUT1$ = OUT1$ + OUT3$

1064    GOTO 1080 IF INSTR(1%, OUT1$, END.STRING$)<>0%
                                        ! CHECK FOR END OF STRING
\       GOTO 1060
\       RESUME 1060

1070    RESUME 1060

1080    IF INSTR(1%, OUT1$, "Error")<>0%
                OR INSTR(1%, OUT1$, "error")<>0%
                OR INSTR(1%, OUT1$, "?")<>0%
                OR INSTR(1%, OUT1$, "%")<>0%
                                        ! SET ERROR INDICATOR IF
                                        !  STRING CONTAINS AN ERROR
                                        !  MESSAGE
        THEN    OPEN PGM$+".ERR" AS FILE 3%
\               PRINT #3%, OUT1$
\               ERROR.IND% = -1%
\               CLOSE 3%

1090    FNEND

!-----------------------------------------------------------------------

32000   IF NOT OPTIONS%(1%)                     ! NOT DETACH
        THEN    PRINT
\               PRINT "END OF COMPIL"

32767   END
```



"You haven't been the same since the boss said you were a programming star."

THE HISTORY OF RSTS

By Peter Dick, Silver Programs, London

The history of RSTS starts in the second World War. There was a requirement to be able to train pilots without having them actually go up in aeroplanes. Project Whirlwind was the name given to the first electronic computer that was able to operate in real time; this meant it could be programmed to behave like an early flight simulator. The connection between Project Whirlwind and RSTS is that one of the people on that project was Ken Olsen, and the fact that the Whirlwind computer was a 16-bit computer.

Ken Olsen stayed with MIT until 1957 when he left and formed Digital Equipment Corporation. Folklore tells us that Ken Olsen started with three other people, one of whom was his brother, using their garage to assemble boards. The truth of course is slightly more boring, and that from the first day of operations they had the backing of a quarter of a million dollar loan.

The big breakthrough for computing was in 1961 when President Kennedy decided that the American people would land a man on the moon before the end of the decade. This decision gave computing in general an enormous source of funds; i.e., NASA.

In 1963 the PDP 6 was announced and on this machine it was possible to do timesharing. In 1964 BASIC was invented by Dartmouth College, New Hampshire — BASIC, of course, standing for Beginners All-purpose Symbolic Instruction Code. 1968 saw TSS/8 being released by Digital which was timesharing on the PDP 8 that in fact gave birth to RSTS. 1970 saw the release of the hardware, the PDP 11, which has become the most successful minicomputer in the history of computing.

RSTS 11 Version 1, 1970. It was never released and it was only tried in-house.

RSTS Version 2A, 1971. Again never released only tried in-house.

RSTS Version 2B, June 1971. This is the date from which Digital claims that RSTS was introduced. There are no known users.

RSTS Version 3A, February 1972. It was supported on the 11/20 which was the only PDP 11 available, and was Pre-SYSGENed.

It obviously worked on an unmapped machine, job max 8K words, swap max allowed for eight users. It was interesting to note that the 20K words were both BASIC and RSTS together.

RSTS Version 3B, May 1972. There are no known users.

RSTS Version 3C, June 1972. At long last we have the first recorded user, Bob Branton, at Southeastern State College. This user was in fact responsible for documenting the first complaint about RSTS, that under Version 3C the monitor was unable to write to a disk file from a terminal while reading that file from another terminal. Very simply, this was because there was no such thing in those days as UPDATE mode.

RSTS Version 4A, October 1972. Support was increased to include the newly announced 11/40 and 11/45. SYSGEN was under DOS and it was interesting to note that the CUSPS were built from paper tape at 10 cps. There were only three manuals in those days, the BASIC PLUS Language Manual, RSTS Users Manual, and the System Managers Manual. RECORD I/O was optional, and in fact there was a separate sub option for UPDATE mode which the book told us took an extra 75 words. There was talk of BADS for future releases. Hardware was fixed disks or RK05's. There was no memory management. The garbage collector was FIP overlayed and it wrote out the program and had to read it back in again. PI was not write protected; i.e., if you were tired of it being 3.142 you could make it 3.1,3,16,189. Maximum program size was 8K words but this could be PATCHed to 9K if you knew Rosemary Phillimore. Swap max was increased from the original eight users to 16. The program SHUTUP achieved the system closing down by PEEKing at an odd address which caused the machine to crash.

RSTS version 4B, contained no new code and was in fact just a fully

Intersil Systems

MEETS THE DEC® DH-11 CHALLENGE

The challenge; increased communications capacity, increased reliability, and improved technology - at decreased cost.

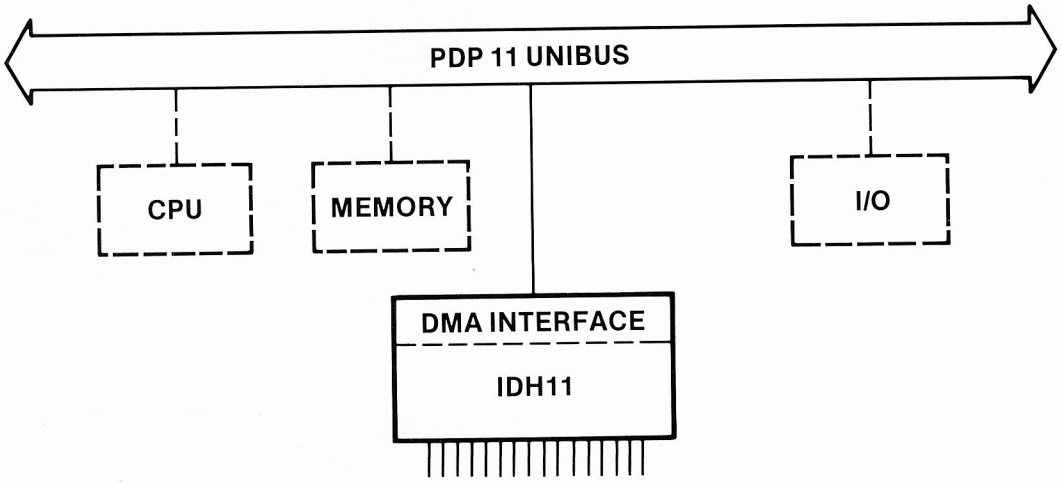
That's not a challenge for Intersil Systems.

- **Increased Capacity –** One Intersil DH-11 replaces nine DEC card slots
- **Increased Reliability –** Only one chance for a failure instead of nine.
- **Improved Technology –** The latest microprocessor based technology is employed.
- **Decreased Cost –** 66% less! One Intersil DH-11 costs 66% less than one DEC DH-11.

In addition the Intersil DH-11 offers these features:

- 16 Asynchronous local or remote channels on the UNIBUS®
- DMA output to free the CPU from Interrupt handling.
- On-board diagnostics.
- Complete software compatibility.

IDH11



16 ASYNCHRONOUS CHANNELS TO LOCAL OR REMOTE TERMINALS

Get all the challenging facts. Call (408) 743-4300, TWX: 910-339-9369, or write Intersil Systems, Inc., 1275 Hammerwood Avenue, Sunnyvale, CA 94086

® Trademarks of Digital Equipment Corporation

CIRCLE 171 ON READER CARD

PATCHed version of 4A. It was released in 1975 and marks the end of RSTS 11.

RSTS/E Version 5A/21, July 1973. It no longer supported the 11/20. SYSGEN was under DOS but at least you had batch command files to help. Job max was still 8K words for users but 16K for privileged users. Swap max had again been increased to allow for 32 users. The maximum machine size was 128K words, of which 124 was usable. Commercial Extensions were available which included Sort, IAM, QUE, SPOOL, Decimal Arithmetic package. Hardware included the RP03, which was the 40M byte disk drive, DH11s and 1200 LPM printers. APPEND as a command was introduced.

RSTS/E Version 5B/24, November 1974. It allowed for CCLs as long as you created them during SYSGEN time. EDIT and EDITCH were introduced. Auto answer was introduced for SYSGEN generations and so were multiple SWAP files. There was a hidden option for overlapped seek with multiple disks. Among the new BASIC PLUS commands were CVT$$, STRINGS$, STATUS, BUFSIZ. It was an extremely untidy release and so four months later . . .

RSTS/E Version 5C-01, March 1975. This in fact was a PATCHed version of Version 5B and contained no new commands.

RSTS/E Version 6A-02, August 1975. Support was now for the 11/34, 11/35, 11/40, 11/45, 11/60 and 11/70. The swap max had been increased to 63 users and maximum memory on the 11/70 was now 2M bytes. The RP02 and the RP04 were added, as was the concept of multiple run time systems; i.e., COBOL, FORTRAN, SORT11 were all added. XBUF was introduced. Pseudo keyboards. STATS was a hidden option. The VT50 was the replacement for the original VT05 and TU16's were available.

RSTS/E Version 6B-02, February 1977. SYSGEN was now under RT11. New hardware included RK05Fs, RK06, RP05, RP06, DZ11, LA180. Contiguous files were allowed. CCLs could be added at run time. EXTEND mode was introduced, Echo control mode 8. Among the run time systems allowed were BASIC PLUS 2, FORTRAN, COBOL, RPG2, APL, RMS-11, RT11 and of course yet another new BACKUP package.

RSTS/E Version 6C-03, February 1978. TECO and VTEDIT were released as part of a DECUS supported package. DECNET was introduced, DATATRIEVE, DIBOL/DECFORM. PIP.SAV replaced the old PIP and PIP EXTEND. Sexy lights were available. Also the program STATUS from Martin Minnow, which is still not supported by Digital. Hidden options included the words GET BLOCK and SPEC %.

RSTS/E Version 7.0, December 1979. Supported CPUs now included the 11/24, 11/34, 11/35, 11/40, 11/44, 11/45, 11/60, 11/70. RDC was introduced. SYSGEN was now under RSTS during time sharing and data caching was introduced. The RSX monitor allowed 31K word programs. Maximum memory on an 11/70 was now 4M bytes. Resident libraries were introduced. XBUF for line printers. SAV/RES for backup. A new version of FIP called large file FIP was created, and by now they had of course stolen the switch panel so that sexy lights were no longer possible.

RSTS/E Version 7.1, Februrary 1982. This now uses separate I & D space to relieve the small buffer problem. DECNET route through is allowed. TRACE, BREAK and DUMP are all hidden options.

RSTS/E Version 7.2, August 1982, supports new UDA50 disks; i.e., the RA80. The typical machine is now an 11/44, 1M byte of memory, RA80, TS11, DZ11E for under 50,000 pounds including a general licence.

RSTS/E Version 8.0, April 1983. Support for the recently announced PDP Micro 23. BASIC Version 2 now allows 31K words and no overhead for the use of RMS.

It is fair to say that the number of users has always been underestimated; Digital's own figues are as follows:

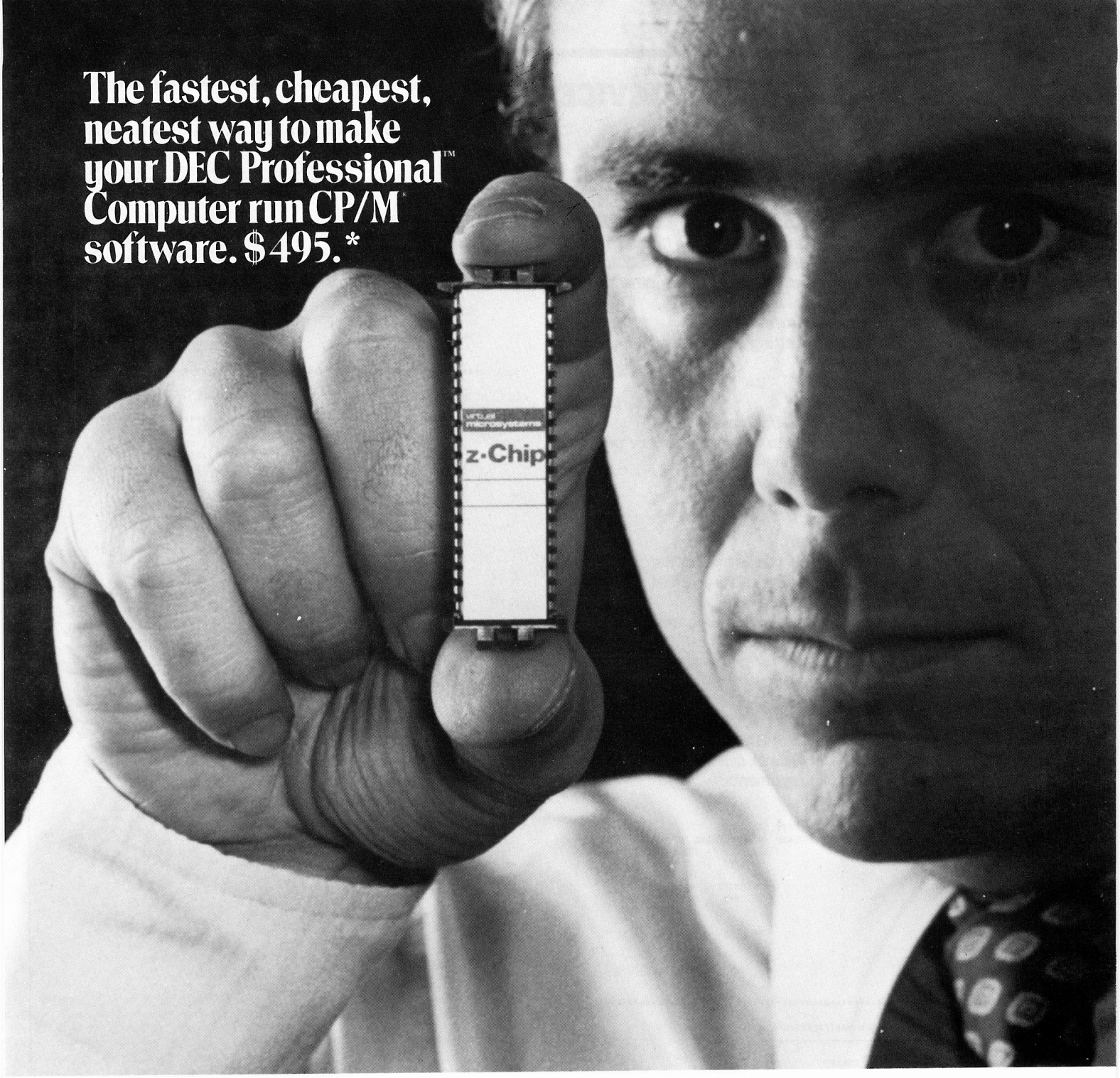
December 1973 — 150 users
August 1975 — 1,200 users
July 1977 — 3,100 users
December 1979 — 5,000 users

It is now thought that there are well over 10,000 licenced users and at least an equal number of unlicenced users!!!.

RSTS has a future, as Digital has yet to implement its greatest secret weapon, separate I & D space in terms of the application programs. It is interesting to note that RSX11M PLUS Version 2 allows the programmers to have separate 31K word areas for both I & D; in other words it should be possible under RSTS to have a total 62K word user area. RSTS remains the key commercial time sharing operating system that Digital sells. ♥

"I KNOW IT'S HARD TO BELIEVE, BUT IT IS THE NAME OF MY COMPUTER SYSTEM!"

The fastest, cheapest, neatest way to make your DEC Professional™ Computer run CP/M® software. $495.*



The z-Chip™. A single chip does what anyone else takes a board (and an expansion slot) to do.

With the Bridge and z-Chip, standard CP/M software will run on the DEC Professional (or any DEC computer based on the 11/23 chipset).

CP/M means access to some pretty potent, powerful, popular packaged software.

Packages like dBase II™, SUPERCALC™, and WORDSTAR®, to name a few.

Virtual Microsystems' Bridge™ family of integrated hardware and software tools means CP/M for just about any DEC system. From personal to VAX™.

And Virtual Microsystems means a single source for easy installation, support, software shopping (we inventory and sell all the best CP/M packages, on *all* DEC media) and service.

So if you'd like your DEC hardware to run the software that has made microcomputers *the* growth industry (without the headaches of dealing with different computer hardware, computer dealers, and computer formats) start with one simple phone call.

(415) 841-9594

Ask for Dave Scott.

CIRCLE 157 ON READER CARD

virtual microsystems

2150 Shattuck Ave., Berkeley, CA 94704

*The Bridge with z-Chip: $495 for the DEC Professional 350; $695 for the DEC 11/23™ and DEC Micro-11™. Trademarks: DEC Professional, VAX, 11/23, Micro-11, Digital Equipment Corporation; dBase II, Ashton-Tate; SUPERCALC, Sorcim Corporation; z-Chip and Bridge, Virtual Microsystems. Registered Trademarks: CP/M, Digital Research, Inc.; Wordstar; MicroPro International Corporation.

QUEUING TO A SPECIFIC DEVICE
QUEDEV.B2S

by Terry Ridgers, DuPont Canada, Corunna, Ontario

QUEDEV is written and compiled in BASIC+2. It can be called from any BASIC+2 main line program. The main line file name .ODL must contain QUEDEV i.e., .FCTR MAIN-LIBR-* (QUEDEV-LIBR, etc.).

QUEDEV when called and when passed valid arguments will queue files to any of the LPn:, BAn: and RJn: devices.

QUEDEV uses five passing arguments, namely:

1) SV.FIL$ - full file name as a string or as a string variable.
2) SV.DEV$ - Device name:
   must be of the form "LP", "BA", or "RJ".
3) SV.UNIT% - Unit number
   i.e., if sent to LP1: SV.UNIT% must = 1%.
4) SV.CPY% - Number of copies to be sent to the line printer
   (valid only for the line printer)
5) SV.DEL% - Delete list file after printing to LP:
   1% - yes
   0% - no
   (valid only for line printer)

NOTE: For unit number, number of copies and delete, the values passed must be integer values. If real values are passed then incorrect number of copies etc., may occur.

Examples of possible calls from the Main Program are:

1) CALL QUEDEV ("SY: (1,91)TEST.LST", "LP",0%, 2%, 1%)
Will queue 'SY:(1,91) TEST.LST" to LP: for two copies and will delete "SY:(1,91)TEST.LST" after listing the last copy.

2) SV.CTL$ = "DR1:(18,5)BATCH.CTL"
CALL QUEDEV(SV.CTL$, "BA", 1%, 0%, 0%)
This call will queue "DR1: (18,5) BATCH.CTL" to BA1: and it ignores the number of copies and will not delete the file regardless of what is passed as the last two arguments.

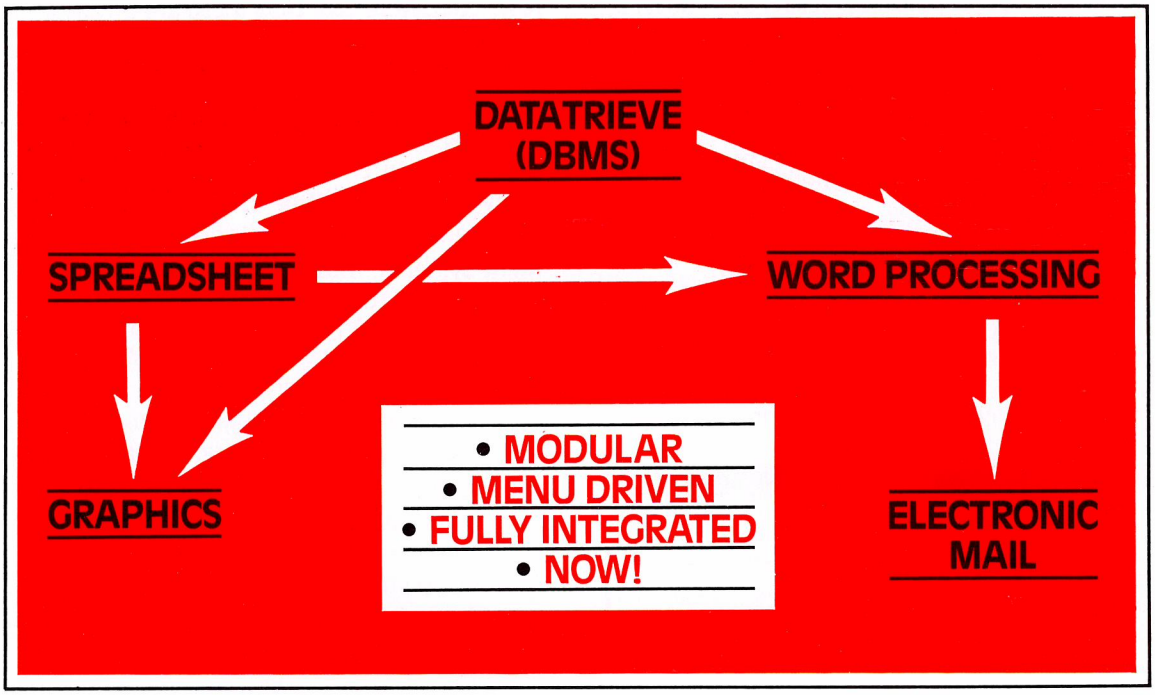
```
1 SUB QUEDEV (SV.FIL$,SV.DEV$,SV.UNIT%,SV.CPY%,SV.DEL%)
                              ! SUBPROGRAM  QUEUES FILES TO VARIOUS DEVICES&
                              !             SV.FIL$  - FILE NAME TO QUEUE  &
                              !             SV.DEV$  - DEVICE NAME         &
                              !                        IE: BA:,LP:,RJ:     &
                              !             SV.UNIT% - DEVICE UNIT #       &
                              !             SV.CPY%  - NO OF COPIES        &
                              !                        APPLIES TO LP:      &
                              !                         ONLY               &
                              !             SV.DEL%  - DELETE FILE AFTER   &
                              !                        1 - YES             &
                              !                        0 - NO              &
                              !                        APPLIES TO LP:      &
                              !                         ONLY               &

10 !*************************************************************** &
   !                                                                &
   ! SUB PROGRAM DESCRIPTION AND MODIFICATION LAYOUT                &
   ! ----------------------------------------------                 &
   !                                                                &
   ! SUB PROGRAM:              QUEDEV                               &
   ! ------------                                                   &
   !                                                                &
   ! PROGRAMMER:               T. RIDGERS                           &
   ! ----------                D U P O N T    CAN.                  &
   !                           ST. CLAIR RIVER WORDS                &
   !                           CORUNNA, ONT.                        &
   !                           NON 1G0                              &
   !                                                                &
   ! SUB-PROGRAM DESCRIPTION:                                       &
   ! ------------------------                                       &
   !                    THIS SUB PROGRAM CAN BE CALLED FROM ANY BASIC +  &
   ! 2 PROGRAM TO QUEUE FILES TO VARIOUS DEVICES.                   &
   !                                                                &
   ! MODIFICATION LOG:                                              &
   ! -----------------                                              &
   !                                                                &
   ! VERSION                DATE              DESCRIPTION           &
   !                                                                &
   !  001                 04-Aug-82          ORIGINAL RELEASE       &
   !                                                                &
   !  999                 DD-MMM-YY          XXXXXXXXXXXXXXXXXXXXXXXXXX  &
   !                                         XXXXXXXXXXXXXXXXXXXXXXXXXX  &
   !                                         XXXXXXXXXXXXXXXXXXXXXXXXXX  &
   !                                                                &
   !*************************************************************** &

1000!*************************************************************** &
    !                                                                &
    ! MAIN SUB-PROGRAM              1000 - 9999                      &
    !                                                                &
    !*************************************************************** &
   IF SEG$(SV.DEV$,1%,2%) <> "LP" THEN        ! ONLY LP: DEVICE CAN SPECIFY   &
                                  SV.DEL1% = 0%!          DELETE            &
 \                                GOSUB 10000  ! QUEUE FILE SV.FIL$ TO DEVICE  &
 \                                SUBEXIT      ! END OF SUB PROGRAM TO QUEUE TO&
                                               !  DEVICE                       &
   ELSE SV.DEL1% = 0%                          ! INITILIZE TO NOT DELETE        &
 \      FOR X% = 1% TO SV.CPY% - 1%           ! INCREMENT THRU # OF            &
                                               !  NON-DELETE COPIES TO LP:   & &
 \       GOSUB 10000                          ! QUEUE SV.FIL$ TO DEVICE BUT    &
                                               !  DON'T DELETE                 &
 \      NEXT X%                               ! NEXT NON-DELETE COPY?          &
 \      SV.DEL1% = 1%                         ! SET TO DELETE THIS COPY        &
                                               !  (WILL DELETE ONLY IF         &
                                               !   SV.DEL% IS ALSO 1%          &
 \      GOSUB 10000                           ! QUEUE SV.FIL$ TO DEVICE BUT    &
                                               !  THIS TIME DELETE FILE        &
 \      SUBEXIT                               ! END OF SUB PROGRAM TO QUEUE TO&
                                               !  DEVICE                       &

10000 !*************************************************************** &
      !                                                                &
      !  SUBROUTINE             - QUEUES FILE TO DEVICE               &
      !                                                                &
      !*************************************************************** &
   T.1$ = SYS(CHR$(  6%)                      +! SYS CALL TO FIP              &
              CHR$(-28%)                      +! SPOOL REQUEST CODE           &
              SPACE$(2%)                      +! NOT USED                     &
              SEG$(FSS$(SV.FIL$,1%),5%,12%)   +! FILE NAME STRING SCAN TO GET  &
                                               !  FILE EXTENSION              &
                                               !  IN RADIX 50 FORMAT          &
              SV.DEV$                         +! QUEUE TO DEVICE:             &
              CHR$(SV.UNIT%)                  +! AS UNIT                      &
              CHR$(0%)                        +! UNIT # FLAG                  &
              CHR$( 0%) + CHR$(0%)            +! MUST BE ZERO                 &
              CHR$(SV.DEL% * SV.DEL1% * 36%)  +! DELETE AND NO HEADER IF = 36 &
               CHR$(0%)                       +!                              &
              SPACE$(2%)                      +! NOT USED                     &
              SEG$(FSS$(SV.FIL$,1%),23%,26%)  +! FILE NAME STRING SCAN TO     &
                                               !  DEFINE DEVICE # AND UNIT #  &
                                               !  AND UNIT # FLAG WHERE FILE  &
                                               !  TO BE SPOOLED FROM          &
              SPACE$(4%))                      ! NOT USED                     &
    \ RETURN                                   ! END OF QUEUING TO DEVICE     &

32767 SUBEND                                   ! END OF QUEUING TO DEVICE     &
                                               ! SUB PROGRAM                  &
```

```
Line Numbers

#10000      1000:3    1000:7    1000:10


Variables

SV.CPY%       #1      1000:6

SV.DEL%       #1     10000

SV.DEL1%   @1000:2   @1000:5   @1000:9   10000

SV.DEV$       #1      1000     10000

SV.FIL$       #1     10000     10000

SV.UNIT%      #1     10000

T.1$       @10000

X%          @1000:6   1000:8


Built-in Functions

CHR$( )    10000     10000     10000     10000     10000     10000     10000
           10000

FSS$( )    10000     10000

SEG$( )     1000     10000     10000

SPACE$( )  10000     10000     10000

SYS( )     10000


Please check the use of the following:

T.1$


Subprogram name: QUEDEV
There were 14 identifiers listed,
and 39 references to them.
17 work-file entries (3 disk blocks) were used.
```

VOX

VAX OFFICE EXCHANGE

HAMILTON'S COMPLETE OFFICE SYSTEM



RENTAL OR SALE of VOX complete or by module with or without VAX hardware.

TIMESHARING VOX and full range of DEC layered products and software tools.

HARDWARE CONFIGURING All DEC systems and peripherals together with Hewlett Packard and Tektronix graphic devices, Dataproducts and letter quality printers, pretested and installed on your site with manufacturers' warranty and maintenance available.

HAMILTON THE HOUSE OF VAX

HGL Software Hamilton Rentals

6 Pearl Court, Allendale, NJ 07401
TOLL FREE 800-631-0298
In New Jersey 201-327-1444

415 Horner Ave, Toronto, M8W4W3
TOLL FREE Ont. & Que. 800-268-2106
All other prov. 800-268-0317
In Canada 416-251-1166

NEW YORK • DALLAS • MONTREAL • CALGARY • LONDON • PARIS • DUSSELDORF

CIRCLE 10 ON READER CARD

RSTS/E DISK OPTIMIZATION . . .

. . . continued from page 18

2). Recently created files are frequently accessed, so putting them at the front of the list decreases open time.

Closer inspection shows these arguments against using NFF.

1). In accounts with more than 31 files, it requires an extra UFD access (a physical write access), to create a new file. All of the name blockettes must be read to make sure that the file does not already exist. Then the pointers in both the first and the last name blockette must be rewritten. If NFF is not used, only the last name blockette must be rewritten. Note too, that these are physical accesses to disk that are not cached.

2). Recently created files are very likely to be deleted, resulting in two side effects. First, two name blockettes must be rewritten when the file is deleted, and they are likely to be in two separate UFD blocks. Without NFF, it is likely that they will be in the same UFD block. Second, in many environments such as word processing and development systems, most files are created, opened only once for reading, and then deleted. The main advantage of NFF, quick file opens, is minimized.

BENCHMARKS

Typical RSTS/E systems are difficult to objectively measure. System load varies from minute to minute and hour to hour. Monitor statistics are easy to gather but difficult to interpret (e.g., what does it mean if directory accesses decrease: less activity, a slower system, or a more efficient UFD structure?). One way to eliminate uncontrollable variables is to run a single job on an otherwise unused system and measure wall clock time. Unfortunately, single-user benchmarks are poor indicators of multi-user performance.

At the System Performance House, we have developed a set of 12 programs which run simultaneously, simulating a multi-user environment. They are heavily disk I/O bound programs which perform a mix of file creations, opens, closes, lookups, logins, logouts, swaps, run-time system loads, and random file accesses. The programs run for a fixed length of time and measure the number of each type of operation they can perform during the allotted time. Although they overstate the true performance differences from changes in disk I/O efficiency, they are very sensitive to small performance changes.

These benchmark programs vividly demonstrate the performance gains generated by the optimizations described above. The benchmark programs showed 50 percent more throughput on this arrangement than they did on the identical system generated by another commercially available "disk structuring" program. This increase was due to both faster disk accesses and fewer directory accesses.

REFERENCES

1. Banks, Scott. "RSTS Disk Directories," RSTS Professional, Vol. 1, No. 1; Vol. 2, No. 1; Vol. 2, No. 3; and Vol. 2, No. 4.

2. Mayfield, Mike. "RSTS/E Disk Internal Structures," Proceedings of the Digital Equipment Computer Users Society, April, 1978.

♥

WATCH and RINGME

By Maury Pepper and Greg Wenzel, STS, Inc., St. Louis, MO

We found that after submitting a task build to ATPK, programmers were frequently using SYSTAT to see if the job had terminated. The two programs included here were designed to increase productivity and ease the anxiety of those waiting for a job to finish.

Any job can be WATCH'd. When the job disappears from the job table (i.e., when it is killed or logged out), RINGME will send three bells (beep, beep, beep) to the terminal which originated the WATCH. We set up a CCL for calling WATCH

UT CCL WA-TCH=[1,2]WATCH.BAC;PRIV 0

The syntax is:

	WATCH jn	watch job number jn
or	WATCH/jn	watch job number jn
or	WATCH	program will prompt for job number

WATCH will determine which terminal you are on and then spawn a job to run RINGME — passing the terminal number and job number to RINGME in core common. Both programs should be saved with a <232> protection code. RINGME will run "logged-out" and detached. It's left as an unnecessary exercise for the purists to have RINGME log itself into the originator's account.

```
1       EXTEND
10      !       Program: WATCH.BAS                              &
        !       Author: Greg Wenzel/Maury Pepper                &
        !       Date: 28-Jan-83                                 &

100     JOBNO$ = CVT$$(RIGHT(SYS(CHR$(7%)),7%),-1%) ! Get Job # From CC &
\       GOSUB 1000      IF JOBNO$ = ''          ! Request Job #     &
\       GOTO 32767      IF JOBNO$ = ''          ! Exit If No Job #  &
\       ON ERROR GOTO 32767                     ! Set Error Trap    &
\       JOBNO% = VAL(JOBNO$)                    ! Validate Job #    &
\       GOTO 32767  IF JOBNO% < 1% OR JOBNO% > 64%  !               &

110     KB$ = MID(SYS(CHR$(6%)+CHR$(26%)+CHR$(0%)+CHR$(0%)),4%,1%)      &
            ! Determine Calling Keyboard                               &

120     I$ = SYS(CHR$(6%)+CHR$(-10%)+'SY:[1,2]RINGME.BAC')              &
            ! Convert Program File Spec For Job Spawn                  &

130     M$ = SYS(CHR$(6%)+CHR$(24%)+CHR$(0%)+CHR$(0%)+MID(I$,5%,8%)+    &
              CVT%$(JOBNO%)+KB$+'       '+MID(I$,23%,4%))              &
              ! Spawn "RINGME"                                         &

140     GOTO 32767                                  ! Exit              &

1000    INPUT "Watch Job # ";JOBNO$                 ! Request Job #     &
\       RETURN                                                          &

32767   END
```

```
1       EXTEND
10      !       Program: RINGME.BAS                             &
        !       Author: Greg Wenzel/Maury Pepper                &
        !       Date: 28-Jan-83                                 &

100     J$ = SYS(CHR$(7%))              ! Get Core Common       &

110     I% = ASCII(MID(J$,3%,1%))       ! Determine KB # For Broadcast  &

120     I$ = SYS(CHR$(6%)+CHR$(-13%)+CHR$(255%)+CHR$(-1%)+CHR$(-8%))    &
\       ON ERROR GOTO 140               ! Lower Priority to -8  &

130     SLEEP 3                                                 &
\       I$ = SYS(CHR$(6%)+CHR$(26%)+CHR$(CVT$%(J$))+CHR$(0%))  &
\       GOTO 130                        ! Loop until SYS call fails     &

140     ON ERROR GOTO 32767                                     &
\       M$ = SYS(CHR$(6%)+CHR$(-5%)+CHR$(I%)+CHR$(7%)+CHR$(7%)+CHR$(7%)) ,&
                                        ! Broadcast 3 Bells to Terminal &

32767   END
```

♥

MEMO
A COMPUTERIZED NOTE FILE

By Mark Gilmore, Data Processing Dept.

California State University at Long Beach, Long Beach, CA

This program is intended to be an aid to people who are constantly logged in and working on their RSTS systems, and who are also subject to frequent interruptions from bosses, users, or their own brilliant thoughts. MEMO will allow you to store your gems of wisdom (or your boss' demands) where they won't be forgotten (or can be "filed" indefinitely).

Memo numbers, once assigned, are never re-used (even though the associated memo may be deleted). This allows you to keep some sort of record based on memo number if you wish to make paper copies of everything. Memo numbers are assigned starting with number one and incrementing by one.

The program creates a file called MEMO.DAT on the disk specified by the variable SY.MEMO.FILE$ in line 1110. Change this variable to specify the disk you want the memo file created on. Once created, the file can be moved to any other disk on the system. The program will check all the disks, looking for a memo file that it has read/write access to. This disk scan routine is the only reason that MEMO must be privileged, and privileges are dropped immediately after this scan is completed. The program will not currently handle systems with more than 20 disks. To allow more, change the dimension statement in line 1010.

The MEMO.DAT file is built with a linked list structure. Records are added to the file as needed, but they are never deleted. Instead, they are added to the list of free (available) records. This means that the file can grow to almost any size (more memos than you'll want to keep anyway) but that it can never shrink (until someone writes a program to compress the file — this is left as an exercise for the reader).

It is possible to remove privileges by placing disk names to be scanned in a DATA statement. These names should be placed in the DISKS$ array and the variable DISKS% should be set to the number of disk names. Do this in line 1100 before the GOTO 15000, and delete lines 15010 — 15020.

If the disk scan feature is not wanted at all (the file specified in SY.MEMO.FILE$ is the only possibility) delete lines 15010 — 15060 and make the following changes:

```
15010       MEMO.FILE$ = SY.MEMO.FILE$
        \   OPEN MEMO.FILE$ FOR INPUT AS FILE #4%
        \   IF (STATUS AND 1024%) = 0% THEN 1110
                ELSE
                PRINT "Don't have write access to"; MEMO.FILE$
        \       GOTO 32767
                    ! If we can't write on the file forget it.

