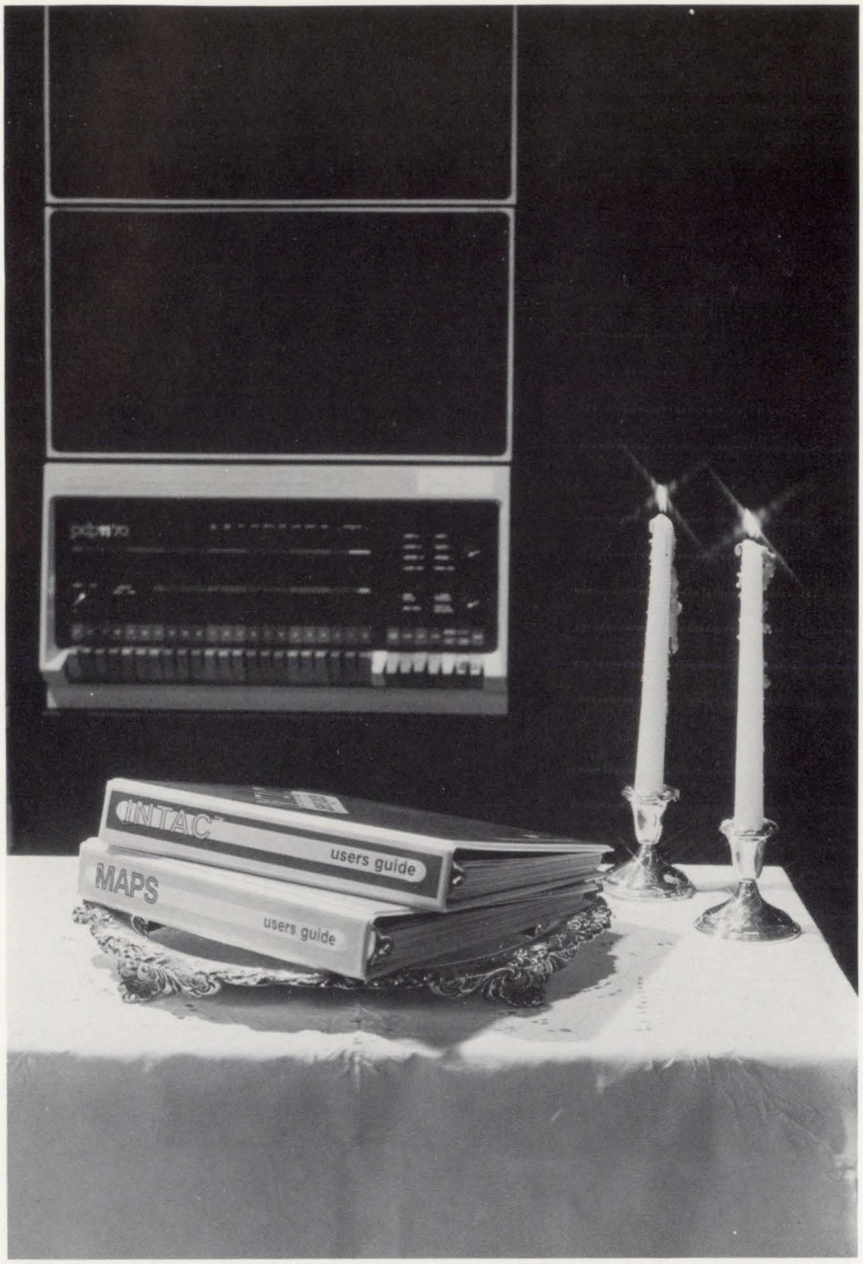
RSTS PROFESSIONAL

Volume 3, Number 3

September 1981
$10.00/issue, $25.00/year

INSIDE:



Two Distinguished Products for PDP-11... And now VAX Users

INTAC™
MAPS™

Interactive Data Base Management

INTAC is a new concept for data storage and retrieval that features an easy-to-use question and answer format, built-in edit rules, multi-key ISAM data access, interactive inquiry and a unique report generator.