32010       IF ERL = 15010 THEN RESUME 20050
                    ! Memo file didn't exist — create it.
```

The PROMPT command allows the user to change the input prompt. This allows a little "customizing" for each user, and could be used to store an important note that the user wants to be reminded of the next time the program is run. The prompt may also be set to print escape sequences to allow the use of some of the special functions available on video terminals (note that this may have undesirable effects if you find yourself on a different terminal type than you anticipated). This feature may be disabled by removing the loop in lines 6020-6030 which converts CHR$(27)s to CHR$(155)s in the prompt.

For video terminals, it may be advantageous to modify the PROMPT command to allow the user to specify several "set-up" sub-prompts. These could include (1) a set-up string to be printed when the program is first run; (2) an input prompt (available now); (3) a string to print before the response to a command; and (4) a string to reset the terminal at the end of execution. For example, this would allow the VT100 user to define the lower part of his screen as a scrolling region for program responses, print a double-width "Function?" at the top of the screen, move to the scrolling region to see the response, and reset the terminal when he exits.

Another feature of the program is that the instructions are printed automatically the first time the program is run. This is possible quite simply due to the fact that the first time the program is run there is no MEMO.DAT file. This feature could be considered for inclusion in any program that creates a data file (or anything else unique to the first run).

While this program has been tested, the disclaimer in the program text should be taken seriously. I am interested in hearing about problems or suggested changes, but I will not promise to do anything about them. Finally, if you find ways to make this program more efficient, please feel free to do so. The inefficiencies are there so that you may have the enjoyment of finding and fixing them. (Thus the larger the inefficiency, the greater my generosity in leaving it for you to find.)

A LIST OF FUNCTIONS

Add — Adds a memo to your memo file.

Delete — Removes a previously added memo.

List — Lists all memos in the file.
Prints entry date and subject.

Type — Prints the contents of one or more memos.
Several options are available:

Type	Prints a memo. Program asks which one.
Type 10	Prints memo 10.
Type *	Prints all memos.
Type STRING	Prints all memos with the specified STRING in the subject field.

An output file may be specified as follows:
Type * > Filename

Search — Lists memos with a specific string in their subject fields.

More — Appends more information to an existing memo.

Prompt — Changes the current prompt to something else.

Help — Gives some basic help to the user.

Exit — Leaves the program.

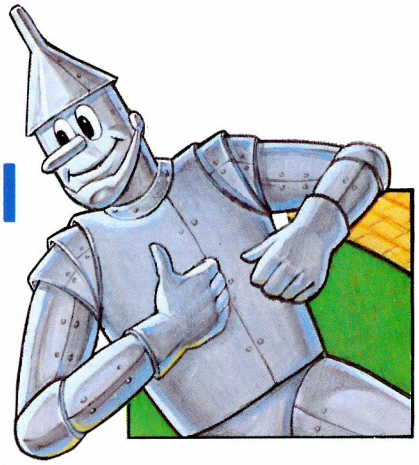
A memo number may be specified with the Type, Delete and More commands.

```
100     !
        !               M E M O
        !
        !       A Computerized Note File
        !               by Mark Gilmore
        !                       based on an idea by John Sandhoff
        !

110     !       (C) Copyright 1982, by Mark Gilmore. No part of this
        !       publication may be reproduced, transmitted, tran-
        !       scribed, stored in an information storage and retrieval
        !       system, or translated into any language or computer
        !       language, in any form or by any means, including, but
        !       not limited to, electronic, mechanical, magnetic,
        !       optical, chemical, manual or otherwise, without the
        !       inclusion of this copyright notice, except in the
        !       case of brief quotations embodied in critical articles
        !       or reviews.
        !
        !       The author makes no representations or warranties
        !       with respect to the correctness of the contents hereof
        !       and specifically disclaims any implied warranties of
        !       merchantability or fitness for any particular purpose,
        !       or that the software described herein will operate as
        !       documented on any computer of any manufacture with
        !       any operating system at any time. Further, the author
        !       reserves the right to revise this publication and to
        !       make changes from time to time in the content hereof
        !       without obligation to notify any person of such
        !       revision or changes.
        !
        !       This software may be freely distributed with the
        !       inclusion of this copyright notice, through any
        !       non-profit means of distribution. Distribution
        !       of this software for profit is expressly prohibited
        !       without the written consent of the author.
        !
        !       All characters in this book (with the exception of
        !       the author) are fictitious. Any resemblance to actual
        !       persons, living or dead,is purely coincidental.
        !

120     !       This program allows users to have the computer keep
        !       track of notes that they may wish to write to
        !       themselves, acting as a computerized note pad or
        !       filing box.
        !
        !       Memos are stored in a linked list. New records are
        !       added to the memo file as needed, and the file may
        !       grow to any size. However there is no PACK function
        !       in this version, so deleting memos from the file
        !       will not shrink the file, but will result in records
        !       being added to the free list to be reallocated later.
        !
        !       One feature of the PROMPT command which may be
        !       useful is that it can be used to store a very
        !       important memo that you may wish printed every
        !       time you run the program (to 490 characters).

130     !                       VARIABLES
        !
        !       A$              General use string
        !       A.NUM%          Memo number to add to (MORE)
        !       BUF$            FIELD to store prompt string
        !       CALL%           Returns value of function call
        !       CALL$           Target string for SYS call
        !       CALL2%          Returns value of function call
        !       CHOICE%         Users choice of function
        !       CRLF$           Carriage return/Line feed combination
        !       CUR.MEMO$       Stores memo number currently listing
        !       DELNUM%         Memo number to delete
        !       DEV$            Device name for disk lookup
        !       DEVCNT%         Table of maximum unit numbers on system
        !       DEVNAM%         Device name table
        !       DEVOKB%         Number of disk devices
        !       DISKS%          Index into DISKS$ table
        !       DISKS$ (20)     Table of disk names
        !       F%              Flag for output file specified
        !       F.LINK$         FIELD - Link to next record in list
        !       FIRST%          Holds first record num in FNFIRST.REC%
        !       FIRST.RUN%      Flag to print help for first-time user
        !       FOUND%          Contains result of SEARCH function
        !       FQB% (30)       Returned values from SYS call
        !       FR.REC%         Free record found by FNUNLINK
        !       FREE%           FREE - constant to link pointers
        !       HEAD%           Flag for header printed in SEARCH
        !       I%              Scratch variable
        !       INIT$           Field used to initialize MEMO.DAT file
        !       JUNK%           Scratch variable
        !       LAST%           Last record number (in functions)
        !       LINKAFTER%      Record to link new data after (FNLINK%)
        !       M%              Flag for memo number specified in TYPE
        !       M.NUM$          Memo number to add to file in ADD
        !       MAX.MEMO$       FIELD - Highest memo number used
        !       MEM.NUM%        Memo number from command
        !       MEMO$           The memo to be placed in the file
        !       MEMO.DAT$       FIELD - MEMO data field (the memo text)
        !       MEMO.DATE$      FIELD - The date the memo was filed
        !       MEMO.FILE$      Name of memo file
        !       MEMO.LINE$      A line of data to add to the memo
        !       MEMO.NUM$       FIELD - Memo number in disk record
```

```
!       MEMO.SEQ$       FIELD - Next available memo number
!       MEMO.SPEC$      Memo number/string/* in TYPE
!       MEMO.UNUSED     FIELD - unused area in MEMO base rec
!       ND%             (END) - constant to link pointers
!       NEW.PROMPT$     The user-specified new prompt
!       NULL.LINK$      Used to indicate end of linked list
!       NXT.PTR$        Pointer to next record
!       NXT.REC%        Next record number (FNNEXT.REC%)
!       NXT.REC$        Pointer to next record
!       OLDCALL%        Save previous value of CALL%
!       P.NUM%          Number of memo to print
!       PART.OF.THE.NEW.PROMPT$         A line of new prompt
!       PREV.REC%       Previous record number (FNPREV.REC%)
!       PRINT.FILE$     Output file for TYPE function
!       PROMPT$         Prompt for user input
!       PROMPT.LEN$     FIELD - Length of PROMPT$ in file
!       PRV.PTR$        Pointer to previous record
!       PTR$ (4)        FIELD - List pointers
!                               (free/used; start/end)
!       R.LINK$         FIELD - Link to prev rec (reverse link)
!       R.TYPE$         FIELD - Record type (free/used)
!       REC%            Record number passed to functions
!       REQUEST$        User input
!       SEARCH$         String to search memo subjects for
!       START%          START - constant to link pointers
!       SUB.ADD%        True if ADD being used as subroutine
!       SUB.SEARCH%     True if SEARCH being used as subroutine
!       SUB.TYPE%       True if TYPE being used as subroutine
!       SUBJECT$        Subject of memo
!       SY.MEMO.FILE$   Name of memo file on system disk
!       TEMP.MEM$       Temp storage for SEARCH
!       THIN.AIR$       Record number to create in FNUNLINK%
!       TODAYS.DATE%    Obvious
!       TYPE%           Record type - FREE or USED
!       TYPE.FILE$      File to TYPE memos to
!       UNUSED.FOR.NOW$ FIELD - Anything you want
!       USED%           USED - pointer to link pointers
!       VER.DATE$       Date of current version
!       VERSION$        Current version/revision data
!
!                       FUNCTIONS
!
!       FNFIRST.REC%    Returns first record in a list
!       FNLAST.REC%     Returns last record in a list
!       FNLINK%         Links a record into a list
!       FNNEXT.REC%     Returns next record in a list
!       FNPREV.REC%     Returns previous record in a list
!       FNUNLINK%       Unlinks a record from a list

1010    DIM PTR$(4%), FQB%(30), DISKS$(20)
      \ USED% = 0%
      \ FREE% = 2%
      \ START% = 1%
      \ ND% = 2%
      \ NULL.LINK$ = CHR$(0%) + CHR$(0%)
      \ CRLF$ = CHR$(13%) + CHR$(10%)
      \ FIRST.RUN% = 0%
      \ TODAYS.DATE% =
            SWAP%(CVT$%(MID(SYS(CHR$(6%)+CHR$(-3%)),27%,2%)))

1100    PROMPT$ = "Function? "
      \ VERSION$ = "1b "
      \ VER.DATE$ = "Nov 82"
      \ TYPE.FILE$ = "_KB:MEMO.KBD"
      \ SY.MEMO.FILE$ = "_DB0:MEMO.DAT"

      \ ON ERROR GOTO 32000
      \ OPEN TYPE.FILE$ FOR INPUT AS FILE #1%
      \ GOTO 15000
                                        ! Set up some defaults
                                        ! and go find the data file.

1110    FIELD #4%,    2%   AS PTR$(1%),       ! Used start pointer
                      2%   AS PTR$(2%),       ! Used end pointer
                      2%   AS PTR$(3%),       ! Free start pointer
                      2%   AS PTR$(4%),       ! Used end pointer
                      2%   AS PROMPT.LEN$,    ! Length of prompt
                      2%   AS MAX.MEMO$,      ! Highest record counter
                      2%   AS MEMO.SEQ$,      ! Memo sequence number
                      8%   AS UNUSED.FOR.NOW$,
                      490% AS BUF$
                                        ! FIELDS for record 1
                                        ! (main pointer block)

      \ FIELD #4%,    2%   AS F.LINK$,        ! Forward link
                      2%   AS R.LINK$,        ! Reverse link
                      1%   AS R.TYPE$,        ! Record type
                      2%   AS MEMO.NUM$,      ! Memo number &
                      2%   AS MEMO.DATE$,     ! Memo date &
                      3%   AS MEMO.UNUSED$,   ! Unused
                      500% AS MEMO.DAT$       ! Memo data
                                        ! FIELDS for data records

      \ GET #4, RECORD 1%
      \ IF CVT$%(PROMPT.LEN$) > 0% THEN
            PROMPT$ = LEFT(BUF$, CVT$%(PROMPT.LEN$))

1120    PRINT
      \ PRINT "M E M O      ";
      \ PRINT RIGHT(SYS(CHR$(6)+CHR$(9)+CHR$(0)),3)
      \ PRINT "Version ";VERSION$;"  ";VER.DATE$
      \ PRINT
      \ GOTO 11000 IF FIRST.RUN%

1200    PRINT
      \ CLOSE 1%
      \ OPEN TYPE.FILE$ AS FILE #1%           ! Re-open keyboard for input
      \ A.NUM%, P.NUM%, DELNUM% = 0%          ! Zero memo nums (Type, Delete, More)
      \ SUB.ADD%, SUB.SEARCH%, SUB.TYPE% = 0%
      \ PRINT PROMPT$;
      \ INPUT LINE #1, REQUEST$
      \ REQUEST$ = CVT$$(REQUEST$, 8% +4%)    ! Dump leading spaces and CR/LF.
      \ IF LEFT( CVT$$( REQUEST$, -1%), 1%) = "T" THEN 13000
                                        ! Check fancy type functions.

1210    MEM.NUM% = VAL(RIGHT(REQUEST$,INSTR(0%,REQUEST$," ")))
                                        ! Get memo number if given.
      \ A.NUM%, P.NUM%, DELNUM% = MEM.NUM%

1220    REQUEST$ = LEFT(CVT$$(REQUEST$,-1%),1%)
      \ GOTO 1200 IF REQUEST$ = ""            ! No function - try again
      \ CHOICE% = INSTR(0%, "ADELMPTHS", REQUEST$) + 1%
      \ ON CHOICE% GOTO
            12000, 2000, 3000, 21000, 5000, 9000, 6000, 7000, 11000, 8000

2000
!       Add a memo to the file

2005    IF A.NUM% <> 0% THEN
            PRINT "I choose the memo number."
      \     GOTO 1200

2020    SUB.ADD% = 0%                           ! Normal entry.
      \ PRINT
      \ PRINT "Subject: ";
      \ INPUT LINE #1, SUBJECT$               ! Get the memo subject.

2030    PRINT "Now type your memo. End with Ctrl-Z."
      \ MEMO$ = SUBJECT$ + CHR$(3%)           ! Choose a good delimiter.
      \ PRINT
      \ WHILE 1%

2035        GET #1%
      \     FIELD #1%, RECOUNT AS MEMO.LINE$
      \     MEMO$ = MEMO$ + MEMO.LINE$        ! Get the memo data
      \ NEXT                                  ! End on Ctrl-Z error

2040    RETURN IF SUB.ADD%                      ! Get out if subroutine.
      \ GET #4%, RECORD 1%                    ! Get the main pointers.
      \ M.NUM$ = MEMO.SEQ$ + ""               ! Get memo sequence number.
      \ LSET MEMO.SEQ$ =
            CVT$$( CVT$%( MEMO.SEQ$)+1%)      ! Increment the seq. num.
      \ PUT #4%, RECORD 1%                    ! And replace the pointers.

2050    CALL% = FNUNLINK%( FREE%, 0%)           ! Get a free record.
      \ GOTO 2090 IF CALL% = 0%               ! Complain if none free.
      \ GET #4%, RECORD CALL%                 ! Get the free record.
      \ LSET MEMO.NUM$ = M.NUM$               ! Store the message number.
      \ LSET MEMO.DATE$ = CVT$$(TODAYS.DATE%)! Store the date.
      \ LSET MEMO.DAT$ = LEFT(MEMO$,500%)     ! Store some memo data.
      \ MEMO$ = RIGHT(MEMO$, 501%)            ! Trim down the memo.
      \ PUT #4, RECORD CALL%                  ! Write the record to the file.
      \ CALL% = FNLINK%( USED%, CALL%, 0%)    ! Link the record in the used queue.
      \ IF LEN(MEMO$) > 0% THEN 2050          ! If more memo, repeat.

2060    PRINT
      \ PRINT "Memo "; CVT$%( M.NUM$); " filed."
      \ GOTO 1200

2090    PRINT
      \ PRINT "? No FREE record available"
      \ PRINT                                 ! Should always be a free
      \ GOTO 21000                            ! record available.
                                              ! Bad error if not.
3000  !
!       Delete a memo
!

3010    PRINT
      \ CALL2% = 0%
      \ PRINT "Delete memo number: ";
                        UNLESS DELNUM% &
      \ INPUT #1%, DELNUM% UNLESS DELNUM%     ! Get memo num if not given.
      \ GOTO 3050 IF DELNUM% < 1%             ! Exit if bad number.
      \ GET #4%, RECORD 1%                    ! Check against high number.
      \ GOTO 3050 IF DELNUM% >= CVT$%( MEMO.SEQ$)
                                              ! Exit if number too high.

3020    CALL% = FNFIRST.REC%( USED%)            ! Start at the beginning.
      \ CALL% = FNNEXT.REC%
                  UNTIL MEMO.NUM$ = CVT$%( DELNUM%)
                        OR CALL% = 0%
                                              ! Find the record or end.
      \ GOTO 3050 IF CALL% = 0%               ! Get out if end.
      \ CALL2% = FNUNLINK%( USED%, CALL%)     ! Unlink the record.
      \ GOTO 3020 UNLESS CALL2% = 0%          ! Repeat for more records.
                  OR CALL2% <> CALL%          ! ...unless there was an error.

3030    IF CALL2% = 0% THEN
            PRINT "? Couldn't unlink record ";CALL%
        ELSE
            PRINT "? Deleted record ";CALL2%;" not ";CALL%

3040    PRINT
      \ GOTO 1200

3050    IF CALL2% <> 0% THEN
            PRINT "Memo "; DELNUM%; " deleted."
        ELSE
            PRINT "Can't find memo ";DELNUM%
                                              !  Print the results.

3060    GOTO 1200

5000
!       List any pending memos.

5005    IF A.NUM% <> 0% THEN
            PRINT "Show some guts - list them all."
      \     GOTO 1200

5010    CALL% = FNFIRST.REC%( USED%)            ! Goto first used record.
      \ GOTO 5050 IF CALL% = 0%               ! If none...no memos.
      \ PRINT
      \ PRINT "Memo", "  Date", "Subject"
      \ PRINT

5020    CUR.MEMO$ = MEMO.NUM$ + ""              ! Copy the memo number.
      \ PRINT CVT$%(MEMO.NUM$),               ! Print the information.
                  DATE$(CVT$%(MEMO.DATE$)),
                  LEFT(MEMO.DAT$,INSTR(0,MEMO.DAT$,CHR$(3))-1%);
      \ CALL% = FNNEXT.REC%
                  UNTIL MEMO.NUM$ <> CUR.MEMO$
                        OR CALL% = 0%
                                              ! Find next memo.
      \ GOTO 5020 IF CALL% <> 0%              ! And print it.

5030    PRINT
      \ PRINT
      \ GOTO 1200                             ! All done.

5050    PRINT
      \ PRINT "No memos on file."
      \ GOTO 5030                             ! Nothing to print.
                                              ! SEARCH func comes here, too.
```

PERSONAL and PROFESSIONAL

Produced for the Digital Personal and Professional Computer User



Each issue of PERSONAL and PROFESSIONAL is packed with the latest information on the world of personal computers in a high-quality, modern, attractive and easy-to-read format. The finest editors, writers, art directors, illustrators, cartoonists and photographers have been assembled to establish a new level of excellence and objectivity in personal computer magazines.

Simply stated, PERSONAL and PROFESSIONAL is the most substantial source on the Digital personal computer user's regular reading list. It's as if a team of computer experts came to your office or home every issue.

PERSONAL and PROFESSIONAL is an independent magazine, not sponsored or approved by or connected in any way with Digital Equipment Corporation.

Charter Subscription Offer

Save More Than 50% Off Cover Price

You can now become a Charter Subscriber to PERSONAL and PROFESSIONAL at absolutely No Risk.

Simply enter your subscription with payment enclosed and Save Over 50% Off The Cover Price. If not satisfied after receiving the first issue, merely return your mailing label within 15 days for a Full Refund. Offer good until July 1, 1983, and only applies to the US and Canada.

SO ACT NOW. Become a PERSONAL and PROFESSIONAL Charter Subscriber today!

PERSONAL and PROFESSIONAL

P.O. BOX 114, SPRINGHOUSE, PA 19477-0114
215/542-7008

NAME _____
(please print full name)

ADDRESS _____ APT. _____

CITY _____ STATE _____ ZIPCODE _____

☐ Bill me $28 for 12 issues, a 33% savings off the cover price. Offer good until July 1, 1983. Rates are double outside the US and Canada.

☐ Payment enclosed. YES, I want to take advantage of your special Charter Subscription Offer. Send me the first 12 issues of PERSONAL and PROFESSIONAL for only $20 (please remit in US currency), saving me over 50% off the cover price. Offer good until July 1, 1983. Charter Rate only applies to the US and Canada.

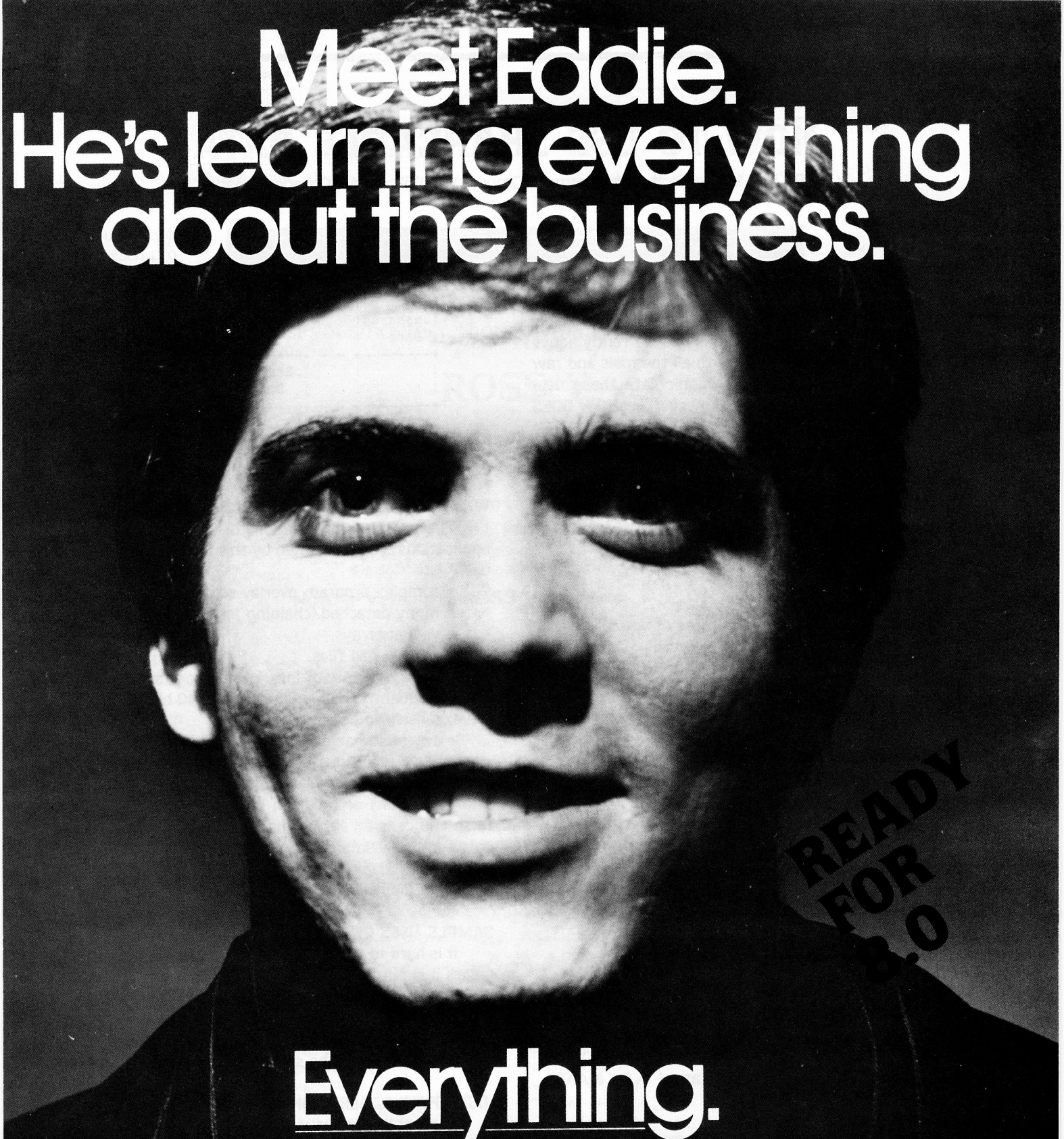
CIRCLE 177 ON READER CARD

```
6000    !
        !       Change the prompt to something else.
        !

6005        IF A.NUM% <> 0% THEN
                PRINT "Change the prompt in a memo? Ridiculous!"
        \       GOTO 1200

6010        PRINT
        \   PRINT "Enter your new prompt, ending with Ctrl-Z."
        \   PRINT
        \   NEW.PROMPT$ = ""
        \   WHILE 1%
        \       GET #1%
        \       FIELD #1%, RECOUNT AS PART.OF.THE.NEW.PROMPT$
        \       NEW.PROMPT$ = NEW.PROMPT$
                            + PART.OF.THE.NEW.PROMPT$
        \       GOTO 6020 IF LEN(NEW.PROMPT$) > 490%
        \   NEXT

6020        GET #4%, RECORD 1%
        \   NEW.PROMPT$ = LEFT(NEW.PROMPT$,490%)
        \   FOR I% = 1% TO LEN(NEW.PROMPT$)
        \       IF ASCII( MID( NEW.PROMPT$, I%, 1%)) = 27%
                    THEN
                    NEW.PROMPT$ = LEFT( NEW.PROMPT$, I%-1%)
                            + CHR$(155%)
                            + RIGHT( NEW.PROMPT$, I%+1%)
                                            ! Set parity on escapes.

6030        NEXT I%
        \   LSET BUF$ = NEW.PROMPT$ + STRING$(407%-LEN(NEW.PROMPT$),0%)
        \   LSET PROMPT.LEN$ = CVT%$(LEN(NEW.PROMPT$))
        \   PUT #4%, RECORD 1%
        \   PROMPT$ = NEW.PROMPT$
        \   GOTO 1200

7000    !
        !       Print a memo
        !

7010        PRINT
        \   SUB.TYPE% = 0%

7012        PRINT "Which memo? "; UNLESS P.NUM%
        \   INPUT #1%, P.NUM% UNLESS P.NUM%
        \   IF SUB.TYPE% THEN
                IF F% THEN
                    OPEN PRINT.FILE$ FOR OUTPUT AS FILE #1%
                ELSE
                    OPEN TYPE.FILE$ FOR OUTPUT AS FILE #1%
                                            ! Open the file if specified.

7015        GOTO 7050 IF P.NUM% < 1%        ! Ask for number - quit if bad.
        \   GET #4%, RECORD 1%              ! Check against high number.
        \   GOTO 7050 IF P.NUM% >= CVT$%( MEMO.SEQ$)
        \   CALL% = FNFIRST.REC%( USED%) &
        \   CALL% = FNNEXT.REC%
                    UNTIL MEMO.NUM$ = CVT%$( P.NUM%)
                        OR CALL% = 0%
                                            ! Find matching memo or end.
        \   GOTO 7050 IF CALL% = 0%         ! Complain if none.

7020        GOSUB 7100 IF INSTR(0%, MEMO.DAT$, CHR$(3))
                                            ! Print header if first rec.
        \   PRINT #1%, CVT$$(RIGHT(MEMO.DAT$,INSTR(0,MEMO.DAT$,CHR$(3))+1%),128%);
        \   CALL% = FNNEXT.REC%
        \   GOTO 7020 IF MEMO.NUM$ = CVT%$( P.NUM%)
                    UNLESS CALL% = 0%
        \   RETURN IF SUB.TYPE%
        \   PRINT
        \   GOTO 1200

7050        PRINT #1%
        \   PRINT #1%, "Memo "; P.NUM%; " not found."
        \   PRINT #1%
        \   RETURN IF SUB.TYPE%
        \   GOTO 1200

7100        PRINT #1%
        \   PRINT #1%
        \   PRINT #1%, "Memo "; P.NUM%, "Date "; DATE$(CVT%$(MEMO.DATE$))
        \   PRINT #1%, "Subject: ";
        \   PRINT #1%, LEFT( MEMO.DAT$, INSTR( 0%, MEMO.DAT$, CHR$(3%)) -1%)
        \   PRINT #1%
        \   RETURN

8000
        !       SEARCH Function


8005        IF A.NUM% <> 0% THEN
                PRINT "Why search if you know the memo number?"
        \       GOTO 1200

8010        PRINT
        \   SUB.SEARCH% = 0%
        \   HEAD% = 0%
        \   PRINT "Search for: ";
        \   INPUT LINE #1%, SEARCH$
        \   CALL% = FNFIRST.REC%( USED%)
        \   GOTO 5050 IF CALL% = 0%             ! Exit if no memos.

8015        SEARCH$ = CVT$$( SEARCH$, 32% + 4%)

8020        I% = INSTR(0%, MEMO.DAT$, CHR$(3%))
        \   GOTO 8030 IF I% = 0%
        \   TEMP.MEM$ = CVT$$( LEFT( MEMO.DAT$, I%), 32% +4%)
        \   FOUND% = INSTR(0%, TEMP.MEM$, SEARCH$)
        \   GOTO 8030 UNLESS FOUND%
        \   RETURN IF SUB.SEARCH%
        \   IF HEAD% = 0% THEN
                PRINT #1%, "Memo", "  Date", "Subject"
        \       HEAD% = 1%

8025        PRINT #1%, CVT$%(MEMO.NUM$),
                       DATE$(CVT$%(MEMO.DATE$)),
                       LEFT(MEMO.DAT$, INSTR(0%, MEMO.DAT$, CHR$(3))-1%);

8030        CALL% = FNNEXT.REC%
        \   GOTO 8020 IF CALL% > 0%

8050        RETURN IF SUB.SEARCH%
        \   PRINT
        \   GOTO 1200

9000
        !       Append to a memo


9010        PRINT
        \   PRINT "Add to which memo? "; UNLESS A.NUM%
        \   INPUT #1, A.NUM% UNLESS A.NUM%      ! Get the memo number.
        \   GOTO 9070 IF A.NUM% < 1%            ! See if too small.
        \   GET #4%, RECORD 1%
        \   GOTO 9070 IF A.NUM% >= CVT$%(MEMO.SEQ$) ! See if too large.
        \   CALL% = FNFIRST.REC%( USED%)
        \   CALL% = FNNEXT.REC%
                    UNTIL MEMO.NUM$ = CVT%$( A.NUM%)
                        OR CALL% = 0%
                                            ! Find the first record of
                                            ! ...the memo to append to
                                            ! ...or end of the list.
        \   GOTO 9070 IF CALL% = 0%         ! Get out if not found.
        \   WHILE MEMO.NUM$ = CVT%$( A.NUM%)    ! While reading the same memo.
                        AND CALL% <> 0%
        \       OLDCALL% = CALL%
        \       CALL% = FNNEXT.REC%         ! Get the next record.
        \   NEXT                            ! Now have last record of memo.
        \   MEMO$ = CRLF$ + "***** Appended "+DATE$(0%)+" *****"+CRLF$
                                            ! Set up append notice.
        \   SUB.ADD% = 1%                   ! Flag add as a subroutine.
        \   PRINT
        \   PRINT "Type your addition. End with Ctrl-Z."
        \   GOSUB 2035                      ! Get the addition.

9030        CALL% = FNUNLINK%( FREE%, 0%)   ! Get a free record.
        \   GOTO 9050 IF CALL% = 0%         ! Complain if none free.
        \   GET #4%, RECORD CALL%           ! Get the free record.
        \   LSET MEMO.NUM$ = CVT%$(A.NUM%)  ! Store the message number.
        \   LSET MEMO.DATE$ = CVT%$(TODAYS.DATE%)! Store the date.
        \   LSET MEMO.DAT$ = LEFT(MEMO$,500%)   ! Store some memo data.
        \   MEMO$ = RIGHT(MEMO$, 501%)      ! Trim down the memo.
        \   PUT #4, RECORD CALL%            ! Write the record to the file.
        \   CALL% = FNLINK%( USED%, CALL%, OLDCALL%) ! Link the record in the used queue.
        \   IF LEN(MEMO$) > 0% THEN         ! If more memo, repeat.
                OLDCALL% = CALL%
        \       GOTO 9030

9040        PRINT
        \   PRINT "Memo "; A.NUM%; " updated."
        \   SUB.ADD% = 0%
        \   GOTO 1200

9050        PRINT
        \   SUB.ADD% = 0%
        \   PRINT "? No FREE record available"
        \   PRINT                           ! Should always be a free
        \   GOTO 1200                       ! record available.
                                            ! Bad error if not.

9070        PRINT
        \   PRINT "Memo "; A.NUM%; " not found."
        \   PRINT
        \   GOTO 1200

11000
        !       H E L P   Function


11010       PRINT
        \   PRINT "     The MEMO program is designed as a computerized 'filing-box'."
        \   PRINT "It s  purpose is to  allow people  who spend a  good deal of time"
        \   PRINT "using the computer  system to use the  computer to  keep track of"
        \   PRINT "ideas or messages."
        \   PRINT
        \   PRINT "     The program creates a file called MEMO.DAT in your"
        \   PRINT "account.  Deleting this file is the quickest way to remove"
        \   PRINT "all your current memos."
        \   PRINT
        \   PRINT "     The TYPE command has several options available. They are:"
        \   PRINT
        \   PRINT "     TYPE                    Type a memo. Program asks which one."
        \   PRINT "     TYPE 10                 Type memo number 10."
        \   PRINT "     TYPE *                  Type all current memos."
        \   PRINT "     TYPE STRING             Type all memos with the"
        \   PRINT "                             specified string in the subject."
        \   PRINT
        \   PRINT "     You may have your memos typed to a file by including"
        \   PRINT "a filename in the TYPE command as follows:"
        \   PRINT
        \   PRINT "     TYPE * > FILENAME       Puts all memos into the"
        \   PRINT "                             file specified by FILENAME."
        \   PRINT
        \   PRINT "Type the RETURN key to continue.";
        \   INPUT LINE #1,A$
        \   PRINT
        \   PRINT
        \   PRINT "     A list of commands follows:"
        \   GOTO 12020

12000
        !       List the legal functions.


12010       PRINT
        \   PRINT "I humbly apologize, but I do not recognize this command."
        \   PRINT "My limited repertoire is as follows:"

12020       PRINT
        \   PRINT "Add          Add a memo to your memo file."
        \   PRINT "Delete       Remove a previously added memo."
        \   PRINT "List         List memos on file."
        \   PRINT "Type         Print the contents of a memo."
        \   PRINT "Search       List memos with a specific subject."
        \   PRINT "More         Append more information to an existing memo."
        \   PRINT "Prompt       Change the current prompt to something else."
        \   PRINT "Help         Gives more details about this program."
        \   PRINT "Exit         Leaves the program."
        \   PRINT
        \   PRINT "All commands may be abbreviated to their first letter. The Type, "
        \   PRINT "Delete and More commands may be followed by a memo number."
        \   GOTO 1200
```

```
13000 !
      !       The magic routine to handle output to a file and
      !       co-ordinate a type by string search function.
      !

13010     REQUEST$ = CVT$$( REQUEST$, 128%+4%)
      \   F% = INSTR( 0%, REQUEST$, ">")        ! Look for file indicator.
      \   IF F% THEN
                PRINT.FILE$ = RIGHT( REQUEST$, F%+1%)
      \         REQUEST$ = LEFT( REQUEST$, F%-1%)
                                                ! Trim file out of command.

13020     REQUEST$ = CVT$$( REQUEST$, 128%)
      \   MEMO.SPEC$ = ""
      \   M% = INSTR( 0%, REQUEST$, " ")        ! Look for memo number/search.
      \   IF M% THEN
                MEMO.SPEC$ = RIGHT( REQUEST$, M%+1%)

13025     P.NUM% = VAL( MEMO.SPEC$)             ! Get memo number (string = err).
      \         GOTO 7000 UNLESS F%             ! If memo number and no file,
                                                ! just print the memo.
      \         SUB.TYPE% = 1%
      \         GOSUB 7012                      ! Print single memo to file.
      \         GOTO 13060

13030     IF MEMO.SPEC$ = "*" THEN 13100        ! Print all the memos.

13040     SUB.SEARCH% = 1%
      \   CALL% = FNFIRST.REC%( USED%)          ! Start at the first record.
      \   CALL% = 1%
      \   FOUND% = 0%
      \   SEARCH$ = MEMO.SPEC$                  ! Get the search string.
      \   IF F% THEN
                OPEN PRINT.FILE$ FOR OUTPUT AS FILE #1%
          ELSE
                OPEN TYPE.FILE$ AS FILE #1%     ! Do file or keyboard.

13050     GOSUB 8015                            ! Look for the record.
      \   IF FOUND% > 0% AND CALL% > 0% THEN
                P.NUM% = CVT$%( MEMO.NUM$)      ! If found, go print it.
      \         SUB.TYPE% = 1%                  ! Set the subroutine flag.
      \         GOSUB 7020
      \         GOTO 13050                      ! And look for another one.

13060     PRINT #1%
      \   PRINT #1%
      \   CLOSE #1%
      \   OPEN TYPE.FILE$ AS FILE #1%
      \   SUB.SEARCH%, SUB.TYPE% = 0%
      \   P.NUM%, FOUND% = 0%
      \   GOTO 1200

13100 !
      !       Print all the memos.
      !

13110     CALL% = FNFIRST.REC%( USED%)          ! Get the first memo.
      \   IF CALL% = 0% THEN                    ! Unless there wasn't one.
                PRINT "No memos to print."
      \         GOTO 13060

13115     IF F% THEN
                OPEN PRINT.FILE$ FOR OUTPUT AS FILE #1%
          ELSE
                OPEN TYPE.FILE$ AS FILE #1%

13120     CUR.MEMO$ = MEMO.NUM$ + ""
      \   P.NUM% = CVT$%( CUR.MEMO$)            ! Get current memo number.
      \   SUB.TYPE% = 1%
      \   GOSUB 7020                            ! Print a memo.
      \   GOTO 13120 IF CALL%                   ! Do another if more.
      \   GOTO 13060

15000
      !       Check out all the disks for the memo file


15010     CHANGE SYS(CHR$(6%) + CHR$(-3%)) TO FQB%
      \   DEVCNT% = FQB%(5%) OR SWAP%(FQB%(6%))

      \   CHANGE SYS(CHR$(6%) + CHR$(-12%)) TO FQB%
      \   DEVNAM% = FQB%(5%) OR SWAP%(FQB%(6%))
      \   DEVOKB% = FQB%(9%) OR SWAP%(FQB%(10%))
                                    ! Get some info from monitor tables.

15020     FOR I% = 0% TO DEVOKB%-2% STEP 2%
      \         DEV$ = CVT$$(SWAP%(PEEK(DEVNAM%+I%)))
      \         FOR J% = 0% TO PEEK(DEVCNT%+I%)
      \               DISKS% = DISKS% + 1%
      \               DISKS$(DISKS%) = "_" + DEV$ + NUM1$(J%) + ":"
      \         NEXT J%
      \   NEXT I%
                                    ! Loop through all disk names
                                    ! Build each disk name in sequence
                                    ! Loop for all units of this disk
                                    ! Increment disk unit counter
                                    ! Build and store complete disk name
                                    ! Until we have all of them

15030     CALL$ = SYS(CHR$(6%) + CHR$(-21))
                                    ! Done with privileges - drop them.

15050     FOR I% = 1% TO DISKS%
      \         MEMO.FILE$ = DISKS$(I%) + "MEMO.DAT"    ! Build a filename.
      \         OPEN MEMO.FILE$ FOR INPUT AS FILE #4%   ! Try to open file.
      \         IF (STATUS AND 1024%) = 0% THEN 1110
                      ELSE
                      PRINT "Don't have write access to ";MEMO.FILE$

15060     NEXT I%
      \   MEMO.FILE$ = SY.MEMO.FILE$   ! Didn't exist - default to system.
      \   GOTO 20050                    ! And go create it.

20000 !
      !       Initialize a new memo file
      !

20010     FIELD #4%, 512% AS INIT$
      \   LSET INIT$ = STRING$(10%,0%)
                     + CVT%$(2%) + CVT%$(1%)
                                    ! Zero all pointers and prompt length.
                                    ! No need to put the prompt
                                    ! string in yet. Program init
                                    ! can handle it. &
      \   PUT #4%, RECORD 1%
      \   FIRST.RUN% = 1%
      \   GOTO 1110

20050     OPEN MEMO.FILE$ FOR OUTPUT AS FILE #4%
      \   GOTO 20010                        ! Re-init a bad memo file.

21000 !
      !       Exit
      !

21010     CLOSE 1,4
      \   GOTO 32767

25000 !
      !       FNUNLINK%
      !
      !       Remove a record from the specified linked list.
      !
      !       TYPE%           Free or Used list indicator.
      !       REC%            Record number to be removed.
      !                               (valid for USED list only)
      !
      !       This routine is the real brains of the linked list
      !       structure of this program. No record should be used
      !       without being UNLINKED from somewhere using this
      !       routine. It performs the only validity checks
      !       you are likely to find in this program.
      !
      !       IF TYPE = FREE
      !            THEN
      !               REMOVE THE FIRST RECORD FROM THE FREE QUEUE
      !               IF THERE WAS A RECORD TO REMOVE
      !                       THEN UPDATE THE FREE QUEUE LINKS
      !               ELSE
      !                       CREATE A NEW RECORD OUT OF THIN AIR
      !
      !       IF TYPE = USED
      !            THEN
      !               REMOVE A RECORD FROM THE USED QUEUE
      !               PLACE IT ON THE FREE QUEUE
      !

25010     DEF* FNUNLINK%( TYPE%, REC% )
      \   GOTO 25050 IF TYPE% = USED%          ! Check for used record.
      \   FR.REC% = FNFIRST.REC%( FREE% )      ! Get the first free record.
      \   GOTO 25030 IF FR.REC% = 0%           ! If none, do something else.
      \   NXT.REC$ = F.LINK$ + ""              ! Save link to next free rec.
      \   GET #4%, RECORD 1%                   ! Go to the main link pointers.
      \   LSET PTR$(FREE%+START%) = NXT.REC$   ! Update the free queue start.
      \   LSET PTR$(FREE%+ND%) = NXT.REC$      ! Update the end pointer
              IF CVT$%(NXT.REC$) = 0%          ! ...if we've emptied the list.
      \   PUT #4%, RECORD 1%                   ! And save it.
      \   IF CVT$%(NXT.REC$) > 0%              ! If there was a next record...
              THEN
                GET #4%, RECORD CVT$%(NXT.REC$) ! ...grab it.
      \         LSET R.LINK$ = NULL.LINK$      ! Null out the reverse pointer
      \         PUT #4%, RECORD CVT$%(NXT.REC$) ! ...and replace the record.

25020   GOTO 25090                              ! And exit gracefully.

25030     GET #4%, RECORD 1%                    ! Go for the big data.
      \   THIN.AIR$ = MAX.MEMO$ + ""            ! Get the next to create.
      \   LSET MAX.MEMO$ =
                CVT%$( CVT$%( MAX.MEMO$) +1%)   ! And move the max ptr. up.
      \   PUT #4%, RECORD 1%                    ! Save the new big data.
      \   LSET F.LINK$ = CVT%$( 0%)             ! Null the forward...
      \   LSET R.LINK$ = CVT%$( 0%)             ! ...and reverse links.
      \   LSET R.TYPE$ = CHR$(FREE%)            ! Initialize as free record.
      \   LSET MEMO.NUM$ = CVT%$( 0%)           ! Might as well do memo num.
      \   LSET MEMO.DATE$ = CVT%$(TODAYS.DATE%)! Put in the date for laughs.
      \   LSET MEMO.DAT$ = STRING$(500%, 0%)    ! And empty out the data fieid.
      \   PUT #4%, RECORD CVT%$( THIN.AIR$)     ! Create record from thin air.
      \   FR.REC% = CVT%$( THIN.AIR$)           ! Keep track of it.
      \   GOTO 25090                            ! Exit...stage left.

25050     FR.REC% = FNFIRST.REC%( USED%)        ! Get the first used record.
      \   FR.REC% = FNNEXT.REC%
                    UNTIL (FR.REC% = REC%)      ! Find the record to remove.
                        OR (FR.REC% = 0%)       ! Unless it's not there.
      \   GOTO 25090 UNLESS FR.REC%             ! Get out if not there.
      \   NXT.PTR$ = F.LINK$ + ""               ! Save forward link.
      \   PRV.PTR$ = R.LINK$ + ""               ! Save reverse link.
      \   IF CVT$%( NXT.PTR$) <> 0%             ! If there was a next record
              THEN
                JUNK% = FNNEXT.REC%             ! Get it
      \         LSET R.LINK$ = PRV.PTR$         ! Update reverse link
      \         PUT #4%, RECORD JUNK%           ! And replace the record.

25060     IF CVT$%( PRV.PTR$) <> 0%             ! If there was a previous record.
              THEN
                JUNK% = FNPREV.REC%             ! Get it
      \         LSET F.LINK$ = NXT.PTR$         ! Update forward link
      \         PUT #4%, RECORD JUNK%           ! And replace the record.

25065     IF CVT$%( NXT.PTR$) = 0%              ! If no next record.
              THEN
                GET #4%, RECORD 1%              ! Get the main pointers.
      \         LSET PTR$( USED%+ ND%) = PRV.PTR$
      \         PUT #4%, RECORD 1%              ! Update used end ptr.

25067     IF CVT$%( PRV.PTR$) = 0%              ! If no previous record.
              THEN
                GET #4%, RECORD 1%              ! Get main pointers.
      \         LSET PTR$( USED%+ START%) = NXT.PTR$
      \         PUT #4%, RECORD 1%              ! Update used start ptr.

25070     IF FR.REC% <> 0% THEN                 ! If a record was removed...
              CALL% = FNLINK%( FREE%, FR.REC%, 0%)! ...place it on the free queue.

25090     FNUNLINK% = FR.REC%                   ! Return the record number.
      \   FNEND
```

```
25100   !
        !       FNLINK%
        !
        !       Add a record to the linked list specified.
        !
        !       TYPE%           Free or Used list indicator
        !       REC%            Record number to add to list
        !       LINKAFTER%      Record number to link record after
        !                       (in USED list only - zero means end)
        !

25110     DEF* FNLINK%(TYPE%, REC%, LINKAFTER%)
        \ GOTO 25150 IF (TYPE% = USED%) AND (LINKAFTER% > 0%)
        \ LAST% = FNLAST.REC%(TYPE%)            ! Move to the last record.
        \ IF LAST% > 0%                         ! If there is a last record
              THEN
               GET #4%, RECORD LAST%            ! Read the last record.
        \      LSET F.LINK$ = CVT%$(REC%)       ! Update the forward link.
        \      PUT #4%, RECORD LAST%            ! And replace the record.
        \      GET #4, RECORD 1%
        \      LSET PTR$(TYPE% + ND%)
                    = CVT%$(REC%)               ! Update last ptr = new record.
        \      PUT #4%, RECORD 1%
        \      GOTO 25130

25120     IF LAST% = 0%                         ! If there was no last record
               THEN
               GET #4%, RECORD 1%               ! Get the main pointers.
        \      LSET PTR$(TYPE%+START%)
                    = CVT%$( REC%)              ! Init the start and
        \      LSET PTR$(TYPE%+ND%)
                    = CVT%$( REC%)              ! end pointers.
        \      PUT #4%, RECORD 1%

25130     GET #4%, RECORD REC%                  ! Get the record to link.
        \ LSET F.LINK$ = NULL.LINK$             ! Null the forward link.
        \ LSET R.LINK$ = CVT%$(LAST%)           ! Old LAST is reverse link.
        \ LSET R.TYPE$ = CHR$(TYPE%)            ! Add record type.
        \ PUT #4%, RECORD REC%                  ! Put the record in the file.

25140     FNLINK% = REC%                        ! Return installed record #.
        \ FNEND

25150     CALL% = FNFIRST.REC%( USED%)          ! Get the first record in list.
        \ CALL% = FNNEXT.REC%
                    UNTIL CALL% = LINKAFTER%
                       OR CALL% = 0%            ! Find the record to add to.
        \ GOTO 25190 IF CALL% = 0%              ! Exit if not there.
        \ NXT.PTR$ = F.LINK$ + ""               ! Save pointer to next record.
        \ PRV.PTR$ = R.LINK$ + ""
        \ LSET F.LINK$ = CVT%$( REC%)           ! Update forward pointer.
        \ PUT #4%, RECORD CALL%                 ! Restore the record.
        \ GOTO 25160 IF NXT.PTR$ = NULL.LINK$
        \ GET #4%, RECORD CVT%$( NXT.PTR$)      ! Get the next record.
        \ PRV.PTR$ = R.LINK$ + ""
        \ LSET R.LINK$ = CVT%$( REC%)           ! Install new pointer.
        \ PUT #4%, RECORD CVT%$( NXT.PTR$)      ! Restore the record.

25160     GET #4%, RECORD REC%                  ! Get the record to link in.
        \ LSET F.LINK$ = NXT.PTR$               ! Set up the links.
        \ LSET R.LINK$ = PRV.PTR$
        \ LSET R.TYPE$ = CHR$( TYPE%)
        \ PUT #4%, RECORD REC%
        \ GOTO 25140

25190     REC% = 0%
        \ GOTO 25140


25200   !
        !       FNNEXT.REC%
        !
        !       Return the next record number in a linked list.
        !       Works with the free or used list, depending on
        !       which list was used last.
        !
        !       F.LINK$         Must remain unaltered after last
        !                               record read. This contains
        !                               the forward link used here.
        !       FNNEXT.REC%     Returns next record number (or zero
        !                               if no next record). If there
        !                               is a next record, it is read.
        !

25210     DEF* FNNEXT.REC%
        \ NXT.REC% = CVT%$(F.LINK$)             ! Read forward link for next.
        \ IF NXT.REC%                           ! If there is a next rec.
              THEN GET #4%, RECORD NXT.REC%     ! ...then read it.

25220     FNNEXT.REC% = NXT.REC%                ! Return the record # (or zero).
        \ FNEND

25300   !
        !       FNPREV.REC%
        !
        !       Return the previous record number in a linked list.
        !       Works with the free or used list, depending on
        !       which list was used last.
        !
        !       F.LINK$         Must remain unaltered after last
        !                               record read. This contains
        !                               the backward link used here.
        !       FNPREV.REC%     Returns previous record number (or zero
        !                               if none). If there is a
        !                               previous record, it is read.
        !

25310     DEF* FNPREV.REC%
        \ PREV.REC% = CVT%$(R.LINK$)            ! Get the backward link.
        \ IF PREV.REC%                          ! If there is one
              THEN GET #4%, RECORD PREV.REC%    ! ..read the record.

25320     FNPREV.REC% = PREV.REC%               ! Return record # (or zero).
        \ FNEND


25400   !
        !       FNLAST.REC%
        !
        !       Return the last record number in a linked list.
        !
        !       TYPE%           List type free or used.
        !       F.LINK$         Must remain unaltered after last
        !                               record read. This contains
        !                               the backward link used here.
        !       FNLAST.REC%     Returns last record number (or zero
        !                               if none). If there is a
        !                               last record, it is read.
        !

25410     DEF* FNLAST.REC%( TYPE% )
        \ GET #4%, RECORD 1%                    ! Get the main links.
        \ LAST% = CVT%$(PTR$(TYPE%+ND%))        ! Get END pointer for TYPE.
        \ IF LAST%                              ! If there is a last record
              THEN GET #4%, RECORD LAST%        ! ..then read it.

25420     FNLAST.REC% = LAST%                   ! Return record number.
        \ FNEND

25500   !
        !       FNFIRST.REC%
        !
        !       Return the first record number in a linked list.
        !
        !       TYPE%           List type free or used.
        !       FNFIRST.REC%    Returns first record number (or zero
        !                               if none). If there is a
        !                               first record, it is read.
        !

25510     DEF* FNFIRST.REC%( TYPE% )
        \ GET #4%, RECORD 1%                    ! Get the main links.
        \ FIRST% = CVT%$(PTR$(TYPE%+START%))    ! Get START pointer for TYPE.
        \ IF FIRST%                             ! If there is a last record
              THEN GET #4%, RECORD FIRST%       ! ..then read it.

25520     FNFIRST.REC% = FIRST%                 ! Return record number.
        \ FNEND

32000   !
        !       Handle those errors
        !

32010   IF ERL = 15050%
               THEN RESUME 15060
                    ! Keep checking disks to find the memo file.

32030   IF ERR = 11% AND ERL = 6010% THEN
               FIELD #1%, RECOUNT -1% AS PART.OF.THE.NEW.PROMPT$
        \      NEW.PROMPT$ = NEW.PROMPT$
                           + PART.OF.THE.NEW.PROMPT$
        \      RESUME 6020
                    ! Handle the control-Z at the end of the prompt.

32040   IF ERR = 11% AND ERL = 2035% THEN
               FIELD #1%, RECOUNT -1% AS MEMO.LINE$
        \      MEMO$ = MEMO$ + MEMO.LINE$
        \      RESUME 2040
                    ! Handle the control-Z at the end of the memo.

32050   IF ERR = 11% AND ERL = 2020% THEN
               RESUME 1200
                    ! Control-Z no good to end memo subject.

32070     IF ERR = 28%
               OR (ERR =11% AND ERL = 1200%) THEN
                    CLOSE 1%,4%
        \           RESUME 32767
                    ! Exit on control-C
                    ! or control-Z at prompt.

32080   IF (ERL = 3010%)
               OR (ERL = 7010%)
               OR (ERL = 8010%) THEN RESUME 1200
                    ! Head back to the prompt if any problems.

32090   IF ERL = 1210% THEN
               RESUME 1220
                    ! Bad number given on input (Type, Delete)

32100   IF ERL = 13025% THEN
               RESUME 13030
                    ! If memo number was a search string

32110   IF (ERL = 13040%)
               OR (ERL = 13115%)
               OR (ERL = 13210%) THEN
                    PRINT
        \           PRINT PRINT.FILE$; "?  Looks like GARBAGE to me!"
        \           PRINT
        \           RESUME 1200
                    ! Complain about filename.

32120   IF ERL = 9010% THEN
               RESUME 1200
                    ! Bail out if bad memo number in MORE.

32600   IF ERL > 25000%
               AND ERL < 32000% THEN
                    PRINT
        \           PRINT "MEMO file format error"
        \           RESUME 32710
                    ! Fix the error in the file.

32610   IF ERL = 32710%
               THEN RESUME 21000
                    ! Quit if error in init file answer.

32620   IF ERL = 11010% THEN
               RESUME 1200
                    ! They got tired of the instructions.

32700     PRINT
        \ PRINT "? Unexpected Error"
        \ PRINT RIGHT(SYS(CHR$(6)+CHR$(9)+CHR$(ERR)),3%);
        \ PRINT " in line "; ERL
        \ RESUME 21000
                    ! Can't expect everything...

32710   PRINT
        \ PRINT "Re-initialize the file? ";
        \ INPUT LINE #1, A$
        \ A$ = LEFT(CVT$$(A$,-1%),1%)
        \ IF A$ = "Y" THEN GOTO 20050
              ELSE PRINT "? Bad MEMO file - can't continue"
        \     CLOSE 1,4

32767   END
```

Meet Eddie. He's learning everything about the business.



Everything.

LOCK-11 is a system security and management package for RSTS.

LOCK-11 gives you absolute control of access by keyboard or user-I.D.

LOCK-11 provides an optional MENU environment that keeps non-privileged users where they belong.

LOCK-11 offers the system manager powerful surveillance utilities that actually improve thruput.

LOCK-11 is very well documented, supported and enhanced regularly.

LOCK-11 is available right now. Circle the response number below for a full set of documentation, or call 215-364-2800.

LOCK-11 ©NDD

CIRCLE 12 ON READER CARD

HOW TO USE SHAREABLE DATA

By Ken Isaacs, Digital Management Group, Ltd., Willowdale, Ontario, Canada

OVERVIEW

With the advent of RSTS/E V7.0, there was considerable support given to resident libraries. They are usable for code or data sharing between many concurrently active jobs. We also have been provided with all the tools and raw data from Digital with which to manipulate these new features. What has not been clear, however, is "HOW DOES ONE SHARE DATA EASILY BETWEEN PROGRAMS WRITTEN IN A HIGH LEVEL LANGUAGE?" Relax, sit back and read. I will present the simple steps you too can take. I will briefly discuss:

- why use resident common areas,
- cookbook instructions on how to make one,
- elephant traps to avoid.

INTENDED AUDIENCE

This document is intended for those of you who know and love RSTS/E and plan to do great things with your system. The following discussion does get rather detailed. I have tried to make it as clear as possible so that it is not necessary to know lots about machine language, task builders and the like. My objective is that you will learn how to share data between concurrently active programs using resident common areas.

PREREQUISITE READING

Although there is no mandatory prerequisite reading, I suggest that you become familiar with:

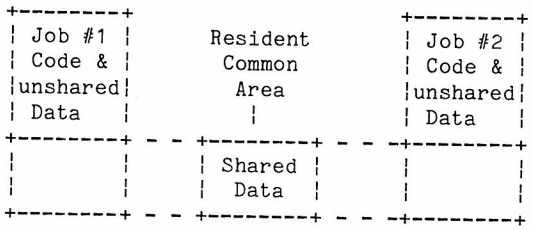
- one high-level language (BASIC+2 or DIBOL) and what it generates in terms of object code (PSECTs, Global symbols, etc.),
- a little (just a little, mind you) MACRO,
- the Task Builder utility.

The following reference manuals will be handy:

- Task Builder Reference Manual,
- MACRO-11 Language Manual,
- RSTS/E System Manager's Guide — for UTILTY,
- RSTS/E Programmer's Utility Manual — for MAKSIL.

DEFINITION OF COMMON AREAS

A common area is a portion of a program's address space that is used to store/retrieve information that is used between modules. Many modules in one program can access this area of common information. The term, resident common area applies to a special case of common area. A resident common area is one that is separately loaded by the system, and is connected to by one or more programs. Thus it is possible for several jobs to easily share their data.



Physically
Separate
and Shared

WHY USE RESIDENT COMMON AREAS?

The addressing limitations of PDP-11s in commercial-applications are well known by this time. Many applications use:

- complex program overlay structures,
- many detached/chaining jobs,
- virtual arrays,
- scratch pad files,
- large amounts of interjob message sending,
- frequently used smallish size (1-40 blocks) tables that are expensive to put in files,
- any of the above,
- all of the above.

I cannot realistically do away with the requirement for a large amount of program segmentation. What I can inform you of is an easy to use mechanism for sharing data between many small jobs. Your application design will have to ensure that all individual programs do not become too large and unwieldy.

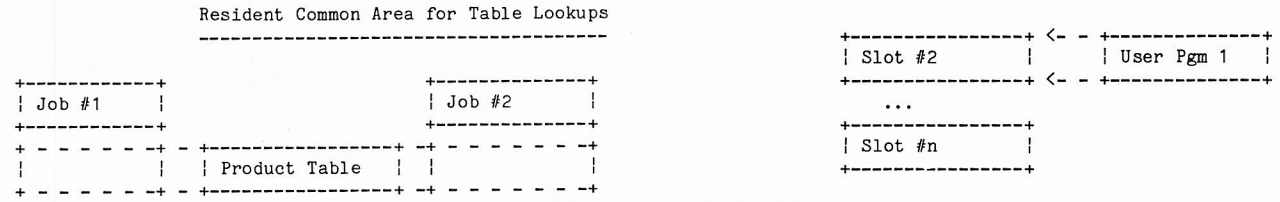
SAMPLE USES OF RESIDENT COMMON AREAS

It is hard to visualize the use of a resident common area unless it is put into an "application" example. The following are two examples of existing applications that could use resident common areas:

- high speed table lookup,
- processing of the same data by several jobs.

HIGH SPEED TABLE LOOKUP

One application involves high volumes of data entry with minimal verification. The longest time spent in verification is looking up the product code to see if it exists. Additionally, the verification table is updated periodically and the changes must be in effect at once. As a result, this table is stored in a file. There are only a couple of hundred product codes. This would fit neatly into an array in a resident common area and allow any product additions or deletions to take effect immediately.



Product Table is part of each program's virtual address space

PROCESSING BY SEVERAL JOBS

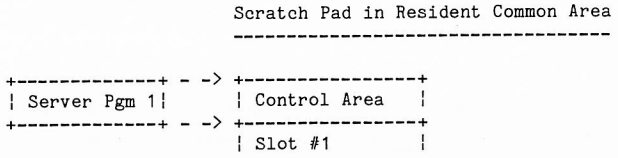
The other application involves several on-line programs and several detached programs processing the same area of information. The detached programs are responsible for handling one function common to accessing the data. For example, the first detached program copies from a master file to a common scratch pad where the on-line programs manipulate the data. The second detached program copies the data back to the master file. The third and fourth programs provide some application dependent functions common to each set of records. A large resident common area could be used to store the scratch pad and the "overlay" feature that I discuss later could be used to address the pertinent portion of the scratch pad.

Each program only maps the part of resident common area that it is currently using.

OTHER POSSIBLE USES

Another use of a large resident common area is that of a "file handler." There are several sites that use file handlers so that it is not necessary to build the file access code into each program (i.e., RMS). As a result, they use a file handler and inter-job messages to access the data. A file handler could use a resident common area in two ways:

- store the file (or portions of it) in its own private resident common area. This amounts to a more sophisticated data caching technique. It also points out another interesting characteristic of resident common areas and that is they do not have to be shared; they can be used by one program exclusively.
- passing the records back and forth between the file handler and the rest of the application.



COMMON AREA COOKBOOK

I will now discuss the steps necessary to build a resident common area suited to your requirements.

- The first thing I must do is address some language dependencies.

RPM CAN DRAMATICALLY INCREASE THE PERFORMANCE OF YOUR SYSTEM

You can double or triple your system's performance without adding costly hardware. RPM provides everything you need to optimize your system and keep it running at peak performance.

RPM analyzes your system performance automatically, identifying problem areas. But, RPM doesn't stop there! It identifies the cause of the problems and makes suggestions for correction in plain English.

AUTOMATIC PERFORMANCE ANALYSIS

Instead of dumping columns of cryptic numbers, RPM gives you a plain English report that describes how your system is performing. It tells you where you have problems, what caused the problems, and how to fix them.

This report analyzes each of the resources that are common problem areas. Any resources that are not being used optimally are identified. The program or files that caused the problem are then identified and suggestions are made for correcting the problem.

```
---- File Processor (FIP) Usage ----

** The file processor (FIP) is being excessively used.  It is in use by at
   least one job 67% of the time.  In addition, an average of  2.7 other users
   are waiting to use FIP.  Although the file processor is in use 67% of the
   time, it is waiting for information from the disk 72% of the time it is in
   use.  The disk information that FIP is waiting for breaks down as follows:
     91% Directory.
         Reading or writing to the directory structure.
      7% Disk Allocation Table (SAT)
         Reading or writing information about free blocks on the disk.
      2% Monitor Overlays.
         Loading monitor overlays into memory.
      0% Miscellaneous.
         Loading disk cache information and other miscellaneous data.
   The five programs that use the most FIP resources are:
      OE047A      275.2
      PAYREC      213.1
      ...TKB      158.0
      BP2COM      110.3
      LOGIN        78.7
   FIP usage can be decreased by optimizing the clustersize of frequently used
   files (see section 4.4.1 of the RPM User's Guide), using contiguous files
   where possible, reordering the directories often and minimizing the opening
```

DETAILED PROGRAM ANALYSIS

In addition to identifying problem programs, RPM can analyze the operation of individual programs, identifying problem areas.

RPM's detailed program analysis breaks the operation of the program down into CPU usage, input/output and system calls. Usage and directory overhead counts are displayed by channel and by system call.

```
RPM> EXAMINE JOB 5 EVERY 5 MINUTES

Job:  5    Program: *ALL*     CPU: 54%    Sample Time:  297 Seconds

Chan          File             Count Ovrhd Chan          File             Count Ovrhd
   0 KB2:                          3          1 * Closed *                    4
   3 DM1: [1,3]CSPCOM.OLB        835     2    4 DM1: [5,1]TEMP05.TMP         39    42
   6 DM1: [5,1]EXAMPL.STB          9    91    7 DM1: [5,1]EXAMPL.TSK        733
  15 DM1: [1,3]TKB    .TSK       278

 EMT   Count Ovrhd  EMT   Count Ovrhd  EMT   Count Ovrhd  EMT   Count Ovrhd
CALFIP   149        .READ  1441     2 .WRITE   460    133 .CORE      7
.TTECH     1        .TTRST    1       .DATE      1        .NAME      2
.RTS       1        .LOGS    12       .CLEAR     2        .CCL       1     5
.FSS      42        .UUO     71       .RSX       1        CLSFQ     62    28
OPNFQ     58    535 CREFQ     2   226 DLNFQ      1     23 RSTFQ      2
LOKFQ     22    259 CRTFQ     1    64 CRBFQ      1    175 UU.ATR    70   250
UU.NAM     1      3
```

EXTENDED PERFORMANCE STATISTICS

RPM adds extended performance statistics to your RSTS/E monitor. This monitor extension captures information about overall system performance plus information on individual programs and files. Using this information, you can improve system performance by minimizing disk head movement, reducing file processor (FIP) waits, reducing swapping, and optimizing disk cache operation.

```
RPM> HISTOGRAM DL0_SEEK_DIST

                  0....1....2....3....4....5....6....7....8....9....0
                  ----------------------------------------------------
DL0:SEEK_DIST_1   ************************                           |   47.1
DL0:SEEK_DIST_4   ************                                       |   22.9
DL0:SEEK_DIST_16  *********                                        |   16.5
DL0:SEEK_DIST_64  ******                                           |   10.8
DL0:SEEK_DIST_100 **                                               |    2.7
                   ---------------------------------------------


RPM> HISTOGRAM DL0_DISK_USAGE

                  0....1....2....3....4....5....6....7....8....9....0
                  ----------------------------------------------------
DL0:DISK_USAGE_0  **                                               |    2.1
DL0:DISK_USAGE_10 **                                               |    2.0
DL0:DISK_USAGE_20 **                                               |    3.1
DL0:DISK_USAGE_30 **                                               |    3.6
DL0:DISK_USAGE_40 *********                                        |   17.6
DL0:DISK_USAGE_50 ***********************                          |   42.2
DL0:DISK_USAGE_60 **********                                       |   19.8
DL0:DISK_USAGE_70 ***                                              |    4.2
DL0:DISK_USAGE_80 **                                               |    3.9
DL0:DISK_USAGE_90 *                                                |    1.1
                   ---------------------------------------------
```

DYNAMIC PLOTTING

In addition to comprehensive reports, RPM can generate a wide variety of graphs. It can plot curves, draw bar charts, and plot histograms using any combination of system information. By plotting one variable against another, you can immediately see correlations between variables. This is especially useful for determining critical resource usage points.

```
RPM> PLOT USER_CPU, MONITOR_CPU BY HOUR

          0....1....2....3....4....5....6....7....8....9....0     *      #
          ----------------------------------------------------
     8.0 |  #                  *                                |  41.2   7.7
     9.0 |   #                  *                               |  44.0   8.1
    10.0 |   #                     *                            |  54.5   8.9
    11.0 |    #                                *                |  79.1  12.6
    12.0 #*                                                      |   0.0   2.0
    13.0 |     #                               *                |  74.3  12.3
    14.0 |      #                                  *            |  82.1  14.2
    15.0 |      #                              *                |  79.3  13.1
    16.0 |     #                               *                |  70.7  10.3
          ----------------------------------------------------

       Variable                  Description                Min     Max     Avg
X SAMPLE_HOUR          Hour of Day                          8.0    16.0    12.0
* USER_CPU             User CPU Time                        0.0    82.1    58.4
# MONITOR_CPU          Monitor CPU time                     2.0    14.2     9.9
```

IDENTIFY PROGRAMS BY RESOURCE USAGE

With RPM, you can identify the programs that will provide the most benefit from optimization. You can determine which programs are used most often and which programs use the most of critical resources.

Once identified, these programs can be optimized. Overall system performance can be increased by making changes only where they will do the most good.

```
RPM> LIST TOP 5 PROGRAMS BY PRG_CPU_TICKS
      PARREC      542
      DAYEND      168
      PAYROL      142
      PAYANA      103
      CLENUP       73
RPM> LIST TOP 5 FILES BY FILE_DIR_OVRHD
      [3,10]PROFIL.JOU     33.1
      [3,10]PROFIL.B2S     27.6
      [2,0]MTHEND.TSK      19.6
      [10,1]EOMREC.JOU     11.3
      [2,0]PAYAUD.DAT       7.8
RPM>
```

NO-RISK GUARANTEE

If all this sounds too good to be true, here's your chance to find out for yourself. Try RPM for 30 days. Use it to tune your system. If RPM isn't everything we say it is and more, return it for a full refund. We're not worried. We know that once you see the difference RPM can make, you won't want to be without it.

RPM -- The Performance Revolution

NORTHWEST DIGITAL SOFTWARE

NORTHWEST DIGITAL SOFTWARE, INC. BOX 2-743 NEWPORT, WA 99156 (509) 447-2620

• Secondly, I'll talk about simple resident common areas.

• Lastly, I'll describe overlaid resident common areas.

LANGUAGE DEPENDENT BACKGROUND

Your choice of high-level language will affect the remainder of the discussion. This is a result of the differing ways that the languages generate common data references. The remainder of this discussion centers on the use of BASIC+2, although it may also be applied to F4P and DIBOL.

BASIC+2 shares data through MAPs or COMmons. The output from the compiler contains program sections (PSECTs). Each MAP/COMmon has one PSECT that has the same name as the MAP/COMmon name. For example:

```
100     Map     (ADATA)
                First$      = 10%
        ,       Second$     = 10%
        ,       ...
```

produces a PSECT named ADATA. The task builder (TKB) builds tasks by combining all references to a PSECT within a segment. This characteristic is important when dealing with overlaid resident common areas and their definitions.

DIBOL, on the other hand, produces global symbols for each of the COMMON variables. This makes using the overlaid resident common areas extremely easy to use in DIBOL, in that the task builder resolves all offsets. However, I haven't run into very many DIBOL shops. If there are those of you who are interested, feel free to give me a call.

FORMS OF SHARING

I have experimented with two different forms of resident common areas:

• non-overlaid resident common area (mapped using a single static window into the resident region),

• overlaid resident common area (mapped using a dynamic window(s) into a LARGE resident region),

and found several differing characteristics between the two forms. The appropriate use of either must be determined at design time.

NON-OVERLAID RESIDENT COMMON AREA

The non-overlaid resident common area is the easiest form to use. The fundamental sequence to generate one is:

1. Define it — determine which PSECTs and/or global symbols are the ones to be shared and design the resident common area.

2. Create allocation for it — using a quicky MACRO program (3-4 lines usually, examples to follow).

3. Build it — using the task builder with resident library switches. (See the examples and Chapter 7 of the RSTS/E Task Builder Manual.)

4. Prepare for installation — run the output task through MAKSIL.

5. Install the resident common area mapped Read/Write.

To use a non-overlaid resident common area, you take the following steps:

1. Create your high-level language program(s).

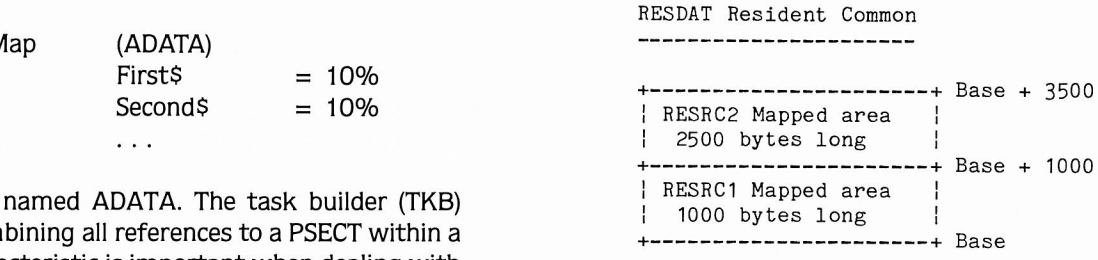
2. Task build using the new resident common area (see Options in TKB manual, Chapter 3.2).

3. Run your programs.

The first set of steps is done only once. After that, using it is as easy as falling out of bed.

DEFINE A COMMON AREA

Let's assume that we have decided to share two record/common areas. We will call them RESRC1 and RESRC2. Their lengths are 1000 bytes and 2500 bytes respectively. We will call the common area RESDAT and use account [1,53] for the development. These names and sizes are examples only and can be anything that you want them to be.



where: BASE is on an even 8k byte boundary.

CREATE IT

The first step is to write the MACRO program, thusly:

```
.TITLE RESDAT       ; Looks pretty on listings.
.IDENT /01KI/       ; For those of you who keep track of
                    ; versions.
.RADIX 10           ; So that you do not have to count in
                    ; octal.
.PSECT  RESRC1,RW,D,GBL,REL,OVR
                    ; The attributes match those
                    ; generated by B+2 and are
                    ; described in TKB manual (section
                    ; 4.2.1). The name must match the
                    ; COMmon/MAP name that you plan
                    ; to use in the program.
.BLKB   1000        ; Reserve 1000 bytes
.PSECT  RESRC2,RW,D,GBL,REL,OVR
.BLKB   2500        ; Reserve 2500 bytes
.END                ; Required by MACRO
```

BUILD IT

Next, you assemble this module (MAC RESDAT=RESDAT) and task build it as follows:

```
TKB
TKB>RESDAT/-HD/PI,,RESDAT=RESDAT
TKB>/
ENTER OPTIONS:
TKB>PAR=RESDAT:0:0
TKB>STACK=0
TKB>//
```

Notice the following points about the task build.

• The task image has NO header and IS position independent.

- NO header is mandatory for resident libraries.
- Position independence is recommended.

• No map was called for. You may if you want, but there is nothing much there.

• A symbol file WAS called for. This is required for MAKSIL.

• The PAR option specified no fixed address range (0:0). This is the preferred way, because the task builder can then fit it in more easily to your program later.

• The stack was set to zero. Remember that this is a shareable thing. Stacks exist in your own job area.

PREPARE FOR INSTALLATION

The next step is to run MAKSIL and create the resident library, sort of like this:

```
RUN $MAKSIL
AKSIL V07 . . .
Resident Library Name ? RESDAT
Task-Built Resident Library Input File <RESDAT. TSK> ?
Include symbol table (Yes/No) <Yes> ?
Symbol Table Input File <RESDAT.STB> ?
Resident Library Output File <RESDAT.LIB> ?
RESDAT built in 2 K-words, 0 symbols in the directory
RESDAT.TSK renamed to RESDAT.TSK<40>
```

INSTALL IT

Finally, you add the resident common area, using UTILTY. Up to this point, all functions may be performed under a non-privileged account. This is the ONLY step that must be done using a privileged account.