Financial Modeling

MAPS, recognized worldwide for over five years as a leader in financial modeling and reporting, is used to construct budgets, financial forecasts, consolidations and "what if" analyses.

Ross Systems, with over seven years of proven capability, now offers these two products to current and prospective PDP-11 and VAX users. INTAC and MAPS enable business managers to produce instant reports themselves, and relieve DP managers from the pressures of special requests.

Ross Systems offers these management tools on our timesharing service, for license on existing computers and as part of a complete, in-house timesharing installation.

Call us collect for more information.

ross systems
incorporated

1900 Embarcadero Road, Suite 208, Palo Alto, CA 94303 • (415) 856-1100 • Other offices in San Francisco, Los Angeles and Dallas

CIRCLE 1 ON READER CARD

Reliable, Economical Printer Systems At 200, 300, 600 & 900 lpm.

For DEC, Data General, Hewlett Packard, Texas Instruments and most other popular computer systems.

☑ **Low Cost.**
Initial low purchase price, plus the long-term savings resulting from inherent reliability.

☑ **High Performance.**
At 200, 300, 600 and 900 lines per minute.

☑ **Quiet.**
Set new industry standards for impact technology.

☑ **Reliable.**
Built for heavy-duty applications.

☑ **Superior Print Quality.**
Achieved through fully formed steel band technology.

☑ **Easy-to-maintain.**
Latest microprocessor technology and diagnostics make these printers the products for the '80s.

☑ **30-Day Delivery.**

☑ **The Industry Leader At Your Service.**

☑ **Guaranteed Compatibility.**
SSI's interfacing technology guarantees compatibility with DEC, DG, HP, TI, Perkin Elmer and most other processors.

☑ **Parallel or Serial.**
SSI's pioneering approach in interface technology allows you the choice of parallel or serial operation. In serial interfacing, SSI also offers both asynchronous and the synchronous needed for IBM and Burroughs Poll and Select, plus others.

☑ **Nationwide Service.**
Unmatched quality of service through SSI's innovative central dispatch control system.

☑ **Band-Change Flexibility.**
Many different bands provide many different typefaces with a single change by the operator. Special bands available also.

I'd like to know more about the following printer system(s):

_ 200 lpm impact matrix
_ The B series (300 or 600 lpm band)
_ High speed 900 lpm band
_ The 2200 series (300, 600, 900 lpm drum)
_ The 2550 (1500 lpm charaband)
_ QT series (600, 1000, 1200 lpm heavy duty band printers)
_ Serial Interfacing (synchronous or asynchronous)
_ Parallel Interfacing

My computer is a________________
____

My requirements are: __Immediate
__3-6 months __Information Only

Name________________________
Title________________________
Company______________________
Address______________________
City_________________________
State________________Zip______
Telephone_________________CD581

SSI
Southern Systems
The Printer System Problem-Solvers
2841 Cypress Creek Road
Fort Lauderdale, Florida 33309
(305) 979-1000; (800) 327-5602
Telex 522135

CIRCLE 2 ON READER CARD

AMBASE PUTS YOU OUT IN FRONT

Everyday we find the pressure to produce is being pushed on the programming staffs around the world. Programmer productivity is the name of the game.

Application Development is a time consuming process, and without the proper tools, even the best programmers can get lost in the shadows.

Amcor has the solution for the productivity problem. . . AMBASE, our revolutionary Application Development System and DBMS. AMBASE is designed to work with DEC PDP-11 computers utilizing the RSTS/E operating system.

AMBASE is helping technical staffs worldwide to improve their productivity from 100-900%. Jobs that took 10 days are being completed in 1 day with AMBASE.

So don't get lost in the shadows, let AMBASE put you out in front.

For more information on AMBASE, or any of AMCOR's complete line of data base oriented applications, give us a call, or use the coupon provided below.

Software for DEC RSTS/E Systems

Please forward information on the following systems:

amcor

1900 PLANTSIDE DRIVE
LOUISVILLE, KY. 40299
502/491-9820

□ AMBASE
□ Accounts Receivable
□ Accounts Payable
□ GL/Financial Mgt. (AMFACS)
□ Payroll
□ Order Processing
□ Inventory Control
□ Sales Analysis

Name____________________

Company ____________________

Address____________________

City________________State____Zip______

Telephone ____________________

Computer Type__________Operating System__________

CIRCLE 3 ON READER CARD

*DEC is a TRADEMARK of Digital Equipment Corp

Contents

COVER: The Remote Diagnostic Center in the foothills of Colorado's Rocky Mountains.

Coming . . .

2nd Anniversary Issue

- Prof. Al Cini
- Directory Anatomy by Scott Banks
- Report from UK RSTS Sig.
- The Vax Scene/ A RSTS to Vax Conversion
- More RSTS Internals by Mike Mayfield
- Macro Man Returns
- Using Sort 11 from Basic + 2
- New 11/40 Continued
- VT100 Functionality
- Software Protection
- Dungeon Map
- Benchmarking Cont.
- More . . .

The RSTS Professional Magazine, September 1, 1981, Vol. 3, No. 3. Published quarterly. Single copy price $10^00, $25^00 per year. Published by M Systems, Inc. 753 Johns Lane, Ambler, Pa. 19002-0245, telephone (215) 542-7008. Send all correspondence and change of address to: Box 361, Ft. Washington, Pa. 19034-0361. Copyright © 1981 by M Systems, Inc. No part of this publication may be reproduced in any form without written permission from the publisher.

From the editors. . .

Dave Mallery

Yesterday I spent about 12 hours trying to make a megabyte 11/40. No one has ever made one before — in fact, it probably never entered the original architect's wildest dreams. I was working with several of the finest engineers I have ever met in an atmosphere of 'lets roll up our sleeves and make it go.' It didn't go, but it will.

The scene in that computer room was one of the high points of my career. I tend to wax philosophical at times, and this was one of them. Here we were, one of the senior design engineers in the industry, a RSTS internals man of the highest caliber, a regional sales manager whose hardware and software expertise overshadows most 'gurus' and yours truly using logic analysers, home made diagnostics and raw intuition to shoot a wierd glitch in the interaction of RSTS as it is initialising itself with the widest selection of foreign peripherals you ever saw - all to make a ten year old design cpu see a megabyte of 64k chips. There's more logic on any board in the backplanes of that machine than the entire cpu, yet that old cpu is such a supurb engine that all the effort is worth while.

What is it that justifies all that hi-priced effort on a ten year old PDP-11? Simple — it runs RSTS.

Last month, I spent a day wandering through two floors of the New York Colesseum for the COMDEX show. That is a show for micro-computer manufacturers and distributors. I saw about fifty variations of the same machine — an 80nn processor with a 10 or 20 MB winchester running endless variations of cp/m based 'operating systems'. Everyone was telling me how they were going to have UNIX real soon and that would be the promised land. The trouble with all these systems is that the software (operating systems) for them is TERRIBLE (You think VAX has it bad!). Then, in a corner of the exhibits, without much fanfare and hoopla, I saw the giant killer. A box containing an 11/23 and a winchester, much like the others, except for one big difference. If you type ctrl/c, it says 'Ready'.

All DEC has to do to sweep that entire industry into the trash heap is to make a realistic oem RSTS license agreement for 23's. The hardware is priced right... how about the software???

THANK YOU

Carl Marbach

In New Orleans in 1979 Dave and I attended the DECUS meetings at the OMNI hotel. The RSTS SIG was in an uproar over funding for the SIG and its activities. DEC, they said, wanted to be less involved in paying for things like our newsletter; we were going to have to pay for it ourselves. As a contributor to that infrequent and too small document I was concerned: I looked forward to its arrival even if it was only twice a year or so (there seemed to be no real schedule). Frankly, it didn't seem worthy of a price but, I thought, we could really have a dynamite newsletter! Paid authors, color, professional typesetting, advertising, regular columns, and a real schedule we could count on . . . this wasn't a newsletter it was . . .