```
RUN $UTILTY
UTILTY V07 . . .
# ADD LIBRARY [1,53]RESDAT<0>/ADDR:???/RW
# EXIT
```

Two points to notice:

1. The use of an explicit protection code.

The default protection code is <42> when you add a library. That implies Read access for everyone, but no Writing. You must supply a valid protection code in the ADD statement despite the protection code that is on the resident library disk file. Now, you set any protection code that you want to protect it at the user, group, or system level. Even privileged programs will get "?Protection violation" if they try to access a library that does not allow them write access. I suggest that you use a protection code of <0>.

2. The use of the /RW switch.

The default mapping of the resident library is Read-only. You must specify that the resident common area is mapped Read/Write or the memory management hardware might take it to mind to zap you, if RSTS/E doesn't get you first.

FIRST STEP FOR USAGE

You now have a resident common area ready to use. The following BASIC+2 example will show just how easy it is to really use it.

```
100     Map     (RESRC1)! Use the same PSECT or
                        ! MAP name, please
                Cust.ID$(199)=5%! 200 customers

\       Map     (RESRC2)! Second record
                Prod.Code$(99) = 5%
        ,       Prod.Desc$(99) = 45%

1000

!               MAIN — LINE STARTS HERE
```

The program itself is standard BASIC+2 code. The two stipulations you have are that:

• The MAP names must match the PSECT names in the MACRO module.

• The length of your MAPs CANNOT exceed the allocated space without incurring some problems. I'll leave it up to you to discover them. They quickly become self evident.

SECOND STEP FOR USAGE

In the command file, you add ONE line to use the resident common area.

```
                    Sample Command File
                    -------------------

                    SY:PROGA,SY:PROGA=SY:PROGA/MP
                    UNITS=12
                    ASG=SY:5:6:7:8:9:10:11:12
                    RESLIB=LB:BASICS/RO
Additional line ->  RESCOM=[1,53]RESDAT/RW
                    EXTTSK=1024
                    //
```

Use TKB to build your program. There should be no error messages issued by TKB.

If there are, you've messed up somewhere. You should check the map file and make sure that the starting address of the first PSECT (RESRC1 in this example) is starting on a high-up 8kb boundary. Since the starting addresses are in octal, the thing to remember is that 8kb is 20000 (octal) bytes. So you should see a line in the memory allocation synopsis that looks like this:

```
RESRC1(RW,D,GBL,REL,OVR)  120000 001750 001000.
                                  ^
                                  |
This is the starting address--+
```

Now, run your program. This is the end of the simpler version of resident common.

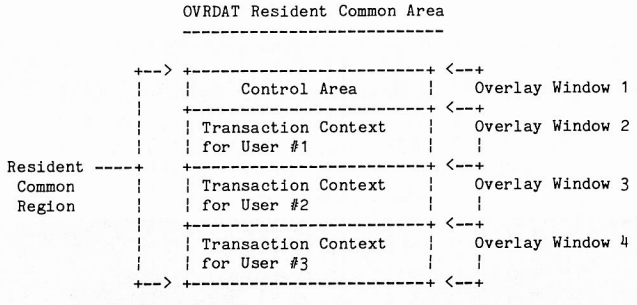
OVERLAID RESIDENT COMMON AREA

The term "over-laid resident common area" is loosely used. The data is not physically overlaid, only the Active Page Registers are modified to map the correct section of data into your virtual address space. All the data resides physically in memory. I use the term overlaid common because it is mapped using the "auto-load overlay mechanism." The concept of building overlayable resident common areas came to me from the combination of two references in the task builder manual. Knowing how to use them has come from a lot of hacking. You can check the TKB references yourself:

- using the .NAME directive to create loadable segments that contain no executable code (Chapter 5, pp. 5-11 to 5-13),
- using the auto-load/overlay mechanism to dynamically remap your window in a resident region (Chapter 7).

SAMPLE USE OF OVERLAID COMMON

Let's assume that we have an application that is a transaction processor. We want to have several common modules that perform a given function for the whole application. For example, a screen manager to handle the screen traffic, several file handlers to handle the RMS traffic and a transaction processing manager to schedule all and sundry activities. As a result, we have many common areas of the same layout, one for each of the currently active users. It is possible to use a scratch pad file or send around a lot of large messages. Both of these approaches involve significant overhead (one on disk accesses and the other on small buffers). Instead let's design a resident common area that looks like this:



The usage of the different areas and sizes of each may be different. This example uses two types of areas and two different sizes. The control region would contain the information used in controlling the remainder of the areas (e.g., areas in use, areas free) and perhaps some application data (e.g., current business date, application status). Each of the context areas would contain the same format for the data area but each would contain data unique to a given user. Then, when a program was servicing user #1, it would "overlay" into the transaction context area for that user, and so on for each user being serviced. In this fashion, a program could easily service many users.

Now, I realize that the example of the application is rather complex, but I have made it so to deliberately show possible uses. There are many more complex and many less complex. Actual uses I will leave to the reader (that's you).

MAKING IT

The fundamental sequence to generate an overlaid resident common area is similar to generating a non-overlaid resident common area:

1. Define it — determine what data is to be shared and design the overlay structure.
   - for BASIC+2 use MAPs/PSECTS
   - for DIBOL use variables/Global symbols
2. Create it — using several MACRO programs.
3. Build it — using an overlay descriptor file.
4. Prepare for installation — by running the output task through MAKSIL.
5. Install the resident common area mapped Read/Write.

To use an overlaid resident common area, you take the following steps:

1. Create your high-level language program(s) with the following characteristics:
   - specifically defined MAPs (more on that later),
   - calls to NAMEd segments to remap the areas as required.
2. Build your program using the new resident common area.
3. Run your program.

As before, the first set of steps is done only once.

The three points that are CRITICAL to the successful completion of this portion are:

1. setting up the MACRO program correctly and sneakily to overcome TKB "features,"
2. specifying the correct overlay description so that your data does not get muxed ip, and
3. coding your program so that you do not tamper with the wrong things.

DEFINING A RESIDENT COMMON AREA

Using the previously described example, we will design an overlaid resident common area with a zero length root and two types of segments.

1. The first segment type is the control area and it exists once. The size of the segment is 1000 bytes and it contains data about the availability of the other three slots and who owns them, etc.
2. The second segment type is the transaction context area and it is duplicated three times, once for each active user. The size of this area is 8192 bytes. It contains three major sections:
   - a screen buffer containing the current data from the entire screen,
   - a workspace that contains the current records and other essential variables for this transaction,
   - and a journaling area for record updates.

Voyager™ I from Tymshare
Your FUTURE Is NOW !

The future has a way of becoming the present. Tymshare has made the future NOW, in announcing the first in a family of DEC* compatible disk subsystems.

Tymshare Voyager Series disk subsystems offer the DEC computer user increased disk storage capacity, reduced space and power requirements and attractive pricing alternatives.

Have you evaluated your present disk subsystem costs?

When it comes to experience and reputation Tymshare is second to none. When it comes to Price/Performance and Mixed Vendor Support, Tymshare is one step beyond . . .

TYMSHARE®

39100 Liberty St. / Fremont, Ca. 94538
(415) 794-2538

*DEC, DIGITAL and VAX are registered trademarks of Digital Equipment Corporation

CIRCLE 36 ON READER CARD

CREATE IT

First we write a MACRO program to define and allocate the space for the control area. It looks something like this (explanations follow details, like in a mystery):

```
.TITLE OVRCTL      ; Keeps it separate from the
                   ; other example.
.IDENT /01KI/
.RADIX 10          ; for decimal phreaks
.PSECT DUMMY,RW,D,GBL,REL,CON
.BLKB  448
.PSECT DUMMY,RW,D,GBL,REL,CON
.BLKB  448
.PSECT DUMMY,RW,D,GBL,REL,CON
.BLKB  104
.END
```

Additionally, we describe a MACRO program to allocate the space for the transaction context area, thusly:

```
.TITLE OVRTXN      ; Pretty, eh?
.IDENT /01KI/
.RADIX 10
.PSECT DUMMY,RW,D,GBL,REL,CON
.BLKB  448
.PSECT DUMMY,RW,D,GBL,REL,CON
.BLKB  448
   .       repeat for a total of 18 times to
   .       get 18*448 bytes (or 8054)
   .
.PSECT DUMMY,RW,D,GBL,REL,CON
.BLKB  78
.END
```

The last MACRO program is required to give a you a zero length PSECT to use in BASIC+2. It's the simplest.

```
.TITLE NULROT
.IDENT /01KI/
.PSECT OVRDAT,RW,D,GBL,REL,OVR
.END
```

Looking at these examples, the first question that comes to mind is "What the @#%↑@# ?". More lucid questions are:

- Why repeat the PSECT names with all those unusual numbers?
- Why have a zero length PSECT?

CREATION EXPLAINED

Answers: First of all, I don't know "What the @#%↑# ?"

- TKB gets very upset about allocating empty space in overlaid segments that have the GBL and NODSK attributes. What this boils down to is that an overlay segment in an overlaid resident common has:
  - the GBL and NODSK attributes set automatically for each overlaid segment. NODSK states that there is no loading of data/code that occurs when the segment is invoked. So in practice, the autoload mechanism will map the window but will not load from disk. This is exactly what we want.
  - to have space allocated so that we can access the required addresses when we reference the data. Combine this with the fact that TKB does not allow pre-allocation of empty space and we have . . .
  - an interesting opportunity, AND
  - a tacky solution. TKB will not complain IF the PSECT has the CONcatenate attributes AND each of references does not exceed 448 bytes (340 octal words). Therefore, you will notice that the attributes for DUMMY are not OVeRlaid.
- A zero length PSECT is only required for BASIC+2. BASIC+2 generates a PSECT with the same name as your MAP. The PSECT is then the common point of reference. TKB will not store PSECT names from overlay segments in the symbol table. Therefore, when you build a resident common you get all PSECTs from the root, but none from the overlay segments and as a result it must be faked. The solution is to define a zero length root with one PSECT name and use TKB's inbred intelligence to overcome the small discrepancies in length (amidst many diagnostic messages).

An aside for DIBOL users, you do not have to go through the dummy PSECT route. You may define your common variables as global symbols at the correct offset. The naming convention of the global symbols is such that the five character variables have a dollar sign appended or preceding it (I am not sure which). Experiment.

BUILD IT WITH AN ODL FILE

If I still have your attention at this point . . . let's get onto the ODL file. Remember this is the second critical point (but not as messy as the previous one):

```
        .ROOT NULROT-*!(AREA0,AREA1,AREA2,AREA3)
        .NAME CTRL00,GBL
        .NAME SLOT01,GBL
        .NAME SLOT02,GBL
        .NAME SLOT03,GBL
AREA0:  .FCTR CTRL00-OVRCTL
AREA1:  .FCTR SLOT01-OVRTXN
AREA2:  .FCTR SLOT02-OVRTXN
AREA3:  .FCTR SLOT03-OVRTXN
        .END
```

and we need the accompanying command file:

```
SY:OVRDAT/-HD/PI,OVRDAT,OVRDAT=SY:OVRDAT/MP
PAR=OVRDAT:0:0
STACK=0
GBLREF=CTRL00,SLOT01,SLOT02,SLOT03
//
```

Task build this arrangement. There should be no errors. Errors that I have encountered are:

- TKB — FATAL — I/O ERROR ON OUTPUT FILE — this very informative message says clearly that the preallocated space in one of your PSECTs in an overlay segment is too big. Reduce the size of the .BLKB directive.

STEPS 3 & 4

You then run it through MAKSIL as in the previous case (just substitute OVRDAT for every occurence of RESDAT).

You must now add the library using UTILTY. The notes mentioned in the non-overlaid common also apply here.

This is the end of creating an overlaid resident common. Now, aren't you glad that it needs be done only once?

Software Packages for VAX/VMS and RSTS/E.

Resource Accounting, Auditing and Billing VAX/VMS and RSTS/E.

RABBIT-1

RABBIT-1 is a general system for auditing, accounting and charging computer usage from the computer center to individual users, groups of users or customers. These may represent real accounts receivable, as in the case of a service bureau, or may be for budgeting and cross-charging in-house system resource consumption.

RABBIT-1 will accurately accumulate and distribute all system usage. This includes not only CPU utilization but DISK, I/O, Printing, Tape Storage, Equipment Rental, Manuals, Communication Costs, and Programming services.

A secondary purpose served by RABBIT-1 is to maximize the usage of the computer facility by encouraging users to distribute their utilization across as much machine availability as possible. Multi-level discounting facilities encourage the shifting of computer demand to off-prime time for more throughput at no additional costs.

System Performance Analysis VAX/VMS and RSTS/E.

RABBIT-2

If your system is suffering from slow response, clogged I/O, reduced throughput, then put RABBIT-2 on your case. RABBIT-2 locates the trouble spots in your operating system and identifies the source of the problems.

RABBIT-2 will chart your system performance, on an hour by hour, user by user, or program by program basis. RABBIT-2 will quickly sketch a profile of your average system day, your average user demands, and your average program resource requirements.

RABBIT-2 provides the tools you need to investigate system throughput in terms of CPU, I/O, memory, connect, KCTs, etc.—over any time period you specify. You can play "What if?" by simulating the removal of the offending program or user and displaying the results of the change.

Easy to use with menu driven reporting. You select options with easy, fill-in-the-blanks prompting.

Easy to install. No SYSGEN. Takes just minutes.

RABBIT-2 capabilities include:
- —Batch and interactive analysis
- —Interval or time displays
- —Resource consumption diagrams
- —User and program investigation
- —Graphic or numeric output
- —WHO was on WHEN
- —Profile analysis of users and programs
- —Ratios of resources utilized
- —Rankings of users and programs
- —Forecasting of future resource consumption

Job Accounting and Performance Monitoring RSTS/E Version 7.

RABBIT-3

RABBIT-3 is a complete performance monitoring and job accounting system designed especially for PDP-11 RSTS/E Version 7 users. Designed as a stand alone system, RABBIT-3 is written entirely in PDP macro assembler for maximum operating efficiency. Fast and small, RABBIT-3 runs in 5K core with only a 1% (approximate) system degredation depending on the sampling rate.

RABBIT-3 is flexible and easy to use. It's also easy to install. After loading the RABBIT-3 tape or disk, just answer a few questions to tailor the system to your needs. In less than an hour of effort, your RABBIT-3 will be generating complete, detailed user information.

Options for RABBIT-3 Users

Auto-Crash Recovery
—automatic restart from system crashes.

Daily Disk Catalog
—generates disk accounting information for each user.

Security Tracer Record
—provides step by step security information of user activities.

Disk Critical Alert
—alerts operator that free disk is below efficient levels.

Security System for RSTS/E Version 7.

RABBIT-4

RABBIT-4 is a stand-alone data file security system that operates under RSTS/E Version 7. RABBIT-4 prevents access to classified or confidential data fields by unauthorized personnel. Even priviledged users may be excluded from data files secured by RABBIT-4. Coupled with normal security measures, RABBIT-4 will ensure file integrity while monitoring all file access attempts.

RABBIT-4 "locks-up" designated data files from would-be intruders. All attempted file violations are logged and available for analysis. Computer operations and security management may be immediately notified of the file violation attempt, and the offending job rolled out or killed. Under RABBIT-4 maximum security, the RSTS operating system may be disabled until the intrusion is cleared.

RABBIT-4 provides up to 6 levels of security on a present maximum of 64 data files. Access to each secured data file is restricted only to authorized users through file access definition tables. Up to 32 wild card notations define the authorized users for each secured file. Access authorization may also be restricted to limited programs to ensure file integrity.

RABBIT-4 creates and maintains a complete log of file access and violation attempts. Included in the log is: date, time, job, program, project-programmer and keyboard. The log of file accesses is available at all times to assigned security management. A recap report of violations and access attempts may be run at will.

RABBIT-4 is written in PDP-11 macro to ensure optimized performance. The logged data is in ASCII stream format for prompt reporting. RABBIT-4 is controllable through OPSER and has an optional system crash recovery capability.

For more information on the RABBIT system contact

Telephone: (404) 955-2553
TWX: 810-766-2256

RAXCO INC.
6520 Powers Ferry Road
Suite 200
Atlanta, Georgia 30339, U.S.A.

CIRCLE 21 ON READER CARD

USING THE EXAMPLE

In order to use this common area, we will create a program that references the resident common area and the overlaid segments. The only requirements are that:

- the correct segment is overlaid (by calling it) prior to reference,
- in BASIC+2, the MAP name that is used must:
  - use the same name as the zero length segment in the root of the resident common,
  - use the FILL variable to offset individual record definitions into the overlaid segment.

STEP 1

The sample data description portion of a program may look like this:

```
Points to notice:           Sample BASIC+2 Program
---------------->          ----------------------

MAP names must be +->100    Map (OVRDAT)    ! Same for all areas
identical         |                 Contrl.Name$(2%)     = 6%
                  |         ,       Contrl.InUse%(2%)
                  |         ,       Contrl.Status%(2%)
                  |
                  |                 ...
                  |
                  +->200    Map (OVRDAT)
                                    Slot.Screen$         = 1000%
                            ,       Slot.Status%
                            ,       Slot.Etc%
```

Now the code part of the program is almost normal, but notice the calls to the NAMEd segments to force the remapping of the OVRDAT map.

```
                    1000    ! Main - line

                            ...

                    1100    ! Access Control Area
CALLs remap to new +->      Call CTRL00     ! Remap to control area
seqment            |        For I% = 0% to 2%
                   |           If Contrl.InUse%(I%)=0%
                   |           Then
                   |                User%            = I% + 1%
                   |                Contrl.InUse%(I%)=-1%
                   |                Contrl.Name$(I%) = ...
                   |        Next I%
                   |
                   |                ...
                   |
                   |  2000   ! Access a transaction slot
                   +->       Call SLOT01 If User% = 1%
                   +->       Call SLOT02 If User% = 2%
                   +->       Call SLOT03 if User% = 3%
                             If (Slot.Status% and 1%) <> 0%
                             Then
                                    ...
```

STEP 2

Compile your program and create a command file that has an additional line in it (or add the additional line if you use the dialogue mode of TKB):

```
                    Sample Command File
                    -------------------

                    SY:PROGB,SY:PROGB=SY:PROGB/MP
                    UNITS=12
                    ASG=SY:5:6:7:8:9:10:11:12
                    RESLIB=LB:BASICS/RO
Additional line ->  RESCOM=[1,53]OVRDAT/RW
                    EXTTSK=1532
                    //
```

At this point, you should get several diagnostic messages from the task builder. There will be one for each different map statement in the source modules. The message informs you that the PSECT reference conflicts with the PSECT in the library OVRDAT. This is to be expected, because the library was built using the PSECT named OVRDAT with a zero length associated with it, and you have just said that it is 8192 bytes long. Now what happens is that you get your task built properly, because TKB uses the starting address of the resident common as the starting address of your PSECT and as a result your variables are now stored in the resident common. If you want further details feel free to give me a call.

ELEPHANT TRAPS

There are several areas that you may term "elephant traps," because they are large holes that you can fall into and never be heard from again. They are the following:

- Remember it's shared, someone can get your MAPs,
- "Sophomores"(semaphores) are essential,
- Beware of single threaded bottlenecks,
- Check overlay segment length carefully,
- You lose chunks of address space (APRs).

SHARED DATA

In the usual programming environment, you know that when you put information into a variable, it doesn't change until you put something else in it. When you share data through resident common areas, remember that all variables in the common area can be changed by another program when your back is turned. Therefore, the protocol for using resident common must be carefully defined prior to implementation. Two possible forms are:

- use of flags in a control area
- use of slots that are unique to one user/transaction.

USE OF SEMAPHORES

You should consider carefully the correct use of semaphores for one program to be able to signal another program about significant events. RSTS/E does not have many system services for allowing different jobs to synchronize their activities. There is Send/Receive (with sleep) or forced data into psuedo keyboards that will wake up a sleeping job. Otherwise, you have to design a flag setting and sleep n seconds arrangement between jobs. One thing to be VERY wary of is SUSPENDing and RESUMEing jobs. It is easy to have both jobs SUSPENDed at the same time.

SINGLE THREADED BOTTLENECKS

DO NOT accidently create another FIP. A single threaded server is a program that will process one request to completion before processing the next request on the queue. When there are only two or three users of disk intensive server that is single threaded, you will see rapid system degradation. Therefore, design any single threaded server with a careful eye to the amount of disk activity that it must perform to finish a request.

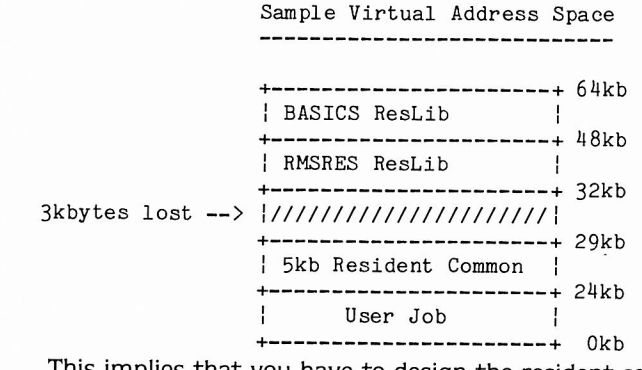
OVERLAY SEGMENT LENGTH

One problem in using overlaid segments with BASIC+2 is that you can not be sure that your MAP does not exceed the actual length of an overlay segment. This is because of the deliberate introduction of a possible conflict that TKB warns you about. The choices are two, either:

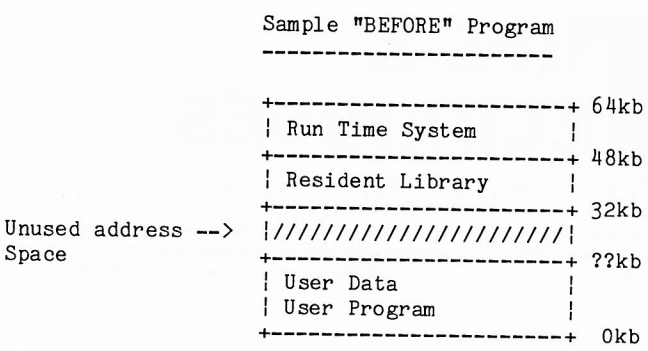
- Make ALL overlay segments reserve 8.kbytes, which will rapidly inflate the physical memory requirements of a resident common region, sometimes even double, or
- Verify the length of a PSECT in your program's MAP against the length of an overlay segment in the resident common area.

LOSS OF ADDRESS SPACE

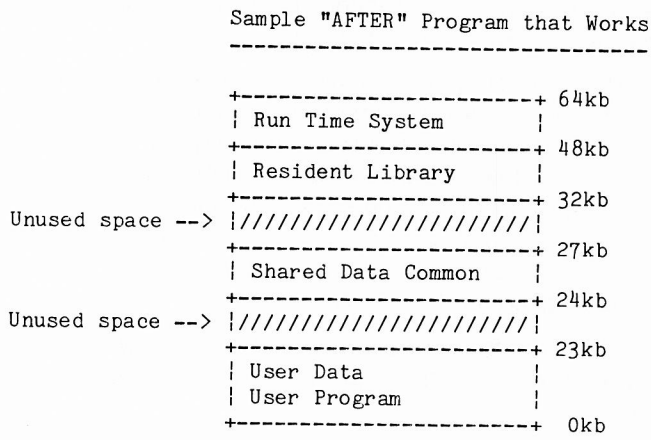
The PDP-11 allows you to divide your virtual address space into eight pages. Each page may be from 64 bytes to 8192 bytes in size. Each page also starts at an 8kbyte boundary. This feature is used by RSTS/E to separately load and control run time systems, resident libraries and programs that are all part of the same job. However, if you use only a portion of one page (e.g., a 5kbyte resident common area) you lose access to the remaining portion of the virtual addresses of that page (e.g., 3kbytes are unavailable). For example,



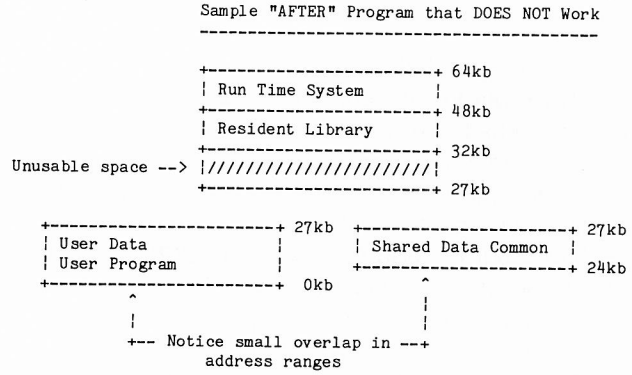
This implies that you have to design the resident common area balancing the advantage of shareable data against the possible loss of virtual address space.



This program will only be able to use a resident common if combined program size and non-shared data is less than the current available memory size minus the size (rounded up to 8kbytes) of the resident common area. For example, if the program is 26kbytes prior to splitting it into 3kbytes of shared data and 23kbytes words of program and non-shared data, then you could do it. It would look like this:



However, if you had a program that was already 30kbytes and you want to share 3kbytes of data you are left with 27kbytes of program and non-shared data. If your program currently uses 32kbytes worth of resident libraries (i.e., BASICS and RMSRES), you will have to resort to overlaying RMS (not a good thing) or use some other shoe-horning if you will insist on trying to use the resident common area.



On the other hand, if it allows you to cleanly break your program into two or three smaller pieces, maybe it is not a bad idea after all.

FINAL NOTE

JUST BE VERY CAREFUL ABOUT ALL THIS GOOD STUFF. YOU SHOULD LEARN HOW TO USE IT BEFORE YOU COMMIT YOURSELF TOO FAR. IT IS VERY USEFUL, BUT EVERYTHING HAS ITS COST. ♥

TIPS & TECHNIQUES

A Column For The Advanced RSTS/E User

Steven L. Edwards, Software Techniques, Inc.

MONITORING FREE DISK SPACE

For production systems, the worst thing that can happen is that the operating system crash. The second worst thing that can happen is that you run out of disk space. This invariably happens five minutes before your all night batch job completes, or after your data entry operators have spent five hours entering transactions into a temporary file. The consequences are rarely convenient.

The ideal solution would be for the operating system to stall that job until the system manager can free up some more disk space. The ideal solution is beyond the scope of this column. However, we can set up a job that will monitor the free disk space, and do something about it.

The program presented here allows the system manager to establish a 'yellow' limit and a 'red' limit for each disk device on the system.

When the free space falls below the 'yellow' limit, the program sends a message to the operator's service console via OPSER.

When the free space falls below the 'red' limit, the program sends a message to the operator's service console. In addition, the program then suspends all jobs, and sets all active terminals to 'SPEED 0' to prevent unintended type-ahead. The program then suspends itself. When the system manager frees up sufficient disk space, s/he can resume this program which will restore all of the jobs it tampered with.

Please note the following:

1. This program has only been tried on two systems, both of which had two RM02 type disks. Be prepared to fix any bugs you may uncover.
2. Entering a value of 0, or just typing <RETURN> to a limit question effectively disables that limit.
3. This program assumes job 1 is ERRCPY and job 2 is OPSER. It does not suspend jobs 1 and 2.
4. This program assumes that you do not want to set KB0 to 'SPEED 0' in case a 'red' limit is exceeded.
5. If you have malicious users, do not establish 'red' limits for any disk they can create files on.
6. If your programs extend files that have clustersizes greater than the pack clustersize, you must ensure that there are free clusters of the size required above the limits set.
7. This program will not catch directory extension failures.

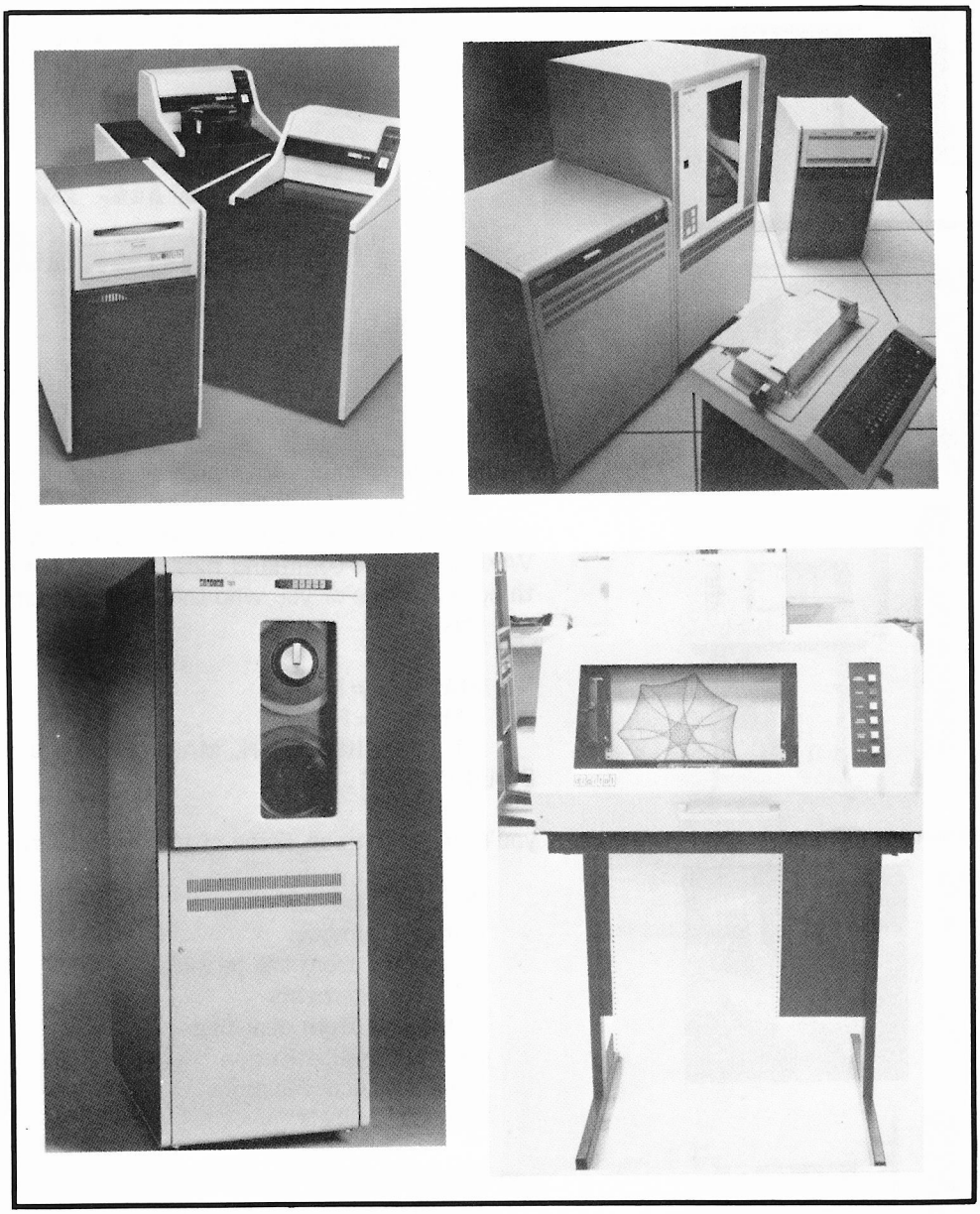
```
1!      Title:          F R E M O N
!
!       Version:        Y0101A
!
!       Edit date:      02-MAY-1983
!
!       Written by:     STEVEN L. EDWARDS
!
!       Date:           17-JAN-1983
!
!       Package:        In-House
!
!       Description:    MONITOR FREE DISK SPACE

11!     Copyright (C) 1983
!       Software Techniques
!       Los Alamitos, CA 90720
!
!       Title  to and ownership of the software shall at all times remain
!       in Software Techniques.
!
!       The  information  in  this  document is subject to change without
!       notice  and  should  not be construed as a commitment by Software
!       Techniques.
!
!       This  software  is  un-released  and  Software  Techniques has no
!       commitment to support it at this time, unless stated elsewhere in
!       writing.

20!
!               Modification History
!
!       Ver/Edit        Date            Reason (Who)
!       --------        ----            ------------

21!     X0101A          17-JAN-1983     Initial conception.
!                                       (SLE)
!
!       X0102A          02-MAY-1983     CHANGE THE SIZE OF THE LIMIT ARRAYS.
!                                       CHANGE METHOD OF DETERMINING WHEN TO
!                                         SEND A MESSAGE TO OPSER.
!                                       MINOR CODE CLEAN-UPS.
!                                       (SLE)
!
!       Y0101A          02-MAY-1983     PROMOTE TO GROUP SUPPORT.
!                                       (SLE)

100!
!               Program Description
!

101!            THIS PROGRAM MONITORS THE FREE SPACE ON ALL DISKS ON THE
!       SYSTEM.
!
!       WHEN THE FREE SPACE FALLS BELOW THE 'YELLOW' LIMIT, THE PROGRAM
!       SENDS A MESSAGE TO THE OPERATOR'S SERVICE CONSOLE VIA OPSER.
!
!       WHEN THE FREE SPACE FALLS BELOW THE 'RED' LIMIT, THE PROGRAM
!       SENDS A MESSAGE TO THE OPERATOR'S SERVICE CONSOLE VIA OPSER. THE
!       PROGRAM THEN SUSPENDS ALL JOBS AND SETS ALL ACTIVE TERMINALS TO
!       SPEED 0 TO PREVENT UNINTENDED TYPE-AHEAD. WHEN THE OPERATOR
!       FREES UP SUFFICIENT DISK SPACE, S/HE CAN RESUME THIS PROGRAM
!       WHICH WILL RESTORE ALL OF THE JOBS IT TAMPERED WITH.

500!
!               Compile time variables
!

501     .DEFINE .NAME$ = "Fremon"
\       .DEFINE .VERSION$ = "Y01.01A"                                   !Y0101A
\       .DEFINE .CHAN.KB% = 1%
!
!       Program Name.
!       Program Version.
!       Channel number for terminal I/O.

900!
!               Dimension Declaration
!
! 901-929 local dimension declarations
! 930-949 library dimension declarations
! 950-979 MAP statements

901     DIM
                PRIORITY%(63%)
               ,RED_LIMIT(80%)                                          !X0102A
               ,TERMINAL_SPEED%(128%)
               ,YELLOW_LIMIT(80%)                                       !X0102A
!
!       JOB PRIORITY ARRAY.
!       RED LIMIT ALARM ARRAY.
!       TERMINAL SPEED ARRAY (RECEIVE,TRANSMIT).
!       YELLOW LIMIT ALARM ARRAY.
```

```
902     MAP     (UUTB1) !
                        ! DATA BLOCK FOR MONITOR TABLES PART I
                        !
                 JOBX2_FILL%    ! JOB NUMBER TIMES 2 / FILL BYTE
                ,CNT_KB_MAXCNT% ! MAX KB NUMBER / MAX JOB NUMBER
                ,DEVCNT%        ! TABLE OF MAX UNIT NUMBER
                ,DEVPTR%        ! TABLE OF POINTERS TO DEVICE DDB's
                ,FILL%          ! FILLER
                ,JOBTBL%        ! JOB TABLE
                ,FILL$ = 6%     ! FILLER
                ,UNTCNT%        ! TABLE OF UNIT STATUS / OPEN COUNT
                ,SATCTL%        ! TABLE OF FREE BLOCK COUNTS (LSW)
                ,FILL%          ! FILLER
                ,SATCTM%        ! TABLE OF FREE BLOCK COUNTS (MSW)
                ,FILL$ = 4%     ! FILLER
                                !
\       MAP     (UUTB1) !
                        ! REMAP THE MONITOR TABLES I DATA BLOCK
                        ! TO BE ABLE TO ACCESS ENTIRE THING AT ONCE
                        !
                 UUTB1$ = 30%   ! UUTB1$ = SYS(CHR$(6%) + CHR$(-3%))
                                !
\       MAP     (UUTB2) !
                        ! DATA BLOCK FOR MONITOR TABLES PART II
                        !
                 FILL$ = 4%     ! FILLER
                ,DEVNAM%        ! DEVICE NAME TABLE
                ,FILL%          ! CSR TABLE
                ,DEVOKB%        ! NUMBER OF DISKS TIMES 2
                ,FILL$ = 10%    ! FILLER
                ,LOGNAM%        ! TABLE OF SYSTEM LOGICAL NAMES
                ,FILL$ = 8%     ! FILLER
                                !
\       MAP     (UUTB2) !
                        ! REMAP THE MONITOR TABLES PART II DATA BLOCK
                        ! TO BE ABLE TO ACCESS ENTIRE THING AT ONCE
                        !
                 UUTB2$ = 30%   ! UUTB2$ = SYS(CHR$(6%) + CHR$(-12%))
                                !