I turned to Dave and said out loud for the first time, "what we need is a MAGAZINE". Without being too dramatic, the RSTS PROFESSIONAL was born in that meeting room. Can we? Could we? Will we? We did! But not without lots of help. We are indebted to our authors for their knowledge and their perseverance in turning it into the written word for all of us to read. We are indebted to our advertisers who went out on a limb with us, especially in the first issue when we didn't even exist in the real world. But we are most indebted to our readers: YOU. Without a readership that has continued to grow beyond our expectations (we print 7000 issues of what you are now reading), without the constant feedback we need to make this a better magazine, without your contributions of articles, and without your financial support we could not be where we are today. Even today, our "tell a friend" and the bounty hunter program are our best source of new subscriptions and sale of the classic back issues.

RSTS is indeed something special. What other operating system could have a Magazine like the RSTS PROFESSIONAL. People are amazed to find a magazine directed toward one type of computer usage, but RSTS has blazed the way for real "user friendly", "full featured", "robust", and functional operating systems. It is the standard that VMS will be measured against, the goal that the RT11 people will shoot for (and make), and a classic in the Computer Hall of Fame. You did it! Never forget, it is people who make things great, and it is all the great applications that are out there that have made RSTS what it is.

We are saying "thank you". Not only in words but in deeds as well. The wealth of material we have been getting is growing all the time, and we hate to hold any of it up for even one issue, but we can't fit any more in the size book we are putting together now. Therefore we offer the following two "thank you's":

First: Our December issue will be our gala second anniversary issue and will have 116 pages instead of our normal 100.

Second: Beginning in 1982 the RSTS PROFESSIONAL will be coming to you every other month, six times a year! As RSTS grows into the 80's we need the extra coverage to keep all the good things coming. So beginning in 1982, you'll be able to see us in February, April, June, August, October and December. — Thank you.

RSTS PROFESSIONAL

Editors
R.D. Mallery
Carl B. Marbach

Assistant Editor/Advertising
Helen Marbach

Controller
Peg Leiby

Administrative Assistant
Hope Makransky

Subscription Fulfillment
Kathi B. Campione

United Kingdom Representative
Pauline Noakes
RTZ Computer Services Ltd.
P.O. Box 19, 1 Redcliff Street
Bristol, BS997JS
Phone: Bristol 24181

Contributors
Scott Banks
Rudy Bazelmans
Mark Diebert
Steven L. Edwards
Jeffrey R. Harrow
Stephen F. Heffner
Kevin Paul Herbert
Paul R. Laba
Richard A. Marino
Mike Mayfield
Gary W. Miller
Ross Miller
David Spencer
David Whip

Cartoons
Douglas Benoit

Photographic Consultant
Bill Marbach

Design & Production
Grossman Graphics

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.)

*This publication is not promoted, not authorized, and is not in any way affiliated with Digital Equipment Corporation. Material presented in this publication in no way reflects specifications or policies of Digital Equipment Corporation. All materials presented are believed accurate, but we cannot assume responsibility for their accuracy or application.

Remember when DEC controller manufacturers discouraged plant tours?

Every business starts with a big idea and a small shop. Emulex was no different. The simple reason we outpaced the industry was our dedication to fundamentals.

Like design. Even in the beginning Emulex customers didn't have to tolerate compromises. Then, as now, Emulex designed controllers that actually out-performed their counterparts.

We also concentrated on product quality. Conventional reliability standards weren't good enough for Emulex (they still aren't). We use only pre-tested/aged parts (MIL STD 883B); dynamic 96-hour environmental temperature cycles; and four complete and distinct levels of testing and quality control certification.

Emulex didn't skimp on support. Application experts help before the sale, technical support and customer service after the sale. Nationwide maintenance is available through Control Data and General Electric.

Concentrating on fundamentals paid off. Today Emulex has the industry's best and broadest line of disk, tape and communications controllers for DEC PDP-11, LSI-11, PDP-11/70, and VAX-11 CPUs.