9991
!               Start of Initialization
!

1000    ONERROR GOTO 19000
!
!       Set standard error trap.

1010    PRINT .NAME$ + HT + .VERSION$ + HT + "Software Techniques"
              + CR + LF + "Free disk space monitor." + CR + LF
              UNLESS E0%
!
!       Print standard header on 'Run' entry.

1030    UUTB1$ = SYS(CHR$(6%) + CHR$(-3%))
\       UUTB2$ = SYS(CHR$(6%) + CHR$(-12%))
\       CNT_KB% = CNT_KB_MAXCNT% AND 255%
\       MAXCNT% = SWAP%(CNT_KB_MAXCNT%) AND 255%
\       OUR_JOB_NUMBER% = (JOBX2_FILL% AND 255%) / 2%
!
\       NULL_9$ = STRING$(9%, 0%)                               !X0102A
\       NULL_10$ = STRING$(10%, 0%)                             !X0102A
\       PASS_INTERVAL% = 30%                                    !X0102A
\       YELLOW_COUNT%, YELLOW_1%, YELLOW_2% = 0%                !X0102A
!
!       GET MONITOR TABLES - PART 1.
!       GET MONITOR TABLES - PART 2.
!       GET THE NUMBER OF TERMINALS ON THIS SYSTEM.
!       GET THE MAXIMUM JOB NUMBER.
!       GET OUR JOB NUMBER.
!
!       A FEW NULLS.                                            X0102A
!       A FEW MORE NULLS.                                       X0102A
!       INITIALIZE OUR PASS INTERVAL.                           X0102A
!       INITIALIZE THE YELLOW MESSAGE COUNTERS.                 X0102A
!
!       Define various variables.

1110    OPEN "KB:KB.IO" FOR OUTPUT AS FILE #.CHAN.KB%
              ,MODE 0%
!
!       Open the terminal.

2000!
!               Start of MAIN
!

2010    UNTCNT_POINTER% = UNTCNT%
\       FOR DEVICE_INDEX% = 0% TO (DEVOKB% - 2%) STEP 2%
\               FOR UNIT_NUMBER% = 0% TO PEEK(DEVCNT% + DEVICE_INDEX%)
\                       UNIT_STATUS% = PEEK(UNTCNT_POINTER%)
\                       GOTO 2020
                                IF      ((UNIT_STATUS% < 0%)
                                OR      ((UNIT_STATUS% AND 4096%) <> 0%))
\                       DSK$ = CVT%$(SWAP%(PEEK(DEVNAM% + DEVICE_INDEX%)))
                                + NUM1$(UNIT_NUMBER%) + ":"
\                       SUB_SCRIPT% = (DEVICE_INDEX% * 4%) + UNIT_NUMBER%
\                       PRINT #.CHAN.KB%, "Enter YELLOW limit for "
                                + DSK$ + SP;
\                       INPUT #.CHAN.KB%, YELLOW_LIMIT(SUB_SCRIPT%)
\                       PRINT #.CHAN.KB%, "Enter RED    limit for "
                                + DSK$ + SP;
\                       INPUT #.CHAN.KB%, RED_LIMIT(SUB_SCRIPT%)

2020                    UNTCNT_POINTER% = UNTCNT_POINTER% + 2%
\               NEXT UNIT_NUMBER%
\       NEXT DEVICE_INDEX%
!
!       MAKE SURE WE ARE USING THE CORRECT POINTER VALUES.
!       LOOP THROUGH THE MONITOR'S TABLES OF DEVICES.
!               LOOP THROUGH ALL UNIT NUMBERS.
!                       SKIP DEVICES THAT ARE NOT MOUNTED OR ARE NFS.
!                       CONSTRUCT THE DEVICE NAME.
!                       CALCULATE THE PROPER SUBSCRIPT.
!                       ENTER THE YELLOW LIMIT FOR THIS DEVICE.
!                       ENTER THE RED LIMIT FOR THIS DEVICE.
!                       BUMP THE POINTER.
!               TRY THE NEXT UNIT NUMBER.
!       TRY THE NEXT DEVICE.

2999    PRINT #.CHAN.KB%; CR + LF
              + "Detaching..."
              + CR + LF
              + CR + LF
              + CR + LF
              + CR + LF
\       CLOSE #.CHAN.KB%
\       TEMP_0$ = SYS(CHR$(6%) + CHR$(7%))
!
!       TELL THE USER WHAT WE ARE ABOUT TO DO.
!       CLOSE THE TERMINAL CHANNEL.
!       DETACH FROM THE TERMINAL.

3000!
!               MAIN PROGRAM LOOP
!

3010    LOGNAM_POINTER% = LOGNAM%
\       SATCTL_POINTER% = SATCTL%
\       SATCTM_POINTER% = SATCTM%
\       UNTCNT_POINTER% = UNTCNT%
\       FOR DEVICE_INDEX% = 0% TO (DEVOKB% - 2%) STEP 2%
\               FOR UNIT_NUMBER% = 0% TO PEEK(DEVCNT% + DEVICE_INDEX%)
\                       UNIT_STATUS% = PEEK(UNTCNT_POINTER%)
\                       GOTO 3020
                                IF      ((UNIT_STATUS% < 0%)
                                OR      ((UNIT_STATUS% AND 4096%) <> 0%))
\                       DSK$ = CVT%$(SWAP%(PEEK(DEVNAM% + DEVICE_INDEX%)))
                                + NUM1$(UNIT_NUMBER%) + ":"
                                + RAD$(PEEK(LOGNAM_POINTER%))
                                + RAD$(PEEK(LOGNAM_POINTER% + 2%))
\                       SUB_SCRIPT% = (DEVICE_INDEX% * 4%) + UNIT_NUMBER%
\                       FREE_BLOCKS = PEEK(SATCTL_POINTER%)
\                       FREE_BLOCKS = FREE_BLOCKS + 65536.0
                                IF      (FREE_BLOCKS < 0.0)
\                       FREE_BLOCKS = FREE_BLOCKS + 65536.0 * PEEK(SATCTM_POINTER%)
\                       IF      FREE_BLOCKS < RED_LIMIT(SUB_SCRIPT%)
                        THEN    GOSUB 5000
                        ELSE    IF      FREE_BLOCKS < YELLOW_LIMIT(SUB_SCRIPT%)
                                THEN    YELLOW_COUNT% = YELLOW_COUNT% + 1%
                                        GOSUB 4000
\
3020                    LOGNAM_POINTER% = LOGNAM_POINTER% + 10%
\                       UNTCNT_POINTER% = UNTCNT_POINTER% + 2%
\                       SATCTL_POINTER% = SATCTL_POINTER% + 2%
\                       SATCTM_POINTER% = SATCTM_POINTER% + 2%
\               NEXT UNIT_NUMBER%
\       NEXT DEVICE_INDEX%
!
!       MAKE SURE WE ARE USING THE CORRECT POINTER VALUES.
!       LOOP THROUGH THE MONITOR'S TABLES OF DEVICES.
!               LOOP THROUGH ALL UNIT NUMBERS.
!                       SKIP DEVICES THAT ARE NOT MOUNTED OR ARE NFS.
!
```

```
!                       CONSTRUCT THE DEVICE NAME.
!                       CALCULATE THE PROPER SUBSCRIPT.
!                       GET THE NUMBER OF FREE BLOCKS ON THIS UNIT.
!                       CHECK FOR A RED LIMIT VIOLATION.
!                       CHECK FOR A YELLOW LIMIT VIOLATION.
!                       BUMP THE VARIOUS POINTERS.
!               TRY THE NEXT UNIT NUMBER.
!       TRY THE NEXT DEVICE.

3030    IF      YELLOW_COUNT%                                           !X0102A
        THEN    YELLOW_1% = YELLOW_1% + 1%                              !X0102A
        ELSE    YELLOW_1%, YELLOW_2% = 0%                               !X0102A
!                                                                       X0102A
!       IF      WE GOT ANY YELLOW LIMIT VIOLATIONS,                     X0102A
!       THEN    INCREMENT THE "DON'T BOTHER OPSER" FLAG,                X0102A
!       ELSE    CLEAR THE FLAGS.                                        X0102A

3999    YELLOW_COUNT% = 0%                                              !X0102A
\       SLEEP PASS_INTERVAL%                                            !X0102A
\       GOTO 3000
!
!       SLEEP FOR A LITTLE WHILE.
!       BRANCH BACK TO TRY AGAIN.

4000!
!               YELLOW LIMIT EXCEEDED
!

4010    IF      YELLOW_2% = 0%                                          !X0102A
        THEN    MESSAGE$ = "Yellow limit exceeded on " + DSK$           !X0102A
                         + BEL + BEL                                    !X0102A
\               GOSUB 10100                                             !X0102A
\               YELLOW_2% = YELLOW_1%                                   !X0102A
        ELSE    YELLOW_2% = YELLOW_2% - 1%                              !X0102A
!                                                                       X0102A
!       SEND A MESSAGE (VIA OPSER) IF WE HAVEN'T LATELY.                X0102A

4999    RETURN
!
!       BACK TO MAINLINE CODE.

5000!
!               RED LIMIT EXCEEDED
!

5010    MESSAGE$ = "Red limit exceeded on " + DSK$ + BEL + BEL
\       GOSUB 10100
!
!       SEND A MESSAGE (VIA OPSER).

5020    PRIORITY%(INDEX%) = -1%
                FOR INDEX% = 1% TO MAXCNT%
\       TERMINAL_SPEED%(INDEX%) = 0%
                FOR INDEX% = 1% TO CNT_KB%
\       JOB_NUMBER% = 3%
!
!       INITIALIZE THE PRIORITY ARRAY.
!       INITIALIZE THE TERMINAL SPEED ARRAY.
!       START AT JOB 3 (ASSUMING JOB 1 IS ERRCPY AND JCB 2 IS OPSER).

5030    TEMP_0% = PEEK(JOBTBL% + (JOB_NUMBER% * 2%))
\       GOTO 5040
                IF      TEMP_0% = 0%
\       GOTO 5050
                IF      TEMP_0% = -1%
\       PRIORITY%(JOB_NUMBER%) = PEEK(TEMP_0% + 28%) AND 255%
\       TEMP_0$ = SYS(CHR$(6%) + CHR$(-13%) + CHR$(JOB_NUMBER%)
                + CHR$(255%) + CHR$(-128%))
                        UNLESS  (JOB_NUMBER% = OUR_JOB_NUMBER%)
!
!       GET THE ADDRESS OF THE JOB DATA BLOCK FOR THE JOB.
!       SKIP THE JOB IF IT DOES NOT EXIST (0).
!       QUIT IF WE ARE AT THE END OF THE JOB TABLE (-1).
!       SAVE THE PRIORITY OF THE JOB.
!       SUSPEND THE JOB.

5040    JOB_NUMBER% = JOB_NUMBER% + 1%
\       GOTO 5030
!
!       BUMP THE JOB NUMBER.

5050    TEMP_0% = PEEK(DEVPTR% + DEVOKB%)
\       FOR INDEX% = 1% TO CNT_KB%
\               TEMP_1% = PEEK(PEEK(TEMP_0% + (INDEX% * 2%)) + 2%)
\               JOB_NUMBER% = TEMP_1% AND 255%
\               GOTO 5060
                        IF      ((JOB_NUMBER% = 0%)
                        OR      ((JOB_NUMBER% AND 1%) = 1%))
\               UNIT_NUMBER% = SWAP%(TEMP_1%) AND 255%
\               TEMP_0$ = SYS(CHR$(6%) + CHR$(-4%) + CHR$(UNIT_NUMBER%)
                        + CHR$(19%))
\               TEMP_0$ = SYS(CHR$(6%) + CHR$(16%) + CHR$(0%)
                        + CHR$(UNIT_NUMBER%))
\               TERMINAL_SPEED%(UNIT_NUMBER%) = (ASCII(MID(TEMP_0$, 14%, 1%))
                        OR SWAP%(ASCII(MID(TEMP_0$, 16%, 1%))))
\               TEMP_0$ = SYS(CHR$(6%) + CHR$(16%) + CHR$(0%)
                        + CHR$(UNIT_NUMBER%) + NULL_9$                  !X0102A
                        + CHR$(1%) + CHR$(0%) + CHR$(1%))

5060    NEXT INDEX%
!
!       STEP THROUGH THE TERMINAL DEVICE DATA BLOCKS, SETTING THE SPEED
!       OF ALL TERMINALS OWNED BY A JOB TO ZERO.  THIS PREVENTS THE
!       USER'S FROM FILLING THEIR TYPE-AHEAD BUFFERS FULL OF CHARACTERS
!       THAT THEY REALLY DON'T WANT.  NOTE THAT WE ALSO SKIP KB0:.

5500    TEMP_0$ = SYS(CHR$(6%) + CHR$(-13%) + CHR$(OUR_JOB_NUMBER%)
                + CHR$(255%) + CHR$(-128%))
\       FOR UNIT_NUMBER% = 1% TO CNT_KB%
\               TEMP_0% = TERMINAL_SPEED%(UNIT_NUMBER%)
\               GOTO 5510
                        IF      (TEMP_0% = 0%)
\               TEMP_0$ = SYS(CHR$(6%) + CHR$(16%) + CHR$(0%)
                        + CHR$(UNIT_NUMBER%) + NULL_9$                  !X0102A
                        + CHR$(TEMP_0% AND 255%) + CHR$(0%)
                        + CHR$(SWAP%(TEMP_0%) AND 255%))
\               TEMP_0$ = SYS(CHR$(6%) + CHR$(-4%) + CHR$(UNIT_NUMBER%)
                        + CHR$(17%))

5510    NEXT UNIT_NUMBER%
\       FOR JOB_NUMBER% = 1% TO MAXCNT%
\               TEMP_0% = PRIORITY%(JOB_NUMBER%)
\               TEMP_0$ = SYS(CHR$(6%) + CHR$(-13%) + CHR$(JOB_NUMBER%)
                        + CHR$(255%) + CHR$(TEMP_0%))
                        UNLESS  ((TEMP_0% = -1%)
                        OR       (JOB_NUMBER% = OUR_JOB_NUMBER%))
\       NEXT JOB_NUMBER%
!
!       SUSPEND OURSELVES.
!       RESUME EVERYONE TO THEIR INITIAL VALUES.

5999    RETURN
!
!       BACK TO MAINLINE CODE.

9999    GOTO 32700
!
!       Prevent fall through errors.

10000!
!               Local Subroutines
!

10100!
!               SEND A MESSAGE TO OPSER
!

10110   IF      LEN(MESSAGE$) <= 19%
        THEN    TEMP_0$ = SYS(CHR$(6%) + CHR$(22%) + CHR$(-1%)
                        + CHR$(0%) + "OPSER" + SP + NULL_10$            !X0102A
                        + CHR$(LEN(MESSAGE$) + 1%) + MESSAGE$)
        ELSE    TEMP_0$ = SYS(CHR$(6%) + CHR$(22%) + CHR$(-1%)
                        + CHR$(0%) + "OPSER" + SP + NULL_10$            !X0102A
                        + CHR$(255%) + LEFT(MESSAGE$, 19%))
\               MESSAGE$ = RIGHT(MESSAGE$, 20%)
\               GOTO 10110
!
!       SEND THE DATA TO 'OPSER'.

10199   RETURN
!
!       BACK TO MAINLINE CODE.

19000!
!               Error Handler
!

19004   IF      ERR = 4%
        THEN    IF      ERL = 10110%
                THEN    SLEEP 1%
\                       RESUME 10110
!
!       NO ROOM FOR USER ON DEVICE WHILE TALKING TO OPSER.

19011   IF      ERR = 11%
        THEN    RESUME 32700
!
!       End of file on device.

19032   IF      ERR = 32%
        THEN    IF      ERL = 10110%
                THEN    SLEEP 1%
\                       RESUME 10110
!
!       NO BUFFER SPACE AVAILABLE WHILE TALKING TO OPSER.

19999   ONERROR GOTO 0
!
!       Give up.

32700!
!               Completion Routines
!

32710   CLOSE #1%
!
!       Close all channels.

32767   END
```

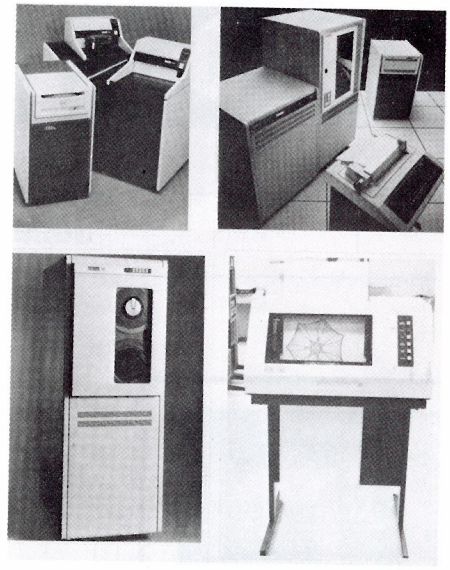
The VAX-SCENE

Number 14 (RSTS PROFESSIONAL, Vol. 5, No. 3) **June 1983**



INSIDE:

Using VAX Command Procedures (or Your File is My Command)



USING VAX COMMAND PROCEDURES (or, Your File is My Command)

By Bob Meyer

Greetings, again, VAX and/or MACRO fans . . .

Being just about the laziest person on the planet, I recently discovered the joys of writing command files on the VAX. Wow! The command files under DCL are really something. For those of you who thought command files looked like this:

```
MAC FILE=FILE
TKB FILE=FILE
UT SEND KBXX: OH, MACRO PERSON, YOUR TASKBUILD IS DONE . . .
```

you're in for a treat. Some of it's many features are:

- String symbol definition
- Numeric symbols
- I/O to and from the terminal
- If-Then statements
- I/O to and from disk files
- Parameter substitution
- Build in lexical functions
- Gotos and labels

and lots more that I haven't even learned yet.

The first use I had for writing command files was a simple list of assembly and link instructions for a small program:

```
MAC BOB
LINK BOB
RUN BOB
```

RSTS/E INTERNALS MANUAL

The RSTS community has been clamoring for years for a book that details the inner workings of RSTS/E. Well, clamor no more. Michael Mayfield of Northwest Digital Software, and M Systems, the publisher of The RSTS Professional and The DEC Professional Magazines, have teamed up to produce the RSTS/E Monitor Internals Manual.

This manual describes the internal workings and data structures of the RSTS/E monitor. It also notes differences in the internal structures between version 7.1 and earlier versions of the monitor. Future updates will include changes for new versions of the monitor.

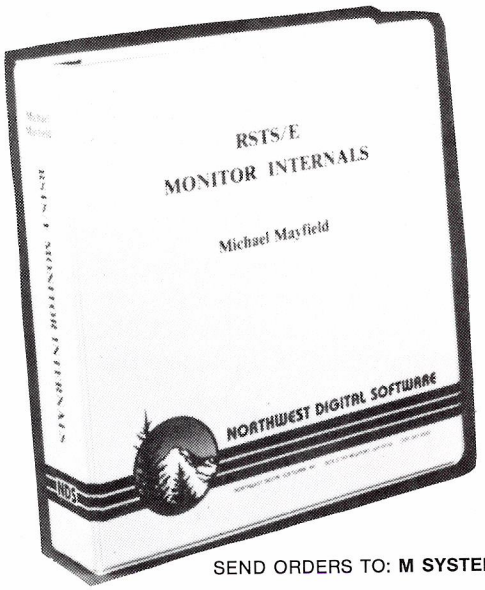
Information is available for all levels of users:

- Gain a basic understanding of the workings of the monitor for optimizing system performance.
- Information on disk structures allows recovery of data from corrupted disk packs.
- Special uses of runtime systems and resident libraries allow complex applications to be developed without degrading system performance.
- Write your own custom device drivers for that "foreign" device you need to add but thought you couldn't.

CONTENTS:

Chapter 1 describes the structures used by the monitor that are resident on disk. These include the directory structure, disk allocation tables, Save Image Library (SIL) formats, bootstrap formats and bad block mapping.

Chapter 2 describes the tables used within the monitor to control system resources and provide program services. These tables provide job, memory, file and device control, as well as program services such as interjob communication.



Chapter 3 contains information on writing and installing a custom device driver. It describes the entry points and information the driver must provide to the monitor as well as the subroutines and macros the monitor provides for the driver.

Chapter 4 contains information that enhances information already provided by Digital on writing custom resident libraries and runtime systems. It concentrates mainly on non-standard uses of resident libraries and runtime systems to increase system performance and functionality.

Appendix A provides six quick reference foldout charts:

- The directory structure.
- The monitor tables.
- Fixed memory locations and common data structures.
- Monitor subroutines.
- Device driver entry points.
- Device driver macros.

Appendix B provides examples of the peek sequences required to access most of the monitor tables. It also contains an example program that uses many of the monitor tables to display a job and open files status.

Appendix C provides an example device driver.

Appendix D provides an example runtime system that doubles as a menu system for restricting specified users to a menu of options.

2ND EDITION FOR 8.0 NOW ON PRESS

$95.00

SEND ORDERS TO: **M SYSTEMS, INC., BOX 361, FORT WASHINGTON, PA 19034-0361**

Soon I learned that the same command file could be used for other assemblies as well, simply by replacing the file names with a parameter symbol: (the dollar signs in the command files are required, and seem to help DCL determine data from commands)

```
$ MAC 'P1'
$ LINK 'P1'
$ RUN 'P1'
```

Where P1 is the first parameter I type after activating the command file: (assuming the cmd file is named 'ASM.COM')

```
@ASM BOB
```

DCL will then replace all occurrences of 'P1' with the string 'BOB' as it executes the file. Up to eight parameters may be specified, separated by spaces. Using single quotes around a string specify that the string should be substituted with something else; in the case of P1 thru P8, the strings are replaced with parameters from the user's command line activating the command file. An example of substitution would be:

```
$INQUIRE FILE "Enter file to assemble"
$MAC 'FILE'
$LINK 'FILE'
$RUN 'FILE'
```

When this command file is executed, DCL will prompt the user with the string specified in the INQUIRE command, and take input from the terminal (actually, from SYS$INPUT). That input will be assigned to the symbol 'FILE':

```
@ASM
Enter file to assemble: BOB
$ MAC BOB
$ LINK BOB
$ RUN BOB
```

We can avoid seeing all this on the screen by adding the line:

```
$SET NO VERIFY
```

to the command file. (This is often used in LOGIN.COM) Comments can be easily inserted in command files:

```
$ !
$ ! command file to assemble & link programs
$ !
$ SET NO VERIFY
$ INQUIRE FILE 'File'
$ MACRO FILE
$ LINK FILE
$ EXIT
```

The manuals recommend typing out commands in full, in an effort to make them more readable.

Several string operators are available; for example:

```
$ IF P1 .EQS. "" THEN GOTO GET—INPUT
$ FILE:='P1'
$ GOTO DOIT
$ GET—INPUT:
$ INQUIRE FILE 'File'
$ DOIT:
$ MACRO FILE
$ LINK FILE
$ EXIT
```

In this case, if parameter 1 (P1) was not specified, the command processor will transfer control to the line following the label GETINPUT:. If, however, a string was supplied when invoking the command file, the symbol 'FILE' will be assigned the string, and the command file will branch around the INQUIRE statement and begin the assembly and link process.

Some other string operators are:

```
.GES.  String greater than or equal to
.GTS.  Greater than
.LES.  Less than or equal to
.LTS.  Less than
.NES.  Not equal to
```

Several math operators are also at your disposal:

```
.EQ.  Equal to
.GE.  Greater than or equal to
.GT.  Greater than
.LE.  Less than or equal to
.LT.  Less than
.NE.  Not equal
```

Also the logicals: .OR., .AND., .NOT. and faithful +,-,*,/.

As an example of the arithmetic operators, let's assume we need to execute a program, a predetermined number of times:

```
$ COUNT=0
$ RUN:
$ RUN DOODAH.EXE
$ COUNT=COUNT+1
$ IF COUNT .GE. 5 THEN GOTO EXIT
$ GOTO RUN
$ EXIT:
$ EXIT
```

String symbols can be used like RSTS CCL's on a user-defined level. For example, the RSTS user 'migrating' to VAX might add something to his/her LOGIN.COM file (which gets executed every time you log in) like this:

```
$ SY:=SHOW SYSTEM
```

The ':=' is a string equate; whenever the string 'SY' is typed to DCL, it is replaced with 'SHOW SYSTEM' and the user gets the VMS equivlant of a RSTS systat.

Some other examples of practical use of the command language were pointed out to me by my comrade Bernie Velivis, system manager of Squibb's Molecular Modeling Graphics Lab in Princeton, NJ. Some of these include:

TE*CO:= =SYSSYSTEM:TECO.EXE/VT/HOLD

The '*' (I can't spell asterisk...) is similar to the '-' (I CAN spell dash) in RSTS's CCL structure; it acts as an ?abbreviation point in the command string, allowing the user to type TE, TEC, or TECO to invoke TECO. The '$' before SYS$SYSTEM: tells VMS to execute the image 'TECO.EXE' and pass to it the parameters '/VT/HOLD' as well as any parameters the user might have typed (such as a file name to edit).

Another interesting feature is the ability to perform I/O to and from disk files. Note the following command file:

```
$ OPEN
  INPUT 'P1'
$ READ:
$ READ
  INPUT
  REC/END
  =DONE
$ WRITE
  SYS$OUT
  PUT REC
$ GOTO READ
$ DONE:
$ CLOSE
  INPUT
$ WRITE
SYS$OUTPUT "[End of file]"
$ EXIT
```

First we open the file specified ('P1') when calling the file. Each time we encounter the READ statement, we get one line from the input file. The input record is assigned to the symbol 'REC'. Then we write the symbol to the terminal (SYS$OUTPUT), and loop. When we hit eof on the input file, we take the error path specified in the read statement (/END=DONE) and control is transferred to the label DONE:. We then close the input file, display a message saying we've reached end of file, and exit. (Reminds me of the good ol' days of BASIC PLUS . . .)

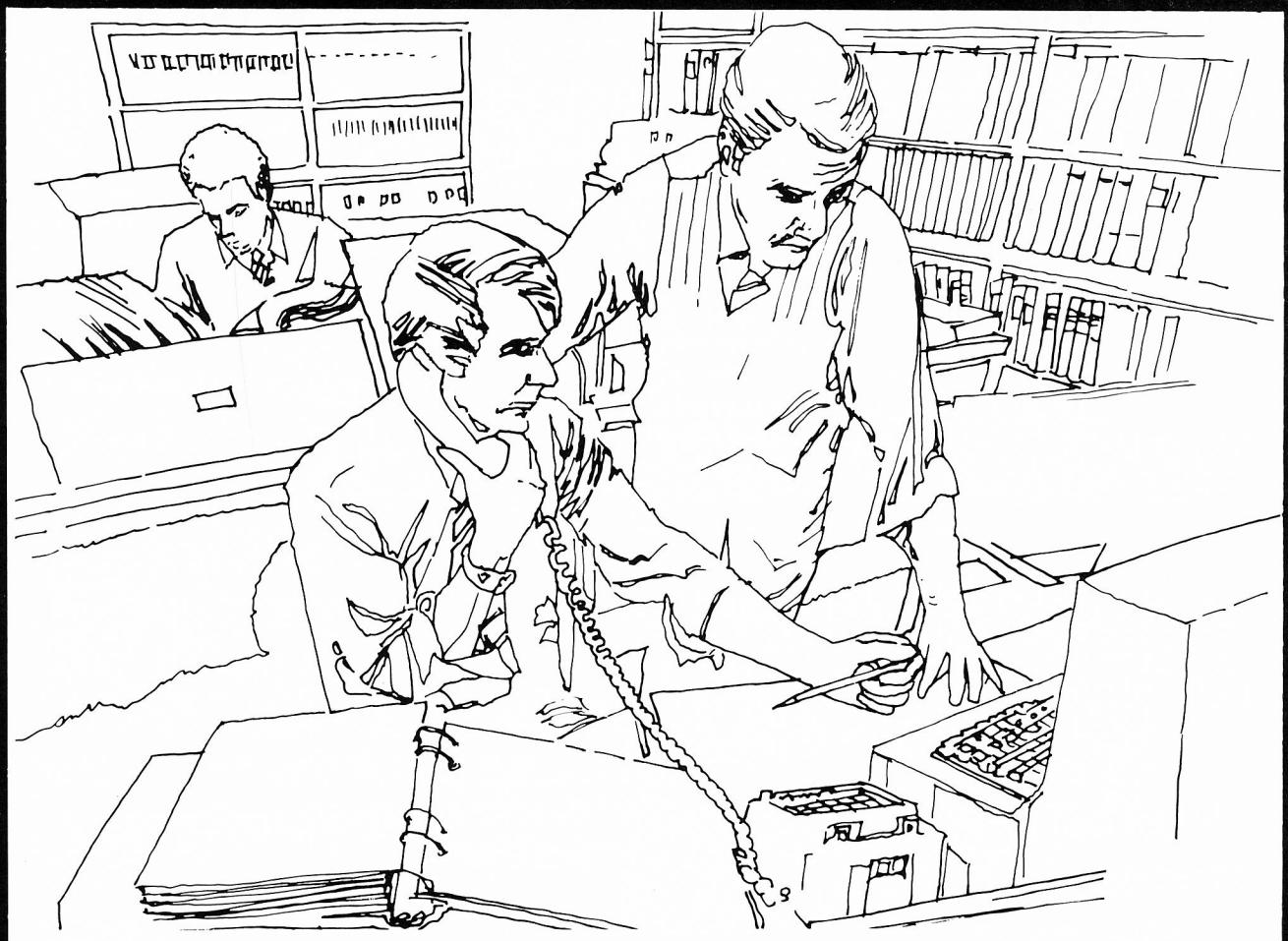
The following command file gives an example of writing a file from DCL. This command file is executed by LGICMD.COM (the command file in SYS$MANAGER: that is executed for every user who logs in) to track user logins by time and terminal.

```
$OPEN/
  WRITE OUT
  SYS$
  MANAGER:
  LOGINS.LIS
$WRITE OUT
  ""'F$LOGICAL
  ("SYS$
  LOGIN")'
  SUCCESS-
  FULLY
  LOGGED
  IN INTO '-
$'F$LOGICAL
  ("tt")' AT
  "F$TIME()"'
$CLOSE OUT
$EXIT:
$EXIT
```

The OPEN/WRITE statement will create a new file if the specified file does not exist or append to a file if it does. The '-' at the end of line 2 is used as a line continuation marker (which is helpful when your in verbose mode). The lexical functions used in the previous example, "F$logical("TT")" and "F$time()", are used to return the current system time and terminal device name. Lexical functions available are:

FUNCTION	ACTION
F$CVSI(Bit-position,width,integ)	Signed value extracted from the specified integer, converted to an ASCII literal.
F$CVUI(Bit-position,width,integ)	Unsigned value extracted from the specified integer, converted to an ASCII literal.
F$DIRECTORY()	Current default directory.
F$EXTRACT(offset,length,string)	substing starting at 'offset' of length 'length'.
F$LENGTH(string)	Length of specified string.
F$LOCATE(substring,string)	Returns the offset of the substring within string indicated; or, the length of the string in the substring is not found.
F$LOGICAL(logical-name)	Returns the logical translation of a logical name.
F$MESSAGE(code)	Message text associated with the specified status code. Useful for displaying error messages.
F$MODE()	One of the character strings INTERACTIVE or BATCH.
F$PROCESS	Current process name string.
F$TIME()	Current date and time of day in the format: dd-mmm-yyy hh:mm:ss.cc.
F$USER()	Current uic ([g,m]
F$VERIFY()	A numeric value of 1 if verification is set on; a numeric value of 0 if verification is set off.

RSTS SOFTWARE



RESOURCE MANAGEMENT and CHARGEBACK SYSTEM

DP Managers use ARSAP for:

- User and Project Accounting
- Monitoring Usage and Trends
- Controlling Performance
- Billing for Services

Also Available for VAX and RSX Systems

GEJAC INCORPORATED

P.O. Box 188
Riverdale, MD 20737 U.S.A. (301) 864-3700

VAX, RSX, and RSTS are trademarks of the Digital Equipment Corp.

CIRCLE 170 ON READER CARD

ACCTNG.B2S

SYSTEM-WIDE RESOURCE ACCOUNTING SYSTEM FOR RSTS/E

By Philip Hunt, OL.LF.B.P., 6400 E. Broad St., Columbus, OH 43213

DESCRIPTION

ACCTNG is a system program that will remove the RSTS imposed restrictions of the amount of CPU/KCT and other resource time that may be used under any one account without 'overflowing' the internal RSTS accounting structures. ACCTNG will keep track of all system usage in an indexed data file that is updated nightly with all usage that day. Since this file is kept in BP2 'double-precision' numbers, data will not be lost.

USAGE

ACCTNG is run from any terminal; it will find its own data file, and initialize one if one doesn't currently exist. It creates its data file in the account in which ACCTNG resides.

Example:

```
Ready

RUN $ACCTNG
ACCTNG — V1.02          RSTS V7.2-04 N   OLFBP   11/70

A)dd, L)ist, C)lear or E)xit:
```

At this point there are four options:

1) Add current statistics to file. This is usually done every night but may be done more often without damage.

2) List all or any account in a money-type format (see enclosed samples) that shows all usage FOR THE YEAR for CPU,KCT,Device Time, Connect-time. It also shows other items of interest. An account that is deleted, will be marked with a '(D)' next to the PPN and no updates are then done until recreated. This brings along an explanation of the last two columns, UPDATE shows how many times the account was found during nightly update and LUD shows the last time the account was found. You may direct the report output to any file or device. If selective mode is requested, a project,programmer number will be requested by ACCTNG.

3) Clear (Priviledged) — This CLEARS the YEARLY FILE after forcing a List operation; it will save the current file under ACCTNG.OLD and create a new ACCTNG.SYS file.

4) Exit the program.

```
LIST Example:
-------------

A)dd, L)ist, C)lear or E)xit: L

Output to <KB:> ?
Selective <N>   ? YES
Project, Programmer ? 1,2

   PPN              CPU-TIME          KCTS   DEVICE  CONNECT UPDATE   LUD
[001,002]             8:02.5         56048        0     1:44      2 10-Jan-83

Project, Programmer ? 1,10

   PPN              CPU-TIME          KCTS   DEVICE  CONNECT UPDATE   LUD
[001,010]             5:33.6         59666       46     1:42      2 10-Jan-83

Project, Programmer ? ^Z


Ready

ADD (Nightly update) Example:
-----------------------------
RUN $ACCTNG
ACCTNG - V1.02              RSTS V7.2-04 N  OLFBP  11/70

A)dd, L)ist, C)lear or E)xit: A
Accounts Updated

Updated   Accounts:  306
New       Accounts:  0


I have also included a sample report showing a complete system dump of all
accounts.

DO NOT FORGET, THIS PROGRAM MUST BE COMPILED IN DOUBLE PRECISION BP2.

Compilation Example:

BP2
OLD ACCTNG.B2S
BUILD/IND
COM/DOUBLE/OBJ
TKB @ACCTNG

PIP [PPN]ACCTNG.TSK<232>=ACCTNG.TSK

        WHERE: [PPN] IS WHERE YOU WOULD LIKE ACCTNG TO RESIDE

        !=======================================================              &
        !                                                                     &
        !               PROGRAM:        ACCTNG                                &
        !               AUTHOR :        PJH  OLFBP                            &
        !               VERSION:        V07                                   &
        !               EDIT   :        07                                    &
        !               DATE   :        072981                                &
        !                                                                     &
        !       THIS PROGRAM WILL DO THREE THINGS:                            &
        !       1) A)DD VALUES FROM SYSTEM TABLES TO ACCTNG                   &
        !          FILE IN DOUBLE PRECISION                                   &
        !       2) L)IST ONE OR ALL VALUES FROM MY DATA FILE                  &
        !          TO OUTPUT SPECIFIED                                        &
        !       3) C)LEAR THE DATA FILE (DOES A LIST OF ALL                   &
        !          FIRST)                                                     &
        !=======================================================              &
        !*****NOTE: THIS PROGRAM MUST BE COMPILED WITH*********               &
        !       D O U B L E   P R E C I S I O N .....                         &
        !=======================================================              &

1000    DIM F1P%(30%)                                    \                    &
        ON ERROR GOTO 19000                                                   &

1005    MAP (PPNREC)                                                          &
                PPN$    = 6%,                                                 &
                CPU,                                                          &
                KCT,                                                          &
                CNNCT,                                                        &
                DEVICE,                                                       &
                LAST.UPD,                                                     &
                UPDT,                                                         &
                LUD$ = 9%                                \                    &
                                                                              &
```

```
        MAP (PPNREC)                                                    &
                DTREC$ = 6%,                                            &
                INIT.DATE$=9%,                                          &
                INIT.TIME$=8%,                                          &
                UPD.DATE$=9%,                                           &
                UPD.TIME$=8%,                                           &
                SYS.ACCT,                                               &
                ACCT.CNT                                                &

1010    VERSION$        = "1"                                   \       &
        EDT$            = "02"                                  \       &
        INSTALLATION$   = ERT$(0%)                              \       &
        R.PACKAGE$      = SYS(CHR$(12%))                        \ !     &
        CHANGE            R.PACKAGE$ TO FIP%                    \ !     &
        PACKAGE.PROJ%   = FIP%(6%)                              \ !DO THIS SO WE...  &
        PACKAGE.PROG%   = FIP%(5%)                              \ !CAN FIND WHERE WE &
        PACKAGE.FSPEC$  = "["+NUM1$(PACKAGE.PROJ%)+               !WE CAME FROM     &
                          ","+NUM1$(PACKAGE.PROG%)+                                 &
                          "]"                                   \ !     &
        PACKAGE.FSPEC$  = "_"+CHR$(FIP%(23%))+                          &
                              CHR$(FIP%(24%))+                          &
                              NUM1$(FIP%(25%))+":"+                     &
                              PACKAGE.FSPEC$                            &
                                  IF FIP%(26%) AND 1%   \ !             &
                                                                        &
        PRINT "ACCTNG - V";VERSION$;".";EDT$,INSTALLATION$ \            &
        PRINT                                                   \       &
        OPEN "KB:" AS FILE 2%                                   \       &
        OPEN PACKAGE.FSPEC$+"ACCTNG.SYS" FOR INPUT AS FILE #1%,         &
                ORGANIZATION INDEXED FIXED,                             &
                ACCESS MODIFY, ALLOW NONE,                              &
                MAP PPNREC,                                             &
                PRIMARY KEY PPN$ NODUPLICATES           \               &
        CLOSE #1                                                        &

1011    OPEN PACKAGE.FSPEC$+"ACCTNG.SYS" AS FILE #1%,                   &
                ORGANIZATION INDEXED FIXED,                             &
                ACCESS MODIFY, ALLOW NONE,                              &
                MAP PPNREC,                                             &
                PRIMARY KEY PPN$ NODUPLICATES           \               &
                                                                        &
        GET #1%, KEY #0% EQ "000000"                    \               &
        F.DATE$ = INIT.DATE$+""                         \               &
        F.TIME$ = INIT.TIME$+""                         \               &
        U.DATE$ = UPD.DATE$+""                          \               &
        U.TIME$ = UPD.TIME$+""                          \               &
        CUR.ACCT= ACCT.CNT                              \               &
        SYSTEM.ACCTS = SYS.ACCT                         \               &
        GOTO 1020                                                       &

1012    !NO DATE REC, BAD FORMAT, SO REINIT IT                          &
        GOSUB 4010                                      \               &
        GOTO 1011                                                       &

1013    !NO FILE FOUND, INIT ONLY                                       &
        GOSUB 4010                                      \               &
        PRINT                                                           &

1020    PRINT "A)dd, L)ist, C)lear or E)xit: ";         \               &
        INPUT #2%, CMD$                                 \               &
        CMD$ = LEFT(CVT$$(CMD$,-1%),1%)                 \               &
        CMD% = INSTR(1%,"ALCZE",CMD$)                   \               &
        GOTO 1020 IF CMD% = 0%                          \               &
        ON CMD% GOSUB 2000,3000,4000,4010,32700,5000,6000 \             &
        GOTO 32700

2000    !ADD TO CURRENT FILE                                            &
        ! (FIRST, CLEAR ALL UPDATE FLAGS)                               &
        RESET.FLAG%=-1%                                 \               &
        FILE.ACCT = 0%                                  \               &
        FIND #1%, KEY#0% GT "000000"                    \               &
        WHILE -1%                                       \               &
                GET #1%                                 \               &
                FILE.ACCT=FILE.ACCT+1                   \               &
                LAST.UPD=0%                                             &
2002            UPDATE #1%                              \               &
        NEXT                                                            &

2005    INDX% = 0%                                      \               &
        SYSTEM.ACCT=0%                                  \               &
        WHILE -1%                                       \               &
                INDX%=INDX%+1%                          \               &
                INFO$=SYS(CHR$(6%)+CHR$(14)+                            &
                        CHR$(INDX%)+CHR$(SWAP%(INDX%))+                 &
                        CHR$(RESET.FLAG%)+CHR$(RESET.FLAG%)+            &
                        CHR$(0%)+CHR$(0%)+CHR$(1%)+             &       &
                                STRING$(0%,21%))\                       &
                SYSTEM.ACCT=SYSTEM.ACCT+1               \               &
                CHANGE INFO$ TO FIP%                    \               &
                PROJ%=FIP%(8%)                          \               &
                PROG%=FIP%(7%)                          \               &
                GOSUB 10000                                             &

2010            FIND #1%, KEY #0% EQ PPN$               \               &
                UPDTD%=UPDTD%+1%                                        &

2015            GOTO 2050                                               &

2020            KCT=0                                   \               &
                CPU=0                                   \               &
                CNNCT=0                                 \               &
                DEVICE=0                                \               &
                LAST.UPD=0                              \               &
                UPDT=0                                  \               &
                NEWADD%=NEWADD%+1%                      \               &
                PUT #1%                                                 &

2050            GET #1%, KEY#0% EQ PPN$                                 &

2060            CPU.TEMP=256.*FIP%(14%)+FIP%(13%)       \               &
                CPU.HIGH=(FIP%(22%) AND 252%)/4%        \               &
!HIGH IS HIGH 6 BITS OF FIP(21,22)                                      &
                CPU.TEMP=CPU.TEMP+(CPU.HIGH*65536.)     \               &
                                                                        &
                DEVICE.TEMP=256.*FIP%(20%)+FIP%(19%)    \               &
                                                                        &
                CNNCT.TEMP =256.*FIP%(16%)+FIP%(15%)    \               &
                                                                        &
                KCT.TEMP   =256.*FIP%(18%)+FIP%(17%)    \               &
!HIGH IS LOW ORDER 10 BITS OF FIP(21,22)                                 &
                KCT.HIGH   =FIP%(21%)+(256.*(FIP%(22%) AND 3%)) \       &
                KCT.TEMP   =KCT.TEMP+(65536.*KCT.HIGH)  \               &
                                                                        &
                KCT=KCT+KCT.TEMP                        \               &
                CPU=CPU+CPU.TEMP                        \               &
                DEVICE=DEVICE+DEVICE.TEMP               \               &
                CNNCT =CNNCT+CNNCT.TEMP                 \               &
                UPDT=UPDT+1                             \               &
                LAST.UPD=-1%                            \               &
                LUD$ = DATE$(0%)                        \               &
                UPDATE #1%                              \               &
        NEXT                                                            &

2500    PRINT "Accounts Updated"                        \               &
                                                                        &
        GET #1%, KEY#0% EQ "000000"                     \               &
        UPD.DATE$=DATE$(0%)                             \               &
        UPD.TIME$=TIME$(0%)                             \               &
        SYS.ACCT = SYSTEM.ACCT                          \               &
        ACCT.CNT = FILE.ACCT+NEWADD%                    \               &
        CUR.ACCT = ACCT.CNT                             \               &
        UPDATE #1%                                      \               &
                                                                        &
        PRINT                                           \               &
        PRINT "Updated   Accounts: ";UPDTD%             \               &
        PRINT "New       Accounts: ";NEWADD%            \               &
        RETURN                                                          &

3000    !LIST FROM CURRENT FILE                                         &
        ZERO.MSG$=""                                    \               &
        PROG%=0%                                        \               &
        PROJ%=0%                                        \               &
        PPN$=STRING$(0%,6%)                             \               &
        IF CLEAR.FLAG%  THEN                                            &
                ZERO.MSG$=" and INITIALIZATION"         \               &
                SELECTIVE$ = "N"                                        &

3005    LINPUT "Output to <KB:> ";OUTP$                 \               &
        OUTP$="_KB:" IF OUTP$=""                        \               &

3010    GOTO 3017 IF CLEAR.FLAG%                        \               &
        INPUT "Selective <N>  ";ANS$                    \               &
        SELECTIVE$="N"                                  \               &
        SELECTIVE$="Y" IF LEFT(ANS$,1%)="Y"                             &
```

```
3015    IF SELECTIVE$="Y" THEN                                          &
                INPUT "Project, Programmer ";PROJ%,PROG%\               &
                GOSUB 10000                             \               &
                GOTO 3015 IF PPN$="000000"              \               &
                TOTLINE%=0%                             \               &
                FIND #1%, KEY #0% EQ PPN$                               &

3017    OPEN OUTP$ FOR OUTPUT AS FILE 3%                                &

3020    GET #1                                          \               &
        GOTO 3020       IF PPN$="000000"                \               &
        GOSUB 11000                                     \               &
        IF SELECTIVE$="Y" THEN 3015 ELSE 3020                           &

3030    !EOF ON LIST FROM CLEAR/INIT FUNCTION                           &
        RETURN                                                          &

4000    !CLEAR CURRENT FILE                                             &
        INPUT "Are you sure <NO> ";AANS$                \               &
\       GOTO 4005 IF AANS$="YES"                        \               &
\       PRINT "CLEAR aborted!!"                                         &

4005    CLEAR.FLAG% = -1%                               \               &
        GOSUB 3000                                                      &

4010    PRINT "Initializing data file"                  \               &
        CLOSE #1%                                       \               &
        kill package.fspec$+"ACCTNG.OLD"                \               &
        goto 4012                                                       &

4011    !no file to kill                                                &
        print "No old file to be killed, continuing"    \               &
\       goto 4012                                                       &


4012    NAME PACKAGE.FSPEC$+"ACCTNG.SYS" AS                             &
               PACKAGE.FSPEC$+"ACCTNG.OLD"              \               &
        PRINT "Old file: ";PACKAGE.FSPEC$+"ACCTNG.SYS"                  &
               +" named to ";PACKAGE.FSPEC$+"ACCTNG.OLD" \              &
        goto 4015                                                       &

4013    !no file to rename                                              &
        PRINT "No current file, continuing"             \               &
        goto 4015                                                       &


4015    OPEN PACKAGE.FSPEC$+"ACCTNG.SYS" FOR OUTPUT AS FILE #1%,        &
               ORGANIZATION INDEXED FIXED,                              &
               FILESIZE 250%,                                           &
               ACCESS MODIFY, ALLOW NONE,                               &
               MAP PPNREC,                                              &
               PRIMARY KEY PPN$ NODUPLICATES            \               &
        PPN$="000000"                                   \               &
        INIT.DATE$= DATE$(0%)                           \               &
        INIT.TIME$= TIME$(0%)                           \               &
        UPD.DATE$ = "NONE"                              \               &
        UPD.TIME$ = "NONE"                              \               &
        SYS.ACCT  = 0%                                  \               &
        ACCT.CNT  = 0%                                  \               &
        PUT #1%                                         \               &
        PRINT   "File Cleared!!!"                       \               &
        RETURN                                                          &


5000    !RESERVED                                                       &
        RETURN                                                          &


6000    !RESERVED                                                       &
        RETURN                                                          &


10000   !GIVEN PROJ%,PROG%, WILL GENERATE PPN$ 6 CHARS LONG             &
        PP1$ = NUM1$(PROJ%)                             \               &
        PP2$ = NUM1$(PROG%)                             \               &
        PP1$ = RIGHT("000",LEN(PP1$)+1%)+PP1$           \               &
        PP2$ = RIGHT("000",LEN(PP2$)+1%)+PP2$           \               &
        PPN$ = PP1$ + PP2$                              \               &
        RETURN                                                          &


11000   !DISPLAY A LINE FROM FILE                                       &
        GOSUB 12000                                     \               &
        PRINT #3%, "["+LEFT(PPN$,3%)+","+RIGHT(PPN$,4%)+"]"; \          &
        PRINT #3%," (D)"; IF LAST.UPD>=0%               \               &
        TT=CPU/10                                       \               &
        TTTS=INT((TT-INT(TT))*10%)                      \               &
        GOSUB 16000                                     \               &
        SS$=SS$+"."+CHR$(48%+TTTS)                      \               &
        SS$=FNSP$(SS$,16%)                              \               &
        PRINT #3%, TAB(13%);SS$;                        \               &
        SS$ = NUM1$(KCT)                                \               &
        SS$ = FNSP$(SS$,12%)                            \               &
        PRINT #3%, TAB(33%);SS$;                        \               &
        TT=DEVICE                                       \               &
        GOSUB 15000                                     \               &
        SS$=FNSP$(SS$,9%)                               \               &
        PRINT #3%, TAB(41%);SS$;                        \               &
        TT=CNNCT                                        \               &
        GOSUB 15000                                     \               &
        SS$=FNSP$(SS$,9%)                               \               &
        PRINT #3%, TAB(53%);SS$;                        \               &
        PRINT #3%, TAB(65%);FNSP$(NUM1$(UPDT),4%);      \               &
        PRINT #3%, " ";LUD$;                            \               &
        PRINT #3%                                       \               &
        PRINT #3% IF SELECTIVE$="Y"                     \               &
        LCOUNT%=LCOUNT%+1%                              \               &
        TOTLINE%=0% IF LCOUNT%>57%                      \               &
        RETURN                                                          &


12000   !DISPLAY TITLE                                                  &
        RETURN IF TOTLINE%                              \               &
        GOTO 12010 IF SELECTIVE$="Y"                    \               &
        PGE%=PGE%+1%                                    \               &
        PRINT #3%, CHR$(12%);                           \               &
        PRINT #3%, "Acctng Dump";ZERO.MSG$;" of Snapshot File on ";     &
               DATE$(0%);" at ";TIME$(0%);TAB(70%);"Page ";PGE% \       &
        PRINT #3%                                       \               &
        PRINT #3%, "File Created: ";F.DATE$;" at ";F.TIME$,\            &
        PRINT #3%, "File   Accounts: ";CUR.ACCT         \               &
        PRINT #3%, "File Updated: ";U.DATE$;" at ";U.TIME$,\            &
        PRINT #3%, "System Accounts: ";SYSTEM.ACCTS                     &

12010   PRINT #3%                                       \               &
        PRINT #3%, "   PPN               CPU-TIME            KCTS ";\   &
        PRINT #3%, "  DEVICE  CONNECT UPDATE   LUD"\                    &
        TOTLINE%=-1%                                    \               &
        LCOUNT% = 7%                                    \               &
        RETURN                                                          &

15000   ! TIME - HHHH:MM FROM TT (BEING MINUTES,NOT SECONDS             &
        TT=TT*60.\                              !GET INTO SECONDS       &
        HOURS.FLAG%=-1%\                                                &
        GOTO 16010                                                      &

16000   !       T I M E -    H H H H : M M : S S . S                    &
        ! RETURNS A STRING SS$ FROM TT                                  &

16005   HOURS.FLAG%=0%                                                  &

16010     SS$=""                                                        &
        \ TT%=0%                                                        &
        \ TT1%=TT/3600.                                                 &
        \ IF TT1%<>0% THEN                                              &
                TT=TT-3600.*TT1%                                        &
        \       GOSUB 16040                                             &
        \       SS$=SS$+":"                                             &
        ! CLEAR STRING TO BE RETURNED AND INITIALIZE                    &
        ! T1%= # OF HOURS - SUBTRACT FROM TOTAL IF NON-ZERO             &

16020   TT1%=TT/60%                                                     &
        \ IF (TT1% OR TT%)<>0% THEN                                     &
                TT=TT-60%*TT1%                                          &
        \       GOSUB 16040                                             &
        \       RETURN IF HOURS.FLAG%    !ONLY WANT HH:MM, SO EXIT      &
        \       SS$=SS$+":"                                             &
        ! NOW T1%= # OF MINUTES - SUBTRACT FROM TOTAL IF NON-ZERO AND   &
        ! TACK ON TO STRING                                             &

16030   TT1%=TT                                                         &
        ! PREPARE TO GET SECONDS.TENTHS OF SEC.                         &

16040   TT$=NUM$(TT1%+TT%*100%)                                         &
        \ SS$=SS$+MID(TT$,2%+TT%,LEN(TT$)-2%-TT%)                       &
        \ TT%=1%                                                        &
        ! TACK THIS PART ONTO THE STRING                                &

16050   RETURN                                                          &
                                                                        &

19000   !ERROR PROCESSING                                               &
        IF ERR=5% AND ERL=1010 THEN                                     &
                PRINT "?ACCTNG- Data file not found-INITIALIZING" \     &
                RESUME 1013                                             &

19010   IF ERR=155% AND ERL=3015 THEN                                   &
                PRINT "%Account not found"              \               &
                RESUME 3015                                             &

19020   IF (ERR=11% OR ERR=155%) AND                                    &
           (ERL=3020) THEN                                              &
                PRINT #3%, CHR$(12%);                   \               &
                CLOSE #3%                               \               &
                RESUME 3030 IF CLEAR.FLAG%              \               &
                RESUME 32700                                            &

19025   IF (ERR=155%) AND (ERL=1011) THEN                               &
                PRINT "%No Date record found"           \               &
                RESUME 1012                                             &

19030   IF ERL=3017 THEN                                                &
                PRINT "?Bad output specification"       \               &
                RESUME 3000                                             &

19040   IF ERR=5% AND ERL=2005 THEN                                     &
                RESUME 2500                                             &

19050   IF ERR=155% AND ERL=2010 THEN                                   &
                RESUME 2020                                             &

19060   IF (ERR=155% OR ERR=11%) AND ERL=2000 THEN                      &
                RESUME 2005                                             &

19070   if erl=4010 then resume 4011    !no old file to kill-ok         &

19080   if erl=4012 then resume 4013    !no current file-ok             &

19980   IF ERR=11% THEN                                                 &
                RESUME 32700                                            &

19990   PRINT "?ACCTNG - ";ERT$(ERR);" (";NUM1$(ERL);")-FATAL"\         &
                RESUME 32700                                            &

32700   PRINT                                                           &
        CLOSE #1%,#2%                                                   &

32710   DEF FNSP$(AR$,P%)=SPACE$(P%-LEN(AR$))+AR$                       &

32767   END                                                             &
```

```
Acctng Dump of Snapshot File on 07-Jan-83 at 05:11 PM          Page  1

File Created: 14-Jan-82 at 10:02 AM       File   Accounts:  373
File Updated: 06-Jan-83 at 08:54 PM       System Accounts:  305
```

PPN		CPU-TIME	KCTS	DEVICE	CONNECT	UPDATE	LUD
[000,001]		0.0	0	0	0	70	06-Jan-83
[001,000]		10:03.0	151976	47	1:39	70	06-Jan-83
[001,001]		48:42:52.7	3584947	28	2:57	70	06-Jan-83
[001,002]		73:09:04.5	38855575	2401:14	1621:49	70	06-Jan-83
[001,003]		1:08:02.1	742939	30	9:28	70	06-Jan-83
[001,004]		1:35:35.0	713129	2:03	10:14	70	06-Jan-83
[001,005]		21:59:59.1	15523246	183:18	253:54	70	06-Jan-83
[001,006]		0.0	0	0	0	70	06-Jan-83
[001,007]		6:22.2	24932	5	2:37	70	06-Jan-83
[001,008]		14.6	1286	0	19	70	06-Jan-83
[001,009]		13:27.9	64593	1:45	2:27	70	06-Jan-83
[001,010]		65:13:36.4	41108124	254:09	1498:10	70	06-Jan-83
[001,011]		1:06.5	8862	0	24	70	06-Jan-83
[001,012]		0.9	45	0	0	70	06-Jan-83
[001,013]		0.0	0	0	0	70	06-Jan-83
[001,014]		0.4	9	0	0	70	06-Jan-83
[001,015]		45.5	6956	8	4	70	06-Jan-83
[001,016]		0.0	0	0	0	70	06-Jan-83
[001,017]		0.0	0	0	0	70	06-Jan-83
[001,018]		21:17.9	287039	49	3:10	70	06-Jan-83
[001,019]		0.0	0	0	0	70	06-Jan-83
[001,020]		7.5	1555	0	5	70	06-Jan-83
[001,021]		75:37:35.1	42391838	256:27	827:23	70	06-Jan-83
[001,022]		1.6	149	0	0	70	06-Jan-83
[001,023]		0.0	0	0	0	70	06-Jan-83
[001,024]		4:56.7	67142	57	1:06	70	06-Jan-83
[001,025]		33:20:00.7	18459087	176:39	798:41	69	06-Jan-83
[001,026]		12:09:08.8	5554523	54:30	454:50	49	06-Jan-83
[001,030]		0.0	0	0	0	70	06-Jan-83
[001,031]		6:13.0	77003	29	1:31	47	06-Jan-83
[001,032]		41.2	2863	0	28	30	06-Jan-83
[001,033]	(D)	0.0	0	0	0	17	05-Sep-82
[001,035]		57:03.6	791168	2:03	52:33	31	06-Jan-83
[001,038]		20:08:44.6	13882483	93:07	118:38	70	06-Jan-83
[001,040]		4:41:15.1	3424433	21:12	119:29	31	06-Jan-83
[001,041]	(D)	0.0	0	0	0	17	02-Jun-82
[001,045]		1:16:18.6	816248	7:55	60:24	31	06-Jan-83
[001,050]		66:01:24.7	41539474	201:12	1610:15	70	06-Jan-83
[001,051]		13:04:47.1	6859340	9:44	233:22	67	06-Jan-83
[001,052]		1:12:50.1	946693	1:41	37:06	70	06-Jan-83
[001,055]		0.0	0	0	0	70	06-Jan-83
[001,056]		0.0	0	0	0	70	06-Jan-83
[001,060]		7:16.2	36739	5	6:17	70	06-Jan-83
[001,061]	(D)	0.0	0	0	0	17	07-Jul-82
[001,062]		5:18.3	57181	6	1:01	48	06-Jan-83
[001,063]		37:29.3	548774	1:48	5:49	31	06-Jan-83
[001,064]		26:23.0	296994	1:29	7:32	31	06-Jan-83
[001,065]		2:43:41.5	1261838	2:59	13:42	11	06-Jan-83
[001,070]		10:47:47.5	9083616	69:04	136:44	30	06-Jan-83
[001,076]		30.8	2074	0	18	70	06-Jan-83
[001,077]		14:22.5	230149	1	2:56	52	06-Jan-83

♥

RSTS/E MULTITERMINAL INPUT/OUTPUT SERVICE

By Michael H. Koplitz, Computer Integrated Management Systems

The multiterminal input/output service allows for the development of software which controls many terminals from one input/output channel. This feature is useful for the development of user interfaces, data entry programs, and other RSTS/E system programs. Several terminals can be doing the same task but only ONE job is in use. The efficiency of multiterminal service can be seen in computer shops where there is a high volume of data input.

SYSGEN OPTION

To be able to use multiterminal input/output service, the system manager must specify during the SYSGEN dialogue that multiterminal service is desired. The SYSGEN question and long description are:

> An optional feature of the RSTS/E terminal service allows one job to interact with several terminals through special forms of the Record I/O GET and PUT statements. This feature is useful in applications where the same basic function is performed on several terminals and a separate job for each is undesirable or at least inefficient. Would you like to include this feature (YES OR NO)?
>
> Multi-terminal service? #Y#

Answer Y or linefeed to this question.

THE MASTER TERMINAL

To use multiterminal input/output service a master terminal must be selected. This master terminal will be opened on one of the twelve RSTS/E channels. The OSC should not be used as the master terminal. This can interfere with OPSER, if OPSER if running.

```
OPEN "KB:" AS FILE #1%
```

This OPEN statement associates the current keyboard with channel one. Any channel, one through twelve can be used. Also a keyboard other than the current keyboard, can be used for the master terminal.

```
OPEN "KB10:" AS FILE #2%
```

This OPEN statement associates keyboard ten with channel two. Keyboard ten becomes the master keyboard.

Keyboards which are to be included in multiterminal input/output services must be assigned to the job.

OUTPUT FOR MULTITERMINAL SERVICES

The PUT or PRINT statements are used to transmit data to the terminals under the multiterminal service. The PUT statement is used in combination with the FIELD statement to transmit data to the multiterminal service.

```
PUT #5%, RECORD 32767% + 1% + T%, COUNT N%
```

The RECORD option must be 32767% + 1% to indicate to RSTS/E that the output is to be for multiterminal services. The variable T% in the RECORD count is a specific keyboard unit number to transmit to. Therefore, all of the terminals assigned to the job can be transmitted to by using RECORD clause 32767% + 1%, or transmission can be to a single terminal by using 32767% + 1% + T%. The COUNT option is used to transmit a specific amount of the buffer to the terminal.

The error, ?NOT A VALID DEVICE (ERR = 6), can occur if the terminal being addressed is not the master keyboard nor one of the slave terminals. A slave terminal is one which is assigned to the job, and is not indicated in the OPEN statement of the master terminal. Another possible error is, ?I/O CHANNEL NOT OPEN (ERR = 9), which indicates that the terminal in question is not assigned to the job.

The PRINT or PRINT-USING statement can be used with multiterminal input/output services. The RECORD clause is used to designate multiterminal input/output services.

```
PRINT #5%, RECORD 32767% + 1% + T%,A$
PRINT #5%, RECORD 32767% + T% + 1% + T%, USING "!!!", A$;
```

The FIELD statement is NOT used if the PRINT or PRINT-USING statements are used.

To transmit binary data add 4096% to the RECORD clause.

INPUT FROM MULTITERMINAL SERVICES

Input from multiterminal services can be requested from a specific keyboard or from any of the terminals that might have some data available. The type of input is specified in the RECORD clause of the GET statement.

To get input from a specific keyboard the following statement is used.

```
GET #5%, RECORD 32767% + 1% + T%
```

The variable T% is the unit number of the terminal from which data is to be accepted. A FIELD statement is then used so that the program can read the data. The variable RECOUNT contains the length of the buffer.

```
FIELD #5%, RECOUNT AS A$
```

The information from keyboard T% is contained in variable A$. If there is not any data available, the error, ?DATA ERROR ON DEVICE (ERR = 13), is generated. The system WILL NOT stall (KB state) on the GET statement.

To get data from any terminal on the multiterminal service the following GET statement is used.

```
GET #5%, RECORD 32767% + 1% + 16384% + S%
```

Where S% has several meanings:

Value: 0%
Meaning:
The GET statement waits until input is available from any of the terminals on the service. The program stalls until data is received. After data is received the error, ?DATA ERROR ON DEVICE (ERR = 13), may be generated under some circumstances (race condition with ↑C). A race condition can occur when two jobs are accessing the same data.

Value: 1 to 255
Meaning:
The GET statement waits up to S% seconds for input from any terminal. If no data is received in S% second the error, ?DATA ERROR ON DEVICE, is generated.

Value: 8192%
Meaning:
The GET statement immediately generates the error, ?DATA ERROR ON DEVICE, if no data is available.

A ↑Z entered at either the master or slave terminal generates the error, ?END OF FILE ON DEVICE.

When the GET statement is executed and data is available, the first character of the buffer is the unit number of the terminal where the data was generated. To convert this value into a unit number use the ASCII function.

LIMIT.BAS — AN EXAMPLE OF MULTITERMINAL SERVICES

LIMIT.BAS allows the system manager to prevent specific terminals from logging into the system. The program uses multiterminal input/output services to accomplish this task. All terminals on the system can be "limited." LIMIT.BAS has special features built in to allow a user to free up the terminal if the LIMIT password is known to him.

LIMIT.BAS has two time parameters. The first is the start time, when should LIMIT.BAS start limiting the terminals. Second is the stop time; when should LIMIT.BAS stop limiting the terminals. LIMIT.BAS will sleep until the start time is reached. It will send a message to KB0: indicating its start and stop times.

LIMIT.BAS allows for a password to be established which when entered on a terminal will cause LIMIT.BAS to free up that terminal for operator use. The password MUST be six characters in length. The password feature need not be used. A<CR> entered to the password question will set up this situation.

A message for the session is prompted for. LIMIT.BAS will display the inputted message each time a user enters something which is not the password on a "limited" terminal. A switch exists to this message. This switch "/S" allows for the entering of individual terminals that are to be limited. If the "/S" is not used ALL terminals that are not in use at the moment will be limited.

It is HIGHLY recommended that if the time set for LIMIT.BAS to start is in the future, that the "/S" switch NOT BE USED. If LIMIT.BAS is to start in the future it will detach. If LIMIT.BAS is to start immediately the prompt LIM> will appear.

Note that there is a ten second time limit on entering a command to LIMIT.BAS. The time limit was set up to insure that any terminals that are "limited" are serviced in some reasonable amount of time. If the program completely stalled at the LIM> prompt, then terminals might not be serviced for very long periods of time.

LIMIT.BAS will first attempt to open the master terminal. LIMIT.BAS uses KB40: as the master. This can be adjusted by changing line 07. J% contains the master terminal number. If the master terminal is not available an error is generated and LIMIT.BAS will ask for a different terminal to be the master terminal.

There might be some terminals which are not assigned to LIMIT.BAS because they are in use. LIMIT.BAS does contain some commands which will force the terminal to become assigned to it. There are commands which cause LIMIT.BAS to free up a terminal. Also assigning terminals which are not in use can be done. The commands are entered at the LIM> prompt. The commands are:

Command	Meaning
ASSIGN	Assigns a terminal to LIMIT.
CHANGE	Changes a LIMIT feature.
DEASSIGN	Deassigns a terminal from LIMIT.
DETACH	Detaches the job. Attaches to the job if other commands are desired.
END	Ends LIMIT.
FREE	Lists the free terminals on the system, those not controlled by LIMIT.
HELP	Displays HELP message.
LIST	Lists all assigned terminals.
SEIZE	Siezes a terminal which is in use.
VIEW	Lists all of LIMIT's parameters.

It is best to detach LIMIT.BAS; it runs more efficiently!

REFERENCES

RSTS/E Programming Manual

RSTS/E System Generation Manual (for SYSGEN dialogue)


```
01 !*****************************************************************
\  !*   THIS PROGRAM ALLOWS SYSTEM MANAGER TO PREVENT CERTAIN TERMINALS
\  !*   FROM LOGGING INTO THE SYSTEM.
\  !*
\  !*   THIS PROGRAM WAS WRITTEN BY M H KOPLITZ
\  !*           COMPUTER INTEGRATED MANAGEMENT SYSTEMS
\  !*
\  !*****************************************************************
\  !*
\  !*   REVISION #1 (TO VERSION 2)  M H KOPLITZ    OCT 2, 1981
\  !*
\  !*           PURPOSE: DO NOT LET ANY TIME BE ENTERED EXCEPT
\  !*                    XX:XX.
\  !*
\  !*****************************************************************

05      DIM A(60%),B(60%)

07      J% = 56%                                !NUMBER OF TERMINALS

08      M$ = "KB" + NUM1$(J%) + ":"             !MASTER KB NUMBER

10      PRINT "LIMIT   V 2.1              "
              "    LIMIT systems use"

15      ON ERROR GOTO 31000                     !ERROR ROUTINE

16      X$ = SYS(CHR$(6%) + CHR$(-7%))          !SET ^C TRAP

17      OPEN M$ AS FILE #12%                    !OPEN THE MASTER FILE FOR
                                                 MULTI-TERMINAL SERVICES
20      PRINT                                   !PRINT HEADER

30      INPUT "Time to start <NOW>";A$          !ASK FOR TIME START

40      IF A$ = "" THEN A$ = CVT$$(TIME$(0),-1%)
\       GOTO 50 IF LEN(A$) = 5%
\       PRINT "?Time must be in the form XX:XX"
\       GOTO 30                                 !IF NULL USE CURRENT TIME
                                                ! ELSE CHECK LEGALITY.

50      INPUT "Time to end";B$
\       IF B$ = "" THEN 32767                   !ASK FOR TIME STOP

60      GOTO 63 IF LEN(B$) = 5%
\       PRINT "?Time must be in the form XX:XX"
\       GOTO 50                                 !IF NULL STOP, ELSE CHECK
                                                ! TO SEE IF VALID.
63      INPUT "Password for session";P9$
\       PRINT "%No password for session"
              IF P9$ = ""
\       P9$ = "~" IF P9$ = ""
\       GOTO 70 IF P9$ = "~"
\       PRINT "%Password must be 6 characters";
              " in length"
              IF LEN(P9$) <> 6
\       GOTO 63
              IF P9$ = "" OR LEN(P9$) <> 6%     !ENTER PASSWORD
                                                 FOR SESSION

70      PRINT "Message for the session";
\       INPUT LINE C$
\       C$ = CVT$$(C$,4%)
\       C1$ = CVT$$(C$,32% + 4%)
\       C$ = LEFT(C$,INSTR(1%,C$,"/")-1%)
              IF INSTR(1%,C$,"/") <> 0%
\       C$ = C$ + CHR$(10) + CHR$(13)
\                                               !MESSAGE TO BE SENT

71      IF A$ > TIME$(0) THEN GOSUB 1500        !TIME TO START NOT YET DETACH

72      IF A$ > TIME$(0) THEN SLEEP 30%
\               L9% = 1%
\               GOTO 72                         !TIME TO START NOT HERE YET

75      X$ = SYS(CHR$(6%) + CHR$(9%) + CHR$(0)) !GET KB NUMBER

77      T% = (ASCII(MID(X$,2,1))/2)             !T% HAS KB NUMBER

78      X$ = SYS(CHR$(6%) + CHR$(-5%) + CHR$(0)
                + "%System will be limited"
                + " until: "
                + B$ + CHR$(13%)
                + CHR$(10))
                                                !SEND MESSAGE TO KB0:

79      GOTO 7000
                IF RIGHT(C1$,
                        INSTR(1%,C1$,"/") + 1%)
                         = "S"
                AND INSTR(1%,C1$,"/") <>0

80      FOR X = 9 TO J%-1%                      !ALL LEGAL TERMINALS

85              IF T% = X THEN B(X) = 0%
\               GOTO 120                        !SKIP CURRENT TERMINAL

90              X$ = SYS(CHR$(6%) + CHR$(10%)
                   + STRING$(20%,0%)
                   + "KB" + CHR$(X)
                   + CHR$(255%))
                                                !ASSIGN THE KB

100             X$ = SYS(CHR$(6%) + CHR$(-5%)
                        + CHR$(X)
                        + "%System will be "
                        + "limited until: "
                        + B$ + CHR$(13%)
                        + CHR$(10))
                                                !SEND SYSTEM LIMITED PHRASE

110             A(X) = X                        !FLAG TERMINAL AS ASSIGNED

120     NEXT X                                  !CONTINUE ASSIGNING TERMINALS

125     GOTO 217 IF L9% = 1%                    !IN DETACHED CONDITION

130     WAIT 10                                 !WAIT 10 SECONDS

140     PRINT "LIM>";                           !READY FOR LIMIT COMMAND INPUT

150     INPUT D$                                !GET THE COMMAND

155     D$ = CVT$$(D$,32% + 4% + 2% + 8%)       !CONVERT ENTRY TO USABLE DATA

160     IF D$ = "END" THEN 32766                !IF END GOTO END

170     IF D$ = "HELP" THEN GOSUB 1000          !GOSUB HELP PARAGRAPH

180     IF D$ = "DEASSIGN" THEN GOSUB 1100      !DEASSIGN SECTION

190     IF D$ = "ASSIGN" THEN GOSUB 1200        !ASSIGN SECTION

195     IF D$ = "SEIZE" THEN GOSUB 8000         !SEIZE A TERMINAL IN USE

197     IF D$ = "CHANGE" THEN GOSUB 9000        !CHANGE LIMIT PARAMETERS

200     IF D$ = "LIST" THEN GOSUB 1300          !LIST ALL ASSIGNED DEVICES

210     IF D$ = "FREE" THEN GOSUB 1400          !LIST ALL UNASSIGNED DEVICES

215     IF D$ = "DETACH" THEN GOSUB 1500        !DETACH OUT

217     GOSUB 2000                              !TIME CHECK

218     IF D$ = "VIEW" THEN GOSUB 9500          !VIEW CERTAIN DATA

219     GOSUB 5000                              !SEE IF ANYONE TALKED TO US

220     X$ = SYS(CHR$(6%) + CHR$(9%) + CHR$(0)) !GET KB NUMBER

223     IF ASCII(MID(X$,2%,1%)) > 80%
                THEN SLEEP 1%
\               GOTO 217                        !DON'T ASK QUESTION IF DET.