We admit that being big is no reason to work with Emulex. But the fundamental reasons why we're the biggest certainly are.

There *is* a real difference between an imitation DEC controller and a genuine Emulex. Come see for yourself.

Emulex. The genuine alternative.

EMULEX

Emulex Corporation
2001 East Deere Avenue
Santa Ana, CA 92705
(714) 557-7580
TWX: 510-595-2521

CIRCLE 58 ON READER CARD

LETTERS to the RSTS Pro...

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

Dear Mr. Marino:

First, I would like to praise your new column in **RSTS Professional** on benchmarks. I think it gives much needed information and am looking forward to future issues. There are a couple of questions I would like to see answered in your column.

I am wondering what kind of performance increases I can realistically expect by adding cache (KK11-A) to an 11/34A that is running RSTS/E v 7.0. DEC says in their literature that a typical program 'could' execute in 40% less time. Their 40% figure comes from using 80% hit ratio with 80% memory reads. Are these figures applicable to a RSTS/E site running application software written in BASIC-plus? Also, what kind of performance can we get from larger foreign caches that hang on the UNIBUS such as ABLE CACHE/434?

These questions stem from the fact that our 34 is beginning to feel the effects of running approximately 25 jobs with about 6 to 8 active at any one time. I am wondering if cache would not be a way of getting a good return on investment.

Keep up the good work. Sincerely,
Ed Horst, Elkhart, IN

RICHARD???

—

Dear Sir,

$REORDR - Sorting Alphabetically

Your readers may be interested in this slight variation to $REORDR which allows it additionally to sort the directory in alphabetic order as well as the normal date and time sorts. This can be useful if you want to look for a particular filename in a directory. I have found it helpful for program development directories where one tends to have a mass of files and date order is not so interesting, but names are. The attached source file $REORDR. APP should be appended to REORDR.BAS and then compiled to $REORDR.BAC <124> or whatever it is called in a particular system. Yours faithfully,

G.T.A. May, Special Systems Division
Software Sciences Limited, UK

P.S. I've attached an installation process and demonstration run. You've got to follow the installation process else RSTS Basic Plus runtime system screws you up because I've redefined function FNI%.

George, thank you. "$REORDR - Sorting Alphabetically" is scheduled for the December 1981 issue.

—

Dear RSTS Pro,

At the Spring Symposium I got the impression that few people know of the dangers and loopholes of an unsecure RSTS system. There are many ways of getting into a system that has not had the relevant safeguards built in. I have been in RSTS for six years and below are a few that I have come across.

1. 1,2 password SYSLIB.
2. 1,1 password same as pack id.
3. Control C to 'Command file name?' prompt on start up.
4. Temporary access to a privileged account, copy UTILITY.SAV <232> to your own account, then rename; change PWs of privileged account when access wanted. Hence check that no dangerous cusps have protection code <232>, even in (1,2).
5. If KB0 (or any KB used by a privileged user) is not set to PRiveleged in SET sub option, then a non privileged user can OPEN 'KB0:' and mimic LOGIN, hence receiving a privileged password as input. He could then login to the privileged account and run a pre-compiled program in his account to log the unsuspecting user in.

Graham Dare, Systems Support Manager
Ladbroke Hotels, Holidays & Taverns
Watford, England

Thanks, Graham, now not only do we know the dangers — we also know the loopholes.

—

Dear RSTS Pro:

First, I wanted to let you know that I have been enjoying your magazine a great deal. It's really fantastic to have a magazine devoted to RSTS!! Secondly, I wanted to thank the folks at Software Techniques for their contributions. I did, however, find a minor problem with the Mini-debugger (DEB, a Basic-Plus 2 Mini-debugger, RSTS Professional, Vol. 3, #2). If your compiler includes the patch to make $CODE and $PDATA read-only psects (patch seq. 45.2.12) the goto and resume functions will not work. This is because the routine depends on psect placement to determine some addresses. The cure is simply to make the $CODD and $CODF psects in the debugger read-only psects (ie: .psect $CODD,RO).

Again, thanks for a great magazine.