230     GOTO 140                                !BACK TO LIM QUESTION
```

```
1000 !*****************************************
\    !*
\    !* PRINT HELP MESSAGES
\    !*
\    !*****************************************
1005    PRINT                                   !HELP MESSAGE
1010    PRINT "ASSIGN";TAB(30);
              "Assign a KB to Limit"
1015    PRINT "CHANGE";TAB(30);
              "Change a Limit feature"
1020    PRINT "DEASSIGN";TAB(30);
              "Deassign a KB from Limit"
1030    PRINT "DETACH";TAB(30);
              "Detach the job"
1040    PRINT "END";TAB(30);
              "End Limit now"
1050    PRINT "FREE";TAB(30);
              "List free terminals"
1060    PRINT "HELP";TAB(30);
              "This help message"
1070    PRINT "LIST";TAB(30);
              "List all assigned KB"
1075    PRINT "SEIZE";TAB(30);
              "Seize a terminal which is inuse"
1077    PRINT "VIEW";TAB(30);
              "View limit parameters"
1080    RETURN                                  !FINSHED

1100 !*****************************************
\    !*
\    !* DEASSIGN A KB:
\    !*
\    !*****************************************
1105    INPUT "Deassign KB:";N%                 !DEASSIGN A KB:
1107    PRINT "% Console can not be affected ";
              "by limit" IF N% = 0%
\             GOTO 1105 IF N% = 0%
1110    X$ = SYS(CHR$(6%) + CHR$(11%)
              + STRING$(20%,0%)
              + "KB" + CHR$(N%) + CHR$(255%))
                                                !DEASSIGN A DEVICE
1115    X$ = SYS(CHR$(6%) + CHR$(-5%) + CHR$(N%)
              + "%Terminal now available"
              + CHR$(13%) + CHR$(10))
                                                !SEND SYSTEM LIMITED PHRASE
1120    A(N%) = 0%                              !DELETE FROM ASSIGNED TO TABLE
1130    B(N%) = N%                              !ADD TO UNASSIGNED TABLE
1140    RETURN                                  !FINISHED

1200 !*****************************************
\    !*
\    !* ASSIGN KB:
\    !*
\    !*****************************************

1203    INPUT "Assign KB:";N%                   !WHICH KB
1204    PRINT "%Console can not be assigned"
              IF N% = 0%
\             GOTO 1203 IF N% = 0%
1205    IF A(N%) > 0
              THEN PRINT
                 "%Device already assigned"
\             GOTO 1250
                                                !ERROR MESSAGE
1207    X$ = SYS(CHR$(6%) + CHR$(9%) + CHR$(0))
\       T% = (ASCII(MID(X$,2,1))/2)             !T% HAS KB NUMBER
1208    PRINT "%Current terminal can not be";
              " assigned" IF T% = N%
\             GOTO 1250 IF T% = N%              !CAN'T ASSIGN TERMINAL
                                                 AT WHICH LIMIT IS CURRENTLY
                                                 RUNNING
1210    X$ = SYS(CHR$(6%) + CHR$(10%)
              + STRING$(20%,0%)
              + "KB" + CHR$(N%) + CHR$(255%))
                                                !ASSIGN THE KB CODE
1215    X$ = SYS(CHR$(6%) + CHR$(-5%) + CHR$(N%)
              + "%System will be limited"
              + " until: "
              + B$ + CHR$(13%) + CHR$(10))
                                                !SEND SYSTEM LIMITED PHRASE
1220     A(N%) = N%                             !PLACE TKB IN ASSIGNED TABLE
1250    RETURN                                  !FINISHED

1300 !*****************************************
\    !*
\    !* LIST ROUTINE
\    !*
\    !*****************************************

1310    FOR X = 9 TO J%                         !LOOP THRU KB:
1320            PRINT "KB";X;":  "; IF A(X) > 0 !PRINT IF ASSIGNED
1330    NEXT X
1335    PRINT
1340    RETURN                                  !FINISHED

1400 !*****************************************
\    !*
\    !* FREE ROUTINE
\    !*
\    !*****************************************

1410    FOR X = 9 TO J%                         !LOOP THRU KB:
1420            PRINT "KB";X;":  "; IF A(X) = 0 !PRINT IF UNASSIGNED
1430    NEXT X
1435    PRINT
1440    RETURN                                  !FINISHED

1500 !*****************************************
\    !*
\    !* DETACH THE LIMIT JOB
\    !*
\    !*****************************************

1510    PRINT "Detaching....."                  !PRINT DETACHING ON KB
1520    X$ = SYS(CHR$(6%) + CHR$(7%))           !THE DETACH CODE
1530    RETURN

2000 !*****************************************
\    !*
\    !* TIME CHECK
\    !*
\    !*****************************************

2010    IF B$ < TIME$(0) THEN 32766             !TIME TO STOP
2030    RETURN                                  !FINISHED
5000 !*****************************************
\    !*
\    !* MULTI-TERMINAL SERVICES
\    !*
\    !*****************************************

5010    FIELD #12%, 7% AS Z9$
\       LSET Z9$ = ""
5020    GET #12%,
           RECORD 32767% + 1% + 16384% + 8192%  !SEE IF ANYTHING OUT THERE
5030    IF MID(Z9$ + "",2%,6%) = P9$ THEN 6000  !INPUTTED PASSWORD
5040    PRINT #12%, RECORD 32767% + 1%
                + ASCII(LEFT(Z9$ + "",1%)),
                C$
5050    GOTO 5010                               !RETURN IS DONE FROM ERROR R.
5060    RETURN

6000 !*****************************************
\    !*
\    !* PART OF MULTI-TERMINAL SERVICES,
\    !*          DEASSIGN TERMINAL
\    !*
\    !*****************************************

6010    X$ = SYS(CHR$(6%) + CHR$(11%)
                + STRING$(20%,0%)
                + "KB" + LEFT(Z9$ + "",1%)
                + CHR$(255%))
                                                !DEASSIGN A DEVICE
6015    X$ = SYS(CHR$(6%) + CHR$(-5%)
                + LEFT(Z9$ + "",1%)
                + "%Terminal is now available"
                + CHR$(13%) + CHR$(10))
                                                !SEND SYSTEM LIMITED PHRASE
6020    A(N%) = 0%                              !DELETE FROM ASSIGNED TO TABLE
6030    B(N%) = N%                              !ADD TO UNASSIGNED TABLE
6040    GOTO 5010

7000 !*****************************************
\    !*
\    !* SELECTED TERMINALS ONLY.
\    !*
\    !*****************************************

7010    PRINT "Enter terminal numbers";
                " (end with <cr>)"
7020    INPUT "KB number";N%                    !INPUT TERMINAL NUMBER
7030    GOTO 125 IF N% = 0%                     !TERMINATE IF ZERO
7040    X$ = SYS(CHR$(6%) + CHR$(9%) + CHR$(0)) !GET KB NUMBER
7050    T% = (ASCII(MID(X$,2,1))/2)             !T% HAS KB NUMBER
7060    PRINT "%Current terminal can not be";
                " assigned" IF N% = T%
\               GOTO 7020 IF N% = T%            !DON'T ASSIGN CURRENT TERMINAL
7070    GOSUB 1205                              !GOTO ASSIGN ROUTINE
                                                 AND ASSIGN TERMINAL
7080    GOTO 7020                               !KEEP PROCESSING

8000 !*****************************************
\    !*
\    !* SEIZE A TERMINAL
\    !*
\    !*****************************************

8010    INPUT "Terminal to seize";N%            !INPUT THE TERMINAL NUMBER
8020    PRINT "%Console can not be seized"
                IF N% = 0%
\       GOTO 8010 IF N% = 0%                    !ERROR, CONSOLE TERMINAL
8030    PRINT "%Terminal already limited"
                IF A(N%) <> 0%
\       GOTO 8010 IF A(N%) <> 0%                !TERMINAL ALREADY ASSIGNED
8040    X$ = SYS(CHR$(6%) + CHR$(9%) + CHR$(0))
\       T% = (ASCII(MID(X$,2,1))/2)             !T% HAS KB NUMBER
8042    PRINT "%Current terminal can not ";
                "be assigned" IF T% = N%
\               GOTO 8100 IF T% = X%            !CAN'T ASSIGN TERMINAL
                                                 AT WHICH LIMIT IS CURRENTLY
                                                 RUNNING
8044    X$ = SYS(CHR$(6%) + CHR$(-4%)
                + CHR$(N%) + CHR$(3%))
                FOR X = 1 TO 3
\                                               !FORCE 3 ^C TO TERMINAL
8050    X$ = SYS(CHR$(6%) + CHR$(-4%)
                + CHR$(N%) + "BYEF"
                + CHR$(13) + CHR$(10))
\                                               !FORCE THE BYEF
8060    U9% = 0%                                !INIT COUNTER FOR ATTEMPTS
                                                 TO SEIZE TERMINAL
8070    X$ = SYS(CHR$(6%) + CHR$(10%)
                + STRING$(20%,0%)
                + "KB" + CHR$(N%)
                + CHR$(255%))
                                                !ASSIGN THE KB CODE
8080    X$ = SYS(CHR$(6%) + CHR$(-5%)
                + CHR$(N%)
                + "%System will be"
                + " limited until: "
                + B$ + CHR$(13%) + CHR$(10))
                                                !SEND SYSTEM LIMITED PHRASE
8090    A(N%) = N%                              !PLACE TKB IN ASSIGNED TABLE
8100    RETURN                                  !FUNCTION FINISHED
```

```
9000 !*****************************************
\    !*
\    !* CHANGE PARAMETER ROUTINES
\    !*
\    !*****************************************

9010    INPUT "Time to stop";Q1$                !NEW TIME TO STOP
9020    B$ = Q1$ IF Q1$ <> ""                   !NULL MEANS NO CHANGE
9030    INPUT "Password change";Q1$             !NEW PASSWORD
9040    GOTO 9050 IF Q1$ = ""
\       PRINT "%Password must be 6 characters";
              " long" IF LEN(Q1$) <> 6
\       GOTO 9030 IF LEN(Q1$) <> 6
\       P9$ = Q1$
\                                               !PASSWORD VERIFIED AND CHANGED
9050    INPUT "Message change";Q1$              !NEW MESSAGE
9060    C$ = Q1$ IF Q1$ <> ""                   !NULL MEANS NO CHANGE
9070    RETURN                                  !FUNCTION FINISHED

9500 !*****************************************
\    !*
\    !* VIEW ROUTINE
\    !*
\    !*****************************************

9510    PRINT "Time to stop: ";B$               !PRINT TIME TO STOP
9520    PRINT "Password is: ";P9$               !PRINT PASSWORD
9530    PRINT "Message is: ";C$                 !PRINT MESSAGE
9540    RETURN                                  !FUNCTION FINISHED

31000 !****************************************
\     !*
\     !* ERROR ROUTINE
\     !*
\     !****************************************

31005   IF ERR = 28 THEN RESUME 32760           !^C TYPED
31007   IF ERR = 11 AND ERL > 130
              THEN RESUME 219
\                                               !^Z TYPED AND BEYOND
                                                 LINE 140 IN CODE
31010   IF ERL = 17 THEN RESUME 31900           !MASTER TERMINAL NOT AVAILABLE
31100   IF ERL = 90 THEN RESUME 120             !DEASSIGN FAILED
31110   IF ERR = 15 THEN PRINT
\             RESUME 217                        !KB WAIT EXHAUSTED
31120   IF ERL = 1210
              THEN PRINT "%Device not";
                   " available"
\             RESUME 1250
31125   IF ERL = 5020 THEN RESUME 5060          !TERMINAL SERVICE ERROR
31130   IF ERL = 8070 THEN RESUME 31800         !SEIZE DID NOT WORK
31140   IF ERL = 31930 THEN RESUME 31910        !ERROR ON NEW MASTER TERMINAL
31190   ON ERROR GOTO 0
31200   GOTO 32767                              !UNKNOWN ERROR-TERMINATE

31800 !****************************************
\     !*
\     !* SIEZE TERMINAL ERROR SECTION
\     !*
\     !****************************************

31810   U9% = U9% + 1                           !INCREMENT COUNTER
31820   SLEEP 1%                                !SLEEP ONE SECOND
31830   GOTO 8070 IF U9% < 10%                  !TRY 10 TIMES TO GET IT
31840   PRINT "%Terminal can not be seized";
              " - try again"
31850   GOTO 8100                               !GOTO RETURN STATEMENT

31900 !****************************************
\     !*
\     !* MASTER TERMINAL NOT AVAILABLE
\     !*
\     !****************************************

31910   PRINT "?Master terminal not available"
31920   PRINT "Input another terminal to be";
              " the master";
\       INPUT K%
\       K$ = "KB" + NUM1$(K%) + ":"             !GET NEW MASTER TERMINAL
                                                 NUMBER
31930   OPEN K$ AS FILE #12%                    !OPEN THE MASTER TERMINAL
31940   GOTO 20                                 !RETURN PROCESSING

32760 !****************************************
\     !*
\     !* EOJ
\     !*
\     !****************************************

32766   X$ = SYS(CHR$(6%) + CHR$(12%))          !DEASSIGN ALL DEVICES
32767   X$ = SYS(CHR$(6%) + CHR$(-5%) + CHR$(X)
         + " %LIMIT going offline--SYSTEM back "
         + "to normal"
         + CHR$(13) + CHR$(10%))
         IF A(X) <> 0 FOR X = 9 TO J%
\       X$ = SYS(CHR$(6%) + CHR$(-5%)
          +  CHR$(0%)
          + "%LIMIT going offline--SYSTEM back"
          + " to normal"
          + CHR$(13) + CHR$(10%))
\       X$ = SYS(CHR$(6%) + CHR$(9%)
          + CHR$(0%))
\       X1$ = SYS(CHR$(6%) + CHR$(8%)
          + CHR$(PEEK(518)/2)
          + SPACE$(23) + CHR$(0%) + CHR$(255%))
            IF ASCII(MID(X$,2,1)) > 80%
\       END                                     !ALL DONE AND OVER,
                                                 DETERMINE WHETHER
                                                 TO KILL THE JOB OFF
                                                 OR NOT THEN DO IF NECESSARY
$
```

♥

A MENU MANAGEMENT PROGRAM

By R. David Broom
Omega Optical
13515 N. Stemmons
Dallas, TX 75234
(214) 241-4141

INTRODUCTION

Every site provides some way of informing its personnel what options are available to them. This is usually accomplished through a menu display. Once the list of options is displayed, the operator keys in a numeric entry or a mnemonic name for the option desired and the menu program chains to the program associated with that option.

OBJECTIVES and GOALS

"MENU" is a menu management program where mnemonic names are associated with the option desired. In the development of "MENU" four goals were set:

1) Lower the number of keystrokes entered by an operator,

2) Lower the number of characters outputted to a terminal,

3) "Hide" the fact that the operator is executing a system program like PIP, DTR, ATPK, et cetera,

4) Allow the operators to do as much as possible on one line of entry. That is, pass user-selected parameters.

Implementing these four goals will give the appearance that the operator is doing more with less work. For example, if we have a program that cancels a customer's order, then under this type of menu management program the operator keys in the following:

ME CAN ACCOUNT.INVOICE

where ME means menu
CAN means cancel account number, invoice number

If we did not allow the passing of parameters, the operator would have typed the same number of keystrokes, but not on the same line. He/she would also have waited for the menu display, and the display of the cancellation program.

Some general goals where added:

1) Provide the ability to page within a menu,
2) Call one menu while in another menu,
3) Provide operator security checks,
4) Pass user parameters by CCL and chaining,
5) Develop a menu that is easy to modify,
6) Display options according to terminal type.

UNITRONIX represents DIGITAL

PDP 11/23 SYSTEMS with 256kB* memory, 160 Mbyte winchester, std. 45 ips 9-track tape backup, VT101 CRT, LA120 printer. Runs RSX11, RSTS, TSX OR UNIX. $32,000 *up to 4MB avail.

TERMINALS
VT100 LA12
VT101 LA34
VT102 LA50
VT125 LA100
VT131 LA120

CPUs
PDP 11/23, 11/44 & VAX

INTERFACES
MODEMS

COMPLETE DATA SYSTEMS
with standard and custom software

VT, LA, LSI, PDP, VAX are trademarks of Digital Equipment Corporation

SPECIAL
LSI-11 MODULES, PERSONAL COMPUTERS

WHY YOU SHOULD CALL UNITRONIX FIRST

- **Immediate Delivery from $6 mm Warehouse Inventory**
- **Complete Equipment Service Capability**
- **Excellent Pricing/Discount Structures**

In just over 6 years, Unitronix Corporation has emerged as a leading supplier of Digital Equipment Corporation (DEC) microcomputers, terminal products, printers, accessories and supplies on both a domestic and international level. Our new, ultra-modern 20,000 sq. ft. facility in Somerville, New Jersey, houses executive offices, in-house programming, customer service, sales and marketing, component testing and complete warehouse facilities.

UNITRONIX CORPORATION
197 Meister Ave., Somerville, NJ 08876
TELEX: 833184

AUTHORIZED digital® TERMINALS DISTRIBUTOR

CALL FOR LATEST PRICES

(201) 231-9400

CIRCLE 25 ON READER CARD

IMPLEMENTATION

To accomplish some of these goals a formatted text file was developed. This text file contains the mnemonic names of options, their description, what action is to be performed, program name or CCL or command file or DTR procedure or menu file to use with the action, line number to chain to and what terminal type can access this option. This type of text file is easy to modify and maintain. Figure 1 is an example of such a text file and Figure 2 is the display associated with it.

Security checks were implemented by using another action code: "O". The ways of doing a security check are infinite. One way is to create an operator file where each operator has a unique ID, and flag what secure options an operator can access. This is currently the type of system we are using.

Terminal type is checked because we have a mixture of VT52s and VT100s. Certain tasks were written to take advantage of the VT100's display options.

An interesting method of implementing this menu system is to apply the feature patch 21.3.5 which modifies the RSX run time system to chain to "MENU". An article with a related idea appears in the December issue of RSTS PRO, "A User Written Keyboard Monitor," by Ken Harris. A few modifications have to be made to "MENU" for this to work, such as the removal of the wait for entry.

Another approach is the feature patch 10.12.4 to modify "LOGIN" to chain to a specific program and add a CCL for "MENU". This is the approach we took. The entire menu is always displayed when our operators first log-in. Once an operator is logged in there is no need to redisplay the menu unless he/she is unsure of the options. To enter an option without displaying the menu listing, the operator enters a CCL for menu followed by an option, followed by parameters if they are allowed.

CONCLUSION

This type of menu system seems to work effectively. "MENU" accomplished our major goals and as a side benefit, operator training time has markedly decreased.

APPENDIX

Digital Equipment Corporation, "Invoking a Menu Program RSX.RTS Feature Patch," RSTS/E V7.2 Maintenance Notebook, (June 1982), Seq 21.3.5.

Digital Equipment Corporation, "Login Can Chain to a Specific Program - LOGIN Feature Patch," RSTS/E V7.2 Maintenance Notebook, (June 1982), Seq 10.12.4.

Harris, Ken, "A User Written Keyboard Monitor," RSTS Professional, (December 1982), Vol. 4, Num. 6


```
Customer Service Menu Selection ...
1;ACCOUNT;A/R account inquiry;C;INQUIR.BAC;0
 ;BYE;Log off the system;L;BYE/Y;0
 ;CLEAR;Clear all temporary files;T;CLEAR.PIP;0
 ;INQUIRY;General laboratory inquiry ;L;INQ;0
 ;LIST;Listing of current transactions for an account;P;ACCOUNT-TRANS;0
 ;MAIN;Main menu for Lab;S;$MAIN.MEN;0
 ;NIGHT;Produce nightly reports;A;NIGHT.LOG=NIGHT.ATP/DET;0
```

FIGURE 1

```
  ================================================================
!  Customer Service Inquiry Menu Selection ...                     !
!                                                                  !
[  ACCOUNT      - A/R account inquiry                              ]
[  BYE          - Log off the system                               ]
[  CLEAR        - Clear all temporary files                        ]
[  INQUIRY      - General laboratory inquiry by Job/Tray/Account   ]
[  LIST         - Listing of current transactions for an account   ]
[  MAIN         - Main menu for Lab                                ]
[  NIGHT        - Produce nightly reports                          ]

   OPTION>
```

FIGURE 2

```
10!*****************************************************************&
!                                                                   &
!       PROGRAM: MENU.B2S                                           &
!       VERSION: 1.1                                                &
!          DATE: 03-Nov-82                                          &
!        AUTHOR: R. David Broom                                     &
!          SITE: Omega Optical, Dallas,Tx                           &
!                                                                   &
!*****************************************************************&
!                                                                   &
!                  C O P Y R I G H T                                &
!                                                                   &
!                Copyright (C) 1982 by                              &
!                 Omega Optical Inc.                                &
!                   Dallas, Texas                                   &
!                                                                   &
!       This software is provided free of charge, and may be copied &
!       only with the inclusion of the author's name and this copyright &
!       notice. No title to or ownership of the software is hereby  &
!       transferred.                                                &
!                                                                   &
!       The information in this software is subject to change without &
!       notice. Neither the author nor Omega Optical Inc. assumes any &
!       responsibility for the use or reliability of its software.  &
!                                                                   &
!*****************************************************************&
!                                                                   &
!       <PROGRAM DESCRIPTION>                                       &
!                                                                   &
!               Menu management program.                            &
!                                                                   &
!               PURPOSE: to provide control and entry for multi-    &
!                        menu system.                               &
!                                                                   &
!               USE AND INSTALLATION:                               &
!                                                                   &
!               1. Install CCL for this program (@ 30000)           &
!                  CCL name can be any appropriate name.            &
!               2. This program chains and/or CCLs on filespecs     &
!                  provided in the control file MENU.TXT.           &
!               3. Build MENU.TXT using the format listed below.    &
!                  MENU.TXT must reside in the user account.        &
!                                                                   &
!               The first line is a title for the menu and cannot   &
!               exceeded 60 characters. The following lines are menu &
!               description lines.                                  &
!                                                                   &
!       Menu line construct:                                        &
!                                                                   &
!       T;OPTIONNAME;TEXT;ACTIONCODE;PROG.NAME;LINENO               &
!                                                                   &
!               WHERE:                                              &
!               T=1     OPTION valid for VT100 only.                &
!               T=5     OPTION valid for VT52 only.                 &
!               T=H     OPTION valid for hardcopy only.             &
!               T=" "   OPTION valid for any terminal type          &
!               OPTIONNAME                                          &
!                       up to 12 character name for menu option.
!               TEXT                                                &
!                       optional descriptive text (up to 60 characters).
```

```
!               ACTIONCODE                                              &
!                       A = Execute ATPK command file PROG.NAME         &
!                       C = chain to PROG.NAME at LINENO passing        &
!                           any other text in core common.              &
!                       D = Execute DTR command file PROG.NAME          &
!                       L = execute CCL using PROG.NAME as CCl and      &
!                           passing any other text as switches.         &
!                       P = DATATRIEVE procedure at PROG.NAME           &
!                       S = Switch to new menu control file at PROG.NAME&
!                       T = Execute a PIP command file                  &
!               PROG.NAME                                               &
!                       Execute name (program name or CCL)              &
!               LINENO                                                  &
!                       Line number used in chaining                    &
!                                                                       &
!       All items are separated by ";"                                  &
!                                                                       &
!       NOTES:                                                          &
!                                                                       &
!       MENU will page. Up to 15 items are displayed per screen.        &
!       This can be modified by increasing the arrays and changing      &
!       MENU.LIMIT%                                                     &
!                                                                       &
!*************************************************************************&

100!************************************************************************&
!                                                                       &
!               C O M M O N   D E C L A R A T I O N                     &
!                                                                       &
        COMMON  M%(30%)                 ! Sys call array for terminal   &
                ,KB.CHN%                ! Keyboard channel              &
                ,MENU.CHN%              ! Menu channel                  &
                ,PAUS.TIME%             ! Time limit on user input      &
                ,MENU.LIMIT%            ! Limit of menu items to display&
                ,OPTION%                ! Selected option               &
                ,OPT.LEN%               ! Length of option entry        &
                ,N%                     ! Menu item counter             &
                ,VTCHK%                 ! Terminal specification        &
                ,LINENO%(15%)           ! Line number used in chaining  &
                ,TEXT$=60%              ! Text to display               &
                ,TITLE$=60%             ! Title of menu                 &
                ,TERM.FLAG$=1%          ! Type of terminal for display  &
                ,OPTION$(15%)=12%       ! Option names                  &
                ,ACTION$(15%)=1%        ! Action to perform             &
                ,PROG.NAME$(15%)=26%    ! Execute name for program      &

1000!***********************************************************************&
!                                                                       &
!               M A I N     P R O G R A M                               &
!                                                                       &
        ON ERROR GOTO 19000                                             &
\       KB.CHN%=1%                                                      &
\       MENU.CHN%=2%                                                    &
\       PAUS.TIME%=60%                                                  &
\       MENU.LIMIT%=15%                                                 &
\       MAIN.MENU$="MENU.TXT"           ! Default menu, THIS ACCOUNT    &
\       OPEN "_KB:" AS FILE KB.CHN%,MODE 256%                           &
\       X$=SYS(CHR$(11%)+CHR$(KB.CHN%)) ! Clear type ahead              &
\       GOSUB 14000                     ! Test for VT100/VT52/HARDCOPY  &
\       OPTION%=0%                                                      &
\       IF CCL%                                                         &
        THEN    GOSUB 10000             ! Read MENU                     &
\               GOSUB 12000 IF OPTION%  ! Execute if possible           &

1010    CCL%,                                                           &
        OPTION%,                                                        &
        OPT.LEN%=0%                                                     &
\       OPT$=""                                                         &
\       GOSUB 10000                     ! Initialize MENU               &
\       GOSUB 11000                     ! Interpret user request        &
\       GOSUB 12000 IF OPTION%          ! Execute if possible           &
\       GOTO 1010                       ! Restart menu display          &

10000!**********************************************************************&
!                                                                       &
!               L O C A L    S U B R O U T I N E S                      &
!                                                                       &
!               Initialize the menu settings                            &
!                                                                       &
        OPEN MAIN.MENU$ FOR INPUT AS FILE MENU.CHN%                     &
\       LINPUT #MENU.CHN%,TITLE$                                        &
                                        ! Open menu file and get title  &

10010   N%=1%                                                           &
\       IF CCL%=0%                                                      &
        THEN    X%=FN.CLEAR.SCREEN%                                     &
\               PRINT TAB(3%);STRING$(73%,61%)                          &
\               PRINT "! ";TITLE$;TAB(79%);"!"                          &
\               PRINT "!";TAB(79%);"!"                                  &
                                        ! If no CCL entry then clear    &
                                        ! screen and display title      &

10020   LINPUT #MENU.CHN%,MENU.LINE$                                    &
\       TERM.FLAG$=LEFT$(MENU.LINE$,1%)                                 &
\       GOTO 10020 IF TERM.FLAG$="H" AND VTCHK%<>1%                     &
                                        ! This is a hardcopy option     &
\       GOTO 10020 IF TERM.FLAG$="5" AND VTCHK%<>2%                     &
                                        ! This is a VT52 option         &
\       GOTO 10020 IF TERM.FLAG$="1" AND VTCHK%<>3%                     &
                                        ! This is a VT100 option        &
```

```
\       SC.POS%=POS(MENU.LINE$,";",1%)                                  &
\       SC.POS1%=POS(MENU.LINE$,";",SC.POS%+1%)                         &
\       OPTION$(N%)=SEG$(MENU.LINE$,SC.POS%+1%,SC.POS1%-1%)             &
                                        ! Option name                   &
\       SC.POS%=POS(MENU.LINE$,";",SC.POS1%+1%)                         &
\       TEXT$=SEG$(MENU.LINE$,SC.POS1%+1%,SC.POS%-1%)                   &
                                        ! Option description            &
\       SC.POS1%=POS(MENU.LINE$,";",SC.POS%+1%)                         &
\       ACTION$(N%)=SEG$(MENU.LINE$,SC.POS%+1%,SC.POS1%-1%)             &
                                        ! Action to perform for option  &
\       SC.POS%=POS(MENU.LINE$,";",SC.POS1%+1%)                         &
\       PROG.NAME$(N%)=SEG$(MENU.LINE$,SC.POS1%+1%,SC.POS%-1%)          &
                                        ! Program or command file to use&
\       LINENO%(N%)=VAL(RIGHT$(MENU.LINE$,SC.POS%+1%))                  &
                                        ! Chaining point for program    &
\       OPTION%=N% IF OPT$=LEFT$(OPTION$(N%),OPT.LEN%) AND CCL%         &
\       RETURN IF OPTION%               ! If a CCL entry and option     &
                                        ! matches then save option value&
                                        ! and get out.                  &
\       GOTO 10020 IF CCL%              ! If a CCL entry do not display &
                                        ! option line                   &
\       PRINT "[ ";OPTION$(N%);" - ";TEXT$;TAB(79%);"]"                 &
\       N%=N%+1%                                                        &
\       GOTO 10020 IF N%<=MENU.LIMIT%                                   &
\       PRINT                                                           &
\       PRINT "[ To PAGE hit [RETURN] ...";TAB(79%);"]"                 &

10030   RETURN IF CCL%                  ! Should never get here on CCL  &
                                        ! unless no match is found      &
\       PRINT                                                           &
\       PRINT "  Option> ";                                             &
\       WAIT PAUS.TIME%                                                 &
\       LINPUT #KB.CHN%, OPT$                                           &
\       OPT$=EDIT$(OPT$,32%)            ! Make it upper case            &
\       GOTO 10010 IF OPT$="" AND N%>MENU.LIMIT%                        &
                                        ! Page for other options        &

10040   CLOSE #MENU.CHN%                                                &
\       RETURN                                                          &


11000!*****************************************************************&
!                                                                       &
!               Determine if user input is executable                   &
!                                                                       &
        X%=POS(OPT$,SP,1%)                                              &
\       USER.DATA$=EDIT$(RIGHT$(OPT$,X%+1%),8%+128%+32%) IF X%          &
                                        ! Save rest of entry line       &
\       OPT$=LEFT$(OPT$,X%-1%) IF X%                                    &
\       OPT.LEN%=LEN(OPT$)                                              &
\       RETURN IF OPT.LEN%<3%           ! Input must be at least 3 chars&
\       RETURN IF LEFT$(OPT$,I%)=LEFT$(OPTION$(OPTION%),I%)             &
                    FOR OPTION%=1% TO N%                                &
                            FOR I%=OPT.LEN% TO 3% STEP -1%              &
                                        ! Loop through all options      &
                                        ! trying to match user input    &
                                        ! with an option                &

11099   OPTION%=0%                      ! No match, so return back with &
\       RETURN                          ! a bad request                 &


12000!*****************************************************************&
!                                                                       &
!               Process MENU function code                              &
!                                                                       &
!       [ A ]  1.       Execute ATPK command file PROG.NAME             &
!       [ C ]  2.       Chain to PROG.NAME at LINENO                    &
!       [ D ]  3.       Execute DTR command file PROG.NAME              &
!       [ L ]  4.       Execute CCL PROG.NAME with USER.DATA$           &
!       [ P ]  5.       Execute DATATRIEVE procedure PROG.NAME          &
!       [ S ]  6.       Switch to MENU PROG.NAME                        &
!       [ T ]  7.       Perform a PIP command file                      &
!                                                                       &
        X$=FN.PUT.CORE.COMMON$("")      ! Clear core common             &
\       ON POS("ACDLPST",ACTION$(OPTION%),1%)                           &
                    GOSUB   12100,          ![ A ]!                     &
                            12150,          ![ C ]!                     &
                            12200,          ![ D ]!                     &
                            12250,          ![ L ]!                     &
                            12300,          ![ P ]!                     &
                            12350,          ![ S ]!                     &
                            12400           ![ T ]!                     &

12010   RETURN                                                          &

12100   PROG.NAME$="PER "+PROG.NAME$(OPTION%)                           &
\       X%=FN.CLEAR.SCREEN%             ! Clear screen                 &
\       X$=FN.CCL$(PROG.NAME$)          ! Perform syscall for CCL      &
\       RETURN                          ! Execute ATPK file PROG.NAME  &

12150   X%=FN.CLEAR.SCREEN%             ! Clear screen                 &
\       X$=FN.PUT.CORE.COMMON$(USER.DATA$) IF LEN(USER.DATA$)           &
                                        ! Put DATA in core common      &
\       CHAIN PROG.NAME$(OPTION%) LINE LINENO%(OPTION%)                 &
\       RETURN                          ! Chain to PROG.NAME at LINENO &

12200   PROG.NAME$="DTR @"+PROG.NAME$(OPTION%)                          &
\       X%=FN.CLEAR.SCREEN%             ! Clear screen                 &
\       X$=FN.CCL$(PROG.NAME$)          ! Perform syscall for CCL      &
\       RETURN          ! Execute DATATRIEVE command file PROG.NAME    &

12250   PROG.NAME$=PROG.NAME$(OPTION%)                                  &
\       PROG.NAME$=PROG.NAME$+" "+USER.DATA$ IF LEN(USER.DATA$)         &
\       X%=FN.CLEAR.SCREEN%             ! Clear screen                 &
\       X$=FN.CCL$(PROG.NAME$)          ! Perform syscall for CCL      &
\       RETURN                  ! Execute CCL PROG.NAME + user input   &

12300   PROG.NAME$="DTR :"+PROG.NAME$(OPTION%)                          &
\       X%=FN.CLEAR.SCREEN%             ! Clear screen                 &
\       X$=FN.CCL$(PROG.NAME$)          ! perform syscall for CCL      &
\       RETURN                  ! Do DATATRIEVE procedure PROG.NAME    &

12350   MAIN.MENU$=PROG.NAME$(OPTION%)  ! Get next menu                 &
\       RETURN                                                          &

12400   PROG.NAME$="PIP @"+PROG.NAME$(OPTION%)                          &
\       X%=FN.CLEAR.SCREEN%             ! Clear screen                 &
\       X$=FN.CCL$(PROG.NAME$)          ! Perform syscall for CCL      &
\       RETURN                                                          &
14000!*****************************************************************&
!                                                                       &
!               Determine Terminal Characteristics                      &
!                                                                       &
        CHANGE SYS(CHR$(6%)+CHR$(16%)+CHR$(0%)+CHR$(255%)+              &
                                        STRING$(24%,0%)) TO M%          &
                                        ! Get terminal characteristics &
\       VTCHK%=1%                       ! Hardcopy terminal            &
\       RETURN IF M%(11%)=128%          ! NO SCOPE set                 &
\       VTCHK%=2%                       ! VT52                         &
\       RETURN IF M%(9%)=128%           ! NO XON set                   &
\       PRINT #KB.CHN%,CHR$(155%);"[c"; ! Ask for VT100 I.D.          &
\       WAIT 1%                         ! Wait only 1 second           &
\       GET #KB.CHN%                                                    &
\       X%=RECOUNT                                                      &
\       MOVE FROM #KB.CHN%,X$=X%                                        &
\       VTCHK%=3% IF POS(X$,"[?1;",1%)                                  &
                        ! This should work for VT100 and CIT-101       &

14020   RETURN                                                          &

15000!*****************************************************************&
!                                                                       &
!               L O C A L   F U N C T I O N S                           &
!                                                                       &
        DEF FN.CLEAR.SCREEN%                                            &
\       PRINT CHR$(155%)+"[2J"+CHR$(155%)+"[0;0H"; IF VTCHK%=3%         &
\       PRINT CHR$(155%)+"H"+CHR$(155%)+"J"; IF VTCHK%=2%               &
\       PRINT                                                           &
\       FNEND                                                           &

15100   DEF FN.PUT.CORE.COMMON$(D$)=SYS(CHR$(8%)+D$)                    &
                                        ! Put DATA D$ in core common    &

15200   DEF FN.CCL$(D$)=SYS(CHR$(14%)+D$)                               &
                                        ! Do CCL                       &
19000!*****************************************************************&
!                                                                       &
!               S T A N D A R D   E R R O R   H A N D L I N G           &
!                                                                       &
        RESUME 32767 IF ERL=10030                                       &
                        ! User is quitting on time-out or CTRL/Z       &
\       RESUME 14020 IF ERL=14000                                       &
                        ! Time-out waiting on VT100 I.D. call          &
\       RESUME 11099 IF ERL=11000                                       &
                        ! Subscript out of range on a CCl entry        &
\       RESUME 10030 IF ERL=10020 AND ERR=11%                           &
                        ! End of file on read                          &
\       RESUME 10020 IF ERL=10020 AND (ERR=52% OR ERR=55%)              &
                        ! Line number specification is illegal number  &
                        ! or subscript out of range                    &

19010   IF ERR=5%                                                       &
        THEN    PRINT "That option does not exist" IF ERL=12150         &
\               RESUME 32767                                            &
                        ! Tried to chain to a non-existent program     &
                        ! Menu file does not exist. Just act like      &
                        ! nothing happen.                              &

19900   PRINT "Error: ";ERR;" ";ERT$(ERR);" at line ";ERL               &
\       RESUME 32767                                                    &

30000!*****************************************************************&
!                                                                       &
!               C C L   E N T R Y   P O I N T                           &
!                                                                       &
        ON ERROR GOTO 19000                                             &
\       OPT$=SYS(CHR$(7%))                                              &
\       CCL%=0%                                                         &
\       USER.DATA$=""                                                   &
\       X%=POS(OPT$,SP,1%)                                              &
\       OPT$=RIGHT$(OPT$,X%+1%)                                         &
\       GOTO 1000 IF LEN(OPT$)<3%                                       &
                                        ! Assume not a CCL entry if    &
                                        ! input is less than 3 chars   &
\       X%=POS(OPT$,SP,1%)                                              &
\       USER.DATA$=RIGHT$(OPT$,X%+1%) IF X%                             &
\       OPT$=LEFT$(OPT$,X%-1%) IF X%                                    &
\       OPT.LEN%=LEN(OPT$)                                              &
\       CCL%=(OPT.LEN%>=3%)                                             &
\       GOTO 1000                                                       &
                                        ! Pull-off any user parameters  &
                                        ! and save them                 &

32767   END                                                             &
```

NEW PRODUCTS

ROSS OFFERS STOCK OPTION SOFTWARE SYSTEM

A stock option software system that manages and tracks all employee stock option plan information is now available from Ross Systems. The new accounting and tracking system meets all current legislative requirements for incentive stock options (ISO) and provides complete information for public, tax and internal reporting of stock option programs.

Through its database design, Ross' Stock Option System gives users complete control over their option plan to accommodate changing regulations. As an interactive system, it tracks all activity relating to the granting, exercising, cancellation and repricing of employee stock options. Additionally, the system meets Federal Incentive Stock Option (ISO) accounting and reporting requirements. The Ross System is available to run on Digital Equipment Corporation's PDP-11 Series and VAX series minicomputers under RSTS/E and VMS operating systems. It provides a selection of standard reports, accommodates specific inquiries about participants and can interface with both company personnel and compensation systems. It can accept information regarding employee status or location on a daily basis and is compatible with a number of communications formats.

The Ross Stock Option System is based on Ross' interactive INTAC database management system for organizing and reporting business information. Ross Systems, Inc. provides financial decision support, applications and database software for financial managers in manufacturing, business services and banking. Software is available for Digital Equipment Corporation minicomputers or through Ross' worldwide timesharing network. For information, contact Jack Brown at Ross Systems, 1860 Embarcadero Road, Palo Alto, CA 94043, (415) 865-1100.

DIGITAL ANNOUNCES DECmail/RSTS

In 1860, the century's most modern method of delivering the mail galloped into the daily lives of many of the people of this country. Relays of swift horses and unflinching men — the Pony Express — raced from St. Joseph, Missouri to Sacramento, California in record time.

Later came Wells Fargo coaches and four. They sped mail, cargo and passengers ever faster. Trains and planes followed. But even they were not the ultimate. Because now, in 1983, an entirely new and incredibly faster method of delivering messages has just been announced by Digital.

This new method enables RSTS/E users to receive and send messages to business associates down the hall, across town, around the country and overseas — elecronically. To individuals or entire groups all at once. To file and retrieve and edit these messages almost without effort at enormous savings in cost and time. It's called DECmail/RSTS and it's here now.

DECmail/RSTS was sired, reared and broken in just for you. It was designed by the original RSTS/E development team — the computer gang that knows the RSTS/E system intimately. Their experience and expertise has made DECmail/RSTS a downright superior product.

What does DECmail do?

Everything an electronic message system should. With DECmail, you can create, edit, send and process messages. The message could be a simple memo: "Your preliminary figures for quarter just ended seem too good to be true. Please check and verify at once."

Or a complicated specification completed overnight and forwarded for first morning review: "Report rechecked and confirmed. Here's the detailed breakdown for immediate action."

DECmail also stores messages it receives while you are out. Before you begin your work in the morning, for example, it brings you up-to-date. "You have 3 new messages." And it shows you their essence and whom they're from.

There's more.

As for filing systems, DECmail will organize your material for you — easily, efficiently, electronically. It stores, searches for and retrieves messages held in your files.

If this sounds as easy as falling off a horse, it is. DECmail designers knew RSTS/E software had to be sophisticated enough for advanced users and yet simple and uncomplicated for computer novices. DECmail is both.

For beginners, easy-to-learn commands and flexible options make learning a cinch. If there's trouble along the way, the on-line help facility can solve problems fast. For advanced users, DECmail offers many features including the flexibility to support many project teams by sharing specified folders. This way, everyone keeps abreast of the latest data. Managers, project leaders, researchers, administrators — all communicate quickly, easily and more effectively with DECmail.

Who can use DECmail/RSTS Version 1.0?

DECmail runs on RSTS/E Version 7.2 and 8.0. No update will be needed for DECmail/RSTS until after RSTS/E V8.0 is released. Thanks to DECmail update kits, you won't be saddled with an obsolete software product down the line.

Specifically, DECmail/RSTS allows single-node use on a single RSTS/E system. When used with DECnet/E, it provides multi-node use. Multiple standalone RSTS/E systems running DECmail/RSTS can be connected in a network, allowing users on one system to exchange mail with users on other systems. VAX/VMS systems running the VMSmail facility also can exchange messages with systems running DECmail/RSTS through DECnet.

DECmail/RSTS can be used from any RSTS/E command terminal. Supported terminals include word processing systems (WS78, WS278 and DECmate) running CX protocol.

All this is available now for only $3,000 per system (only $1500 per additional license). And just as important, DECmail can increase productivity and improve your working environment. It is a new way for people in business to communicate with each other.

Order your DECmail/RSTS software

now. As a gesture of appreciation for acting promptly, Digital will send you an authentic western belt buckle as a gift. Only one per pardner, please. Call toll free (800) DEC-INFO any working day between 8:30 a.m. and 5:00 p.m. EST.

NORTHWEST DIGITAL SOFTWARE ANNOUNCES RPM — RSTS

RPM is a new performance optimization package from Michael Mayfield and Northwest Digital Software. RPM can drastically improve total system performance by identifying problem areas in system usage, the programs that caused the problems and the problem areas within each program. RPM is the only product that can do all this.

System tuning with RPM uses a step by step "cookbook" approach. No knowledge of system tuning or monitor internals is required. Problem areas are identified using an automatic procedure which provides a report describing, in plain English, not numbers, where the system performance can be improved. It even makes suggestions for improvement.

On-line plotting, histograms and other reports can be used to further identify problem areas. Extended monitor data collection allows plotting of information not normally available, such as seek distance, disk usage, and cache, memory, file processor (FIP), and small buffer utilization. The programs causing the problems are then identified and can be examined in extreme detail. Detailed examination of a program includes CPU usage, a count of I/O requests and disk overhead by channel and a count of monitor calls and disk overhead by call. For information, contact Northwest Digital Software, Inc., Box 2-743, Spring Valley Road, Newport, WA 99156, (509) 447-2620.

REPORTING SYSTEM AVAILABLE FOR VAX

Enterprise Technology Corporation has announced the release of the Generalized Reporting System (GRS) for Digital Equipment Corporation VAX-11 computers. GRS is a user-friendly report writer, query and audit package designed for use with a wide variety of file formats and database systems, and is a low-cost alternative to Digital's Datatrieve facility.

GRS is compatible with the applications packages from most major software vendors, as well as with systems developed by end users. It includes a powerful English-like query language which can be used by both programmers and non-programmers. With GRS, management personnel can meet their own needs for ad-hoc information retrieval, and systems personnel can vastly decrease the amount of time spent responding to user requests for reports. GRS is also ideal for use by auditors.

The GRS software for VAX systems is supplied under a perpetual license for a one-time fee of $6,500.00 which includes full documentation and one year of support. More information can be obtained by contacting Enterprise Technology Corporation at 305 Madison Avenue, New York, NY 10165, (212) 972-1860.

SPSORT FOR RSTS/E

The System Performance House, Inc. is pleased to announce a new high-performance RSTS/E sort called SPSORT.

— SPSORT can sort a file onto itself using no scratch space except for a 5 block command file.

— SPSORT can sort a file from one device onto another file on another device.

— SPSORT sorts a wide variety of data types including all Basic-Plus CVT types, all 8-, 16-, 32- and 64-bit signed and unsigned data types, FORTRAN IV-Plus I*4, DIBOL decimal, packed, and EBCDIC. In addition, the user may specify any two byte-collating sequences desired.

— SPSORT will sort any type of fixed length record file whose record length is 512 bytes or less and whose total size is one billion bytes or less. There is no limit on record count.

— SPSORT is extremely fast. In one test, SPSORT sorted 100,000 64-byte records with 23 byte keys in 16 minutes on an 11/24 with RK07 disk drives.

— SPSORT has an extremely versatile calling format enabling the user to chain into and out of it in a number of ways.

Available with SPSORT are "front-end" modules which emulate OMSI SORT-1 Plus from Oregon Software, FSORT3 from E. G. and H., and DEC's SORTG/SORTM without change in user software!

SPSORT, complete with emulators, is attractively priced. The System Performance House will provide a superior sort at a lower price than its competitors. Fixed priced unlimited licenses are available to authorized DIGITAL distributors and OEM's at exceptionally low rates. For specifications and pricing information, contact the System Performance House, Inc., 5522 Loch More Court, Dublin, OH 43017, 614-265-7788.

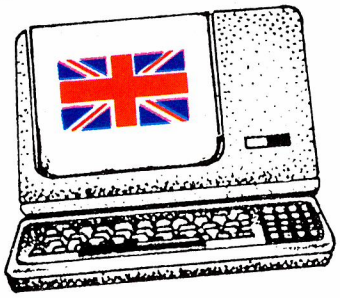
VSELECT FOR VAX FROM EG&H

Evans Griffiths & Hart, Inc. has announced the release of a new product, VSELECT, a high speed machine-language record extractor for the VAX under VMS. The package handles RMS fixed-record-length sequential, relative, and indexed files (prologue 2 only).

VSELECT employs user-specified selection criteria to extract from one or more input files those records needed for specific report or application. Optionally, the extracted records are modified by the deletion, insertion, or rearrangement of fields. The records are then written to an output file that may be sorted by VSORT, EGH's fast machine-language sort package. VSORT may be invoked directly through VSELECT.

Selection keys for VSELECT include ASCII, signed and unsigned integer, floating point (single, double, giant, or huge), packed decimal, and the relative file record occupancy flag.

VSELECT supports a number of sophisticated options, among which are: (1) hooks for user-coded selection criteria and output field transformations; (2) conditional field copying; (3) the conditional generation of multiple output records from a single input record; (4) the generation of backpointers into the input file(s) —



You are invited to

London International Press Centre

on

Tuesday 28th June 1983
From 10 am to 6 pm

to attend a seminar presented by

Carl Marbach
Dave Mallery
Al Cini

The Team from DEC Professional, RSTS Professional and Personal and Professional Magazines in the U.S.A.

Price: £50, if payment received before 6th June 1983 and £60 thereafter

and includes:
Morning Coffee & Biscuits
Three-Course Lunch with Wine
Afternoon Tea & Biscuits

For more information:

Digital Equipment Computer Users Society
P.O. Box 53, Reading, RG2 0TW
Telephone: Reading (0734) 387725
Telex: 848327/8

including RFAs for RMS files; and (5) the insertion of string literals in output records.

VSELECT is callable from any of the VAX native mode languages that support the standard DEC procedure calling conventions. In addition, the package is supplied with a self-contained interactive interface that can accept partial or complete command files.

A single-CPU license for VSELECT is $1500.00, discounted to $500.00 when the package is purchased with VSORT. OEM discounts are available. Contact Evans Griffiths & Hart, Inc., 55 Waltham St. Lexington, MA 02173. Tel: (617) 861-0670.

STAR PLAN INTRODCES MARCO LANGUAGE UTILITY

Star Plan Data Processing of Milwaukee, Wisconsin is pleased to announce MPR, a multipurpose macro language utility. MPR is a system engineering tool designed primarily as a program language preprocessor. The basic technical principles involved are the reduction in volume of the code body, and the addition of a level of abstraction above the source language. This decreases the cost of maintenance and increases the flexibility and portability of the application system.

Users of the utility are able to work it with anything that is kept in text files. Examples include BASIC+2, COBOL, DIBOL, TECO, BASIC+, RPG, C, RNO, PASCAL, LISP, ATPK and batch control files, and documentation.

MPR includes features for interactive applications and is frequently used to tailor control files according to the user's specific requirements.

The product's single most common application is sharing data definitions used by many programs in a system. For example, if you put record definitions for an application into external files and ".INCLUDE" them in the programs in the application, the entire application can be rebuilt with the new definitions after changing only one file.

With individual programs, "common" areas can be defined in one place for a program with many external subroutines. This can save considerable time when compared to the conventional technique of editing each module separately to bring them up to date.

Another practical application for MPR is source-level libraries of program code. By using MPR on standard system routines, programmers can have the advantages of standard libraries without the memory and runtime cost of automatically bringing in unnecessary code.

OEMs often need to create several versions of a software product with minor, but pervasive differences. Using MPR, it is possible to have a single body of source code for a class of very similar applications, which expands differently for different clients or operating systems. It is also possible to create an integrated software package and unbundle various features by either including or excluding the relevant code when creating a client's system.

MPR is a highly versatile software tool that will benefit large scale application programming jobs. It has proven its adaptability to a wide range of specific in-house methodologies.

MPR is priced at $875.00 for a single-CPU binary license and is available for all of Digital Equipment Corporation's PDP-11 series minicomputers using RSTS/E, and will soon be released under VAX VMS and RSX-11M+. Further information about MPR can be obtained from Star Plan Data Processing, 2040 W. Wisconsin Avenue, Suite #354, Milwaukee, WI 53233, ATTN: Noah M. Dixon, (414) 933-0800.

BASIC+, BASIC+2, DIBOL, RSTS/E, VAX VMS, and RSX11 are trademarks of Digital Equipment Corp.

MPR is a trademark of Star Plan Data Processing, Inc.

FREE LITERATURE FROM CROSS

Cross Information Company is offering a series of articles entitled: Advanced Business Communications: Applications for Computer Tele/conferencing. They are designed to help businesses and educational institutions develop and implement productive and cost-effective computer tele/conferencing on VAX systems.

The CIC series offers practical advice on planning, cost justification, and strategies for using computer tele/conferencing. CIC has organized these articles into specific applications:

- Association management
- Improving software development and programmer productivity
- Managing the sales/marketing department
- Educational and training applications
- Uses in quality/study circles
- Telecommunications management
- Corporate strategic planning
- General corporate management, and
- Legal office and education.

Requestors are asked to indentify which articles are desired. There is no charge for these articles. There are available from Cross Information Company, Corporate Communications, 934 Pearl Mall — Suite B, Boulder, Colorado 80302-5181, 303-499-8888.

EDT MANUAL FROM COMPUTEREASE

DEC's EDT editor, available for its PDP-11 and VAX product lines, is a flexible, powerful editor. However, the complexity of the EDT manual has intimidated many of DEC's less experienced users, programmers as well as clerical workers. Many users learn as much as they need to know for their basic functions and then end their learning process, as far as EDT goes. As a solution to this problem, ComputerEase Publishing announces the availability of their latest publication, "Understanding EDT."

Recognizing that a good manual takes technical skill as well as good writing ability, ComuterEase teamed a programmer with an experienced technical writer to produce the manual. To further ensure the technical accuracy of the manual, all composition was performed using the EDT Editor itself.

"Understanding EDT" is especially

worthwhile for novices; it begins by explaining how to use a CRT and the basic concepts of using a text processor. The command section of the manual progresses in difficulty from the basic commands to the more advanced ones. In presenting the material this way, the aim is bring the user along gradually, so that he or she gains more capability with each succeeding chapter. "Understanding EDT" covers all three editing modes: keypad, nokeypad, and line mode, with the emphasis on keypad mode. In addition, the manual discusses strategies in using the editor and provides examples for applying it in text processing and programming environments.

The "Understanding EDT" manual contains more than 150 pages of instructional material. It is available from ComputerEase Publishing for $49.95 per manual in quantities of nine or less, and for $34.95 each in quantities of ten or greater. To place an order or for more information, contact Director of Marketing, ComputerEase Publishing, P.O. Box 390272, Mountain View, CA 94039.

PRODUCT UPDATES

SORTC FOR RSTS FROM JBM GROUP

The JBM Group announces that its SORTC Compounding Sort V2.5 is now available for PDP-11s using the RSTS/E operating system. This disk sort, written entirely in MARCO-11, is one of the many software products in JBM's MIDWARE line.

Originally written as a high-speed, direct replacement for DEC's SORTG/SORTM DIBOL sort package, it is now expanded to handle DMS-500 Type 1 and Type 2 files. This permits use in BASIC-Plus and BASIC-Plus-2 shops that use native mode RSTS block I/O.

SORTC operates in standalone mode or is available in subroutine form to be called from within a DIBOL, MACRO-11 or BASIC-Plus-2 program.

The compounding feature permits summing of selected numeric fields, consolidating data records and producing totals without the need for a post-processing routine. This reduces the number of times data records are shuffled about during sorting thus limiting both CPU usage and disk requirements.

SORTC may be fine-tuned by the user to balance system resource loading. Complete documentation includes numerous examples of option use.

SORTC is available from stock on either 800 or 1600 bpi 9-track tape. The standalone version, including basic sort mechanics, is priced at $1,400.00. The optional subroutine capability is an additional $500.00. If purchased together the price is $1,700.00. For further information or to order SORTC, contact: The JBM Group, Inc. Dept. 22B, 332 West Church Road, King of Prussia, PA 19406, TWX 510-660-3999, TEL 215-337-3138.

PORTRAIT DISPLAY OPTION FOR AMBASSADOR

Ann Arbor has announced a portrait display configuration of the Ann Arbor Ambassador for text editing applications. This configuration uses a vertical screen to provide a full-page display in an 8-½ x 11 format.

The Portrait Display Ambassador, unlike other full-page terminals, allows the user to select the display size most appropriate for the application. If small characters become tiring, the user can "zoom" part of the text off-screen to increase character size, yet instantly "zoom" it back to a full-page display for context or review and final editing. This multiple-format capability also permits use of existing programs written for conventional 24-line terminals while the user adds full-page programs to his or her library.

The Ann Arbor Ambassador is especially suited to text editing because it is one of the few interactive terminals available that can be made to locally rearrange data on the screen. In addition, its editing commands are controlled by hardware line pointers rather than firmware, which speeds up their execution and virtually eliminates the need for pad characters. Its ANSI-standard coding permits the use of decimal parameters in the edit commands so the application software can insert or delete multiple lines or characters without repetitively sending the same command. A meta shift is provided for use with editors, such as EMACS, that use the parity bit for data transmission.

Other features of the Ambassador include detachable keyboard with sixty programmable key levels, a split-screen operation with user-definable scrolling regions, multiple page/window capability, printer output, and both block-mode and KSR operation. Options include DEC mode for VT100 software compatibility and a tilt/swivel stand.

The Portrait Display Ambassador is available from Ann Arbor distributors or from the factory direct. The suggested retail price is $1795; delivery is from stock to 30 days. For further information, contact Kathryn Straith, Ann Arbor Terminals, Inc., 6175 Jackson Road, Ann Arbor, MI, 313/663-8000.

COMBOARD/3780 NOW AVAILABLE FOR VAX, PDP-11S

Software Results Corporation is announcing the addition of the ability to communicate via 3780 protocol to the COMBOARD™ series of front-end processors for DEC VAX and PDP-11 systems.

The COMBOARD/3780 is a complete hardware/software system designed to function under the VMS, RXS-111M/M+ and UNIX operating systems. It provides an RJE link to IBM or any other mainframe supporting the standard 3780 protocol. It performs the communications processing at speeds up to 56 kbps. The off-loading of this processing from the DEC CPU allows the DEC machines to respond rapidly, processing user applications more efficiently.

The COMBOARD/3780 offers the ability to route files directly from the central system to end-user disk areas. The end-user is automatically notified

when files are placed. This saves the users time and provides security for the received data. Files can be previewed before valuable printing time is used. Graphics can be previewed before being spooled to plotting devices.

The COMBOARD/3780 also has the ability to act as a central system. This feature allows the DEC systems to communicate with PC's or any RJE station able to communicate via 3780 protocol.

End users make more cost effective use of both the central system and DEC machines because of the ease of use and efficiencies of the COMBOARD.

COMBOARD/3780 is available in three models supporting transfer rates from 1200 to 56,000 bps. Prices range from $9,800 to $17,900 including hardware and software. For information contact Software Results Corp., 2887 Silver Drive, Columbus, Ohio 43211 (614/267-2203).

HYBRICON OFFERS UNIBUS, Q-BUS BOARDS OFF-THE-SHELF

Hybricon Corporation offers wire wrappable socket boards "off-the-shelf." These boards are available with socket pins installed (4-format) or for use with DIP sockets (2-format). Hybricon's dual size 2-DE2-VHF can hold 52-16 pin DIPS, and costs only $80.00 in unit quantities. The quad size, 2-DE4B-VHF, can handle 111-16 pin DIPS at a unit cost of $140.70. Loaded with socket pins, the 4-DE2-VHF sells for $198.60 and the 4-DE4B-VHF for $358.30. Hybricon's patented pattern, VHF ground planes and VHF accessories enables you to mix analog and digital logic as well as high speeds (in excess of 250 MHz). These boards are directly compatible with Digital Equipment Corporation's hardware. Extender boards related accessories, wire wrapping services and custom design are all offered by Hybricon Corporation, 410 Great Road, Littleton, Massachusetts 01460, (617) 486-3174.

INMAC HAS COMPACT GENDER CHANGER

A self-contained, compact gender changer available from Inmac makes it easy and economical to reconfigure office terminals by joining cables whose genders conflict.

Inmac offers the gender changers as an inexpensive alternative to replacing or modifying EIA 232 cables. The unit — which will fit into a shirt pocket — consists of two EIA 232 25-pin connectors placed back to back. Both male-to-male and female-to-female connectors are available, with all 25 pins connected.

The gender changers are marketed by the company's Data Communications Division, and are guaranteed for one year. Data Communications also manufactures standard and Clear Signal™ custom cables for a wide variety of micro and minicomputers and peripheral devices.

Inmac offers 24 hour shipment of phone and mail orders from regional sales and distribution centers throughout the United States. The company also maintains distribution facilities in Europe. For a free copy of Inmac's 100-page, four-color catalog listing over 2,000 mini and computer accessories, contact: Ron Becht, Product Marketing Manager, INMAC, Department 118, 2465 Augustine Drive, Santa Clara, CA 95051, 408-727-1970.

SOFTOOL RELEASES CHANGE CONTROL ENVIRONMENT

Softool Corporation is pleased to announce the release of its Change Control (CC) Environment.

CC automates the management of software changes and their documentation. It also controls who can make changes and to what components, thereby eliminating many unworkable manual procedures. CC can handle anything: source code, object code, test data, documents, . . . and it can deal with any language: FORTRAN, COBOL, PASCAL, ADA® JOVIAL, Assembly . . .

CC provides comprehensive features that include: automatic reconstruction of previous versions, problem tracking, difference reports, management reports, access control, archiving, compression, encryption, automatic recovery, and more. CC is an interactive tool with an easy-to-use interface, on-line help, and interactive tutorials.

CC is a stand-alone component of SOFTOOL, which is an integrated set of tools. CC offers complete support for source code management. CC is upgradable to the SOFTOOL Change and Configuration Control (CCC™) Environment which, in addition to change control, also offers full support for configuration control. A Programming Environment is also available.

A permanent license for CC is $12,000 with significant multiple copy discounts and a fifty percent credit toward any subsequent upgrade to CCC. For further details, contact Softool Corporation at 340 South Kellogg Avenue, Goleta, CA 93117. Telephone: (805) 964-0560.

SEMINARS, MEETINGS, SHOWS

PRECISION VISUALS SPONSORS FREE GRAPHICS SEMINARS

Precision Visuals, Inc., is currently sponsoring a free half-day graphics seminar in cities throughout the United States and Canada for end users, programmers, analysts, technical managers, and others who design and implement applications systems employing computer graphics.

Entitled "Understanding and Applying Today's Computer Graphics," the PVI seminar will be given in more than 40 cities this year, and is open to any interested graphics user by simply contacting PVI in advance to reserve a place.

According to Don Van Dyken, PVI's director of marketing, the seminar will emphasize the potential for increased productivity offered by computer graphics applications in the fields of business, industry, government, education, and enginering. Participants will be given demonstrations of such diverse applications as business graphics, computer-aided design, contouring, demographics, 3-D surfaces, and electronics design.

In addition to surveying graphics applications and trends, the seminar will present options available in computer graphics software and hardware, and define the issues a potential buyer must consider when shopping for a graphics package. The advantages of a device-independent package will be explored as part of a discussion of efforts currently underway by standards organizations to establish a device-independent graphics standard.

Precision Visuals, based in Boulder, Colorado, is a leading supplier of graphics software tools offering device, machine, and even application independence. The company's DI-3000 package has been installed at more than 400 sites worldwide.

According to Van Dyken, seating is limited at the seminars, and those interested in attending should make reservations as soon as possible. For dates and locations of the seminars and registration information, contact Robbie Becker at Precision Visuals, 6260 Lookout Road, Boulder, CO 80301, (303) 530-9000.

PEOPLE, PLACES, THINGS

NCCS, EG&H INTEGRATE VSORT

North County Computer Services, Inc. of Escondido, California and Evans Griffiths & Hart, Inc. of Lexington, Massachusetts announced that they have agreed to integrate OEM versions of the software packages VSORT and VSELECT, developed by Evans Griffiths & Hart, into NCCS's new USER-11V database package for Digital Equipment Corporation's VAX computer. USER-11V is the VAX version of USER-11, a high-performance database management system for RSTS/E and CTS-500 operating environments, which is currently in use at over 200 installations. It is expected that the increased sort speed provided by VSORT and the record extraction and reformatting capabilities provided by VSELECT will significantly improve the performance and increase the capabilities of USER-11V's SORT/SELECT module.

The USER-11V database management system is organized in functional modules that are quickly combined and modified to implement many different types of applications. In addition to SORT/SELECT, there are database, report, data entry and modification, transfer, and program generation modules. USER-11V is fully dictionary-driven to simplify the maintenance of applications as they change over time.

VSORT, a very fast sort package for the VAX was released by Evans Griffiths & Hart in November 1982. It sorts fixed-length records from sequential or relative files from three to seven times faster than DEC's SORT-11, and also requires less disk space. VSELECT is a high speed package that extracts and optionally reformats records that meet user-supplied selection criteria. The package was released in March 1983. For further information contact: North County Computer Services, 2235 Meyer Avenue, Escondido, California 92025. Telephone: 619-745-6006, or Evans Griffiths and Hart, Inc., 55 Waltham Street, Lexington, Massachusetts 02173. Telephone: 617-861-0670.

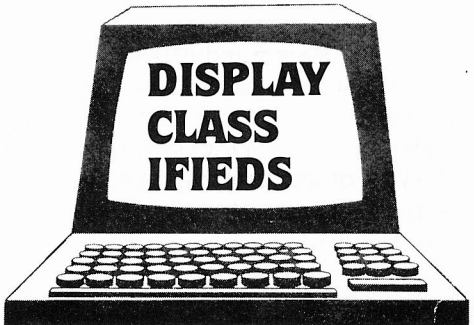
. . . continued on page 84

CLASSIFIED

Send Classified Ads to: RSTS Classified, P.O. Box 361, Ft. Washington, PA 19034-0361.
$1.00 per word, first 12 words free with one year's subscription. [Be sure to include a phone number or address in your message.]

I am leaving the RSTS PROFESSIONAL for a warmer climate in SW New Mexico, and I will seek temp. contract work in the fall of '83. 10+ years of DP, 5 of them in RSTS, mostly Basic+; please ask publisher Dave Mallery if I'm any good. Rob Frazer, Programmer at Large, c/o THE DEC PROFESSIONAL, P.O. Box 114, Springhouse, PA 19477.

Buy, Sell, Trade: DEC Systems, Parts, Peripherals. Call Paul, Digital Computer Exchange, Inc., 27892 Adobe Court, Hayward, CA 94542. (415) 886-8088.



Classified ads are priced at $1.00 per word. Display ads are $35.00 per column inch, plus $1.00 per word. If we set, this includes border and 2 lines in bolder and/or larger type size, if desired, — please specify.

LIST OF ADVERTISERS

. . . continued from page 81

SYSTEM INDUSTRIES AND MASSTOR JOIN FORCES

System Industries and MASSTOR Systems Corporation recently announced an Agreement providing for the joint marketing and development of products.

This Agreement enables System Industries to offer the DEC VAX mini and supermini-computer user storage capacities ranging from 160 megabytes through 440 giga bytes per subsystem. The products will allow large main frames such as CDC, IBM, HONEYWELL and UNIVAC real time access to very large on-line data bases in addition to the DEC VAX systems. MASSTOR's M860 Mass Storage System allows storage capacities ranging from 55 giga bytes to 440 giga bytes per system. The System Industries/MASSTOR local area networking architecture will provide interconnection of up to 64 computers over distances of one mile and longer distances with commercially available telecommunications services. Reliable multimegabit data communcations at speeds up to 50 million bits per second is provided.

MASSTOR designs, produces, markets and services very large-capacity (55 gigabytes or more on-line) data storage systems and high speed (approximately 50 megabits per second) general purpose local area and remotely-coupled data networks.

System Industries designs, produces, markets and services large capacity disk systems and high performance tape systems for DEC and Data General computers. Now in its 14th year, System Industries has over 30,000 data storage systems installed worldwide. The firm is publicly held, traded OTC (NASDAQ symbol — SYSM). For information, contact System Industries, Inc., 408/942-1212.

EMULEX ANNOUNCES AGREEMENT WITH SYSTIME COMPUTERS

Emulex Corporation recently announced that it has reached agreement with Systime Computers Limited (of Leeds, England) to supply selected Emulex products for use in Systime's activity as one of Britain's largest computer manufacturers.

"Systime has agreed to use Emulex's disk and disk/tape controllers and Emulex's communications controllers and multiplexers for its PDP-11 and VAX-11 systems," reports James F. Martin, Emulex's Vice President, International Sales. "The products involved are Emulex's SC21, SC71, SC750, and V-Master/780 disk controllers, TC11 tape controllers, and CS11 communications multiplexers."

"We're delighted that Systime has selected Emulex products after an extensive evaluation process," stated Mr. Martin.

The SC21 is a single-board, fully-embedded, software transparent disk controller that Systime will use in its PDP-11/44 applications. In VAX-11/730 applications, the SC21/V will be used, which consists of a hardware/software package developed and supported by Emulex. The SC71 is, where requested, a fully embedded, software transparent disk controller designed for use with the PDP-11/70.

The SC750 is a single-board, fully-embedded disk controller designed specifically for the VAX-11/750 CMI bus. It allows the user to mix up to four disk drives of varying capacities and media types, while maintaining full software transparency. The SC750 emulates the DEC RH750 Massbus Adapter with RM03, RM05, RM80, and RP06 drives.

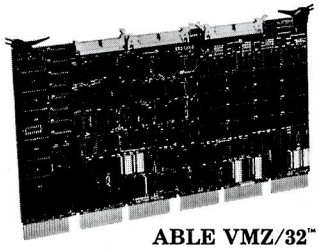
The V-Master/780 is a four-board, fully-embedded disk or disk/tape controller designed specifically for the VAX-11/780 SBI bus. While maintaining full software transparency, it allows the user to mix up to eight disk drives or four disk drives and four tape drives — all of varying capacities and media types.

Emulex, headquartered in Costa Mesa, California, designs, manufactures, and markets disk and tape controllers, communications controllers and multiplexers, and mass storage peripheral subsystems for mini- and micro- computers manufactured by Digital Equipment Corporation. ♥

If you're in the market for communications modules, make the ABLE connection now. And join the thousands who already have.

We are known as the innovators. Most of our products are industry "firsts" which become popular quickly, then settle into a stage of steady long-term acceptance. These four DEC-compatible, communications devices fit the pattern perfectly. They are ABLE originals. They achieved instant success worldwide. They provide top performance. And they are very reliable. Read on to find the one for you.

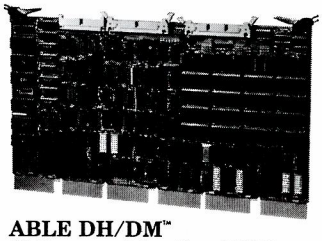
INCREASED VAX THROUGHPUT.



ABLE VMZ/32™
16-line DMF/32 subset

Here's an asynchronous microcontroller with programmable DMA, fully transparent to VAX/VMS as two 8-line DMF 32's and contained on a single board. Priced below the DZ11-E, it outperforms DZ or DH devices under VMS v.3, has interrupt-driven modem control on every line, and includes an output throttle which lets peripheral devices optimize their own data rate.

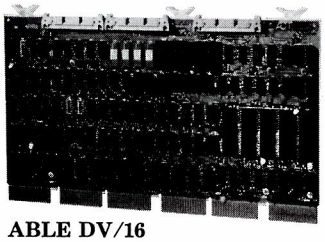
#1 UNIBUS DMA.



ABLE DH/DM™
16-line combination DH11
& DM11 replacement

Then there's our DH/DM, the original multiplexer which puts 16 lines with modem control on a single board. This popular device meets UNIX VAX system needs for DMA communications requirements, serves UNIBUS systems equally well, and beats them all for MTBF, throughput and price. Other features include on-board diagnostics, modem control on all lines, superior on-board silo depth and variable prom-set.

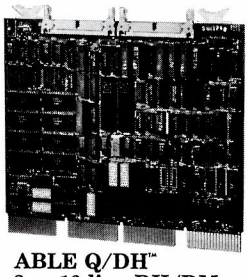
SYNC/ASYNC FLEXIBILITY.



ABLE DV/16
16-line DV11 replacement

A controller for the PDP-11 user, the DV/16 contributes microprocessor-derived flexibility, which permits mixing of sync and async lines in combinations of 4 or 8 lines with modem control and full system software compatibility. It takes less than half the space of a DV11 and uses word transfer instead of byte DMA to gain a 2 to 1 speed advantage or permit operation in half the bandwidth required for data transfers.

Q-BUS DMA.



ABLE Q/DH™
8 or 16-line DH/DM
for Q-BUS

The Q/DH is an asynchronous controller which makes DH-class performance possible on PDP-11/23 and LSI-11/23 Q-BUS systems. It connects the standard Q-BUS to as many as 16 async lines with DMA output capabilities and allows optimum Q-BUS utilization. Features include software compatibility with RSTS/E and RSX operating systems, large input silo, modem control on all lines.

Write for details on our complete line of DEC-compatible products. Be on the lookout for exciting new ABLE communications products soon to come.

For Immediate, Toll-Free Information, Dial 800 332 ABLE.

ABLE COMPUTER

CORPORATE OFFICES
ABLE COMPUTER
1732 Reynolds Avenue
Irvine, CA 92714 • (714) 979-7030

NATIONAL OFFICES
Burlington, MA (617) 272-1330
Irvine, CA (714) 979-7030

INTERNATIONAL OFFICES
Canada (Toronto) (416) 270-8086
England (Newbury) (0635) 32125
W. Germany (Munich) 089/463080

IB Graph for DEC users. The most cost-effective way to get in touch with your data.

Meet the latest Used Software package from Data Processing Design, Inc. It's called IB Graph.™ Nothing less than the most complete, most cost-effective system for business graphics on Digital PDP-11™ and VAX™ processors.

Of course, you could buy more expensive graphics software, but you'd probably end up with more capabilities than you'd actually need—which wouldn't be cost-effective. Or you could buy *less* expensive graphics software, that simply can't do the job. Again, not very cost-effective. IB Graph is the ideal compromise, giving you literally everything you need, and no more.

IB Graph is a multi-user, interactive businss graphics system that lets you generate graphs in minutes instead of days. And get quality, full-color bar charts, line charts, or pie charts for immediate publication or presentation. With IB Graph, you can better visualize and analyze your data. You can make quicker decisions and increase your efficiency—not to mention your company's profits. Best of all, IB Graph is easy to learn and simple to use, whether you have no experience in computers at all, or are an experienced programmer.

IB Graph outputs to a variety of graphic CRTs and plotting devices. And it will continue to be compatible with future hardware announcements by DEC.

If you'd like more information, call or write to us at DPD. We'll be happy to tell you more about IB Graph.

IB Graph. You can buy a more expensive system. Or a less effective one.

You can't buy a better one.

IB Graph

dpd Data Processing Design, Inc.
AUTHORIZED digital® COMPUTER DISTRIBUTOR

CORPORATE OFFICE • 181 W. Orangethorpe, Suite F • Placentia, CA 92670 • 714-993-4160 Telex 182-278

CIRCLE 59 ON READER CARD