Dave Patterson, Programer/Analyst
Sivalls, Inc., Odessa, Texas

You're welcome, Dave. And, a "thank you" to you also — our readers help make it possible.

—

Dear RSTS:

I am very thankful for the many useful articles in the *RSTS PROFESSIONAL* dealing with disk optimization and monitor internals. I have been making good use of the information found in some of the articles in your March '81 issue, but I have found some problems and inaccuracies I'd like to convey:

On "Disk Structure Notes" by Dave Mallery:

Point #4 you said "Dump the SATT". This is disastrous, I think, for a RSTS pack. Could you explain more?

Point #10, you said creating a non-deletable zero length file would save the contiguity of a UFD. Upon close inspection, I've found that zeroing such an account will reduce the size of the UFD to one cluster — hence zeroing a UFD will break it up in any case.

On "All Things BRIGHT . . ."

This program was very helpful. I used what was in it to write another program, too.

The only problem I found was at line #9040 — if "ROOT%" is negative, the error "Interger Overflow" would occur. ROOT% would have to be an F.P. variable to work for such large disks.

Enclosed is a listing of a program we use to create accounts (UFD's). It will create an extended, contiguous account, on line, placing it near the center of the pack. Please feel free to publish if you wish. Thank you.

Sincerely,
James Swanson, Systems Analyst
Area Two Educational Computer Center

Thank you, James. Readers can find your program "CREATE.BAS" on page 26 of this issue.

Response from D. Mallery: Point 4 - "I meant, inspect the satt to see how far towards the center we have progressed (the SATT algorithm assigns from lo to hi end)."

—

Hopefully you're planning to publish the VMS Professional soon. Please advise date of charter issue and put us on waiting list. Stanley J. Swikart, Dublin, Ca.

Will do, Stanley.

—

Gentlemen,

Is this a record?

```
UT LIST CCL
SÉTOST-=[1,2]NET.TSK;PRIV 30000
NETRST-=[1,2]NET.TSK;PRIV 30000
DEC-ATH=[81,0]DECATH.BAC;PRIV 32000
ATT-ACH=[1,2]LOGIN.*;PRIV 30000
ATP-K=[1,2]ATPK.*;PRIV 30000
@-=[1,2]ATPK.*;PRIV 30000
BAC-KUP=[1,5]BACKUP.*;PRIV 30000
RES-TOR=[1,5]BACKUP.*;PRIV 30000
BCK-=[1,2]RMSBCK.TSK;0
BP2-IC2=[1,2]BASIC2.TSK;30000
BPC-REF=[1,2]BPCREF.*;30000
BYE-=[1,2]LOGOUT.*;PRIV 0
CNV-=[1,2]RMSCNV.TSK;0
COB-OL=SY:[1,2]COBOL.TSK;0
MRG-=SY:[1,2]MRG.TSK;0
RFM-=SY:[1,2]RFM.TSK;0
CRE-ATE=[1,2]EDIT.*;30000
DBUC-=[1,2]DBUILD.*;0
DBUD-=[1,2]DBUILR.*;0
DEF-=[1,2]RMSDEF.TSK;0
DFN-=[1,2]RMSDFN.TSK;0
DES-=[1,2]RMSDES.TSK;0
DIS-MOUNT=[1,2]UMOUNT.*;PRIV 30000
DI-RECTORY=[1,2]DIRECT.*;PRIV 30000
DSP-=[1,2]RMSDSP.TSK;0
DTR-=[1,2]DTR.TSK;0
EDT-=[1,2]EDT.TSK;0
ED-IT=[1,2]EDIT.*;30000
FOR-M=[1,2]INDSHW.TSK;30000
F4-RAN=[1,2]FORTRA.SAV;0
FIT-=[1,2]FIT.*;PRIV 30000
FOCC-=[1,2]FOCOMP.*;0
FOCR-=[1,2]FOCOMR.*;0
SHWIND-=[1,2]INDSHW.TSK;30000
FMP-=[1,2]INDFMP.TSK;PRIV 30000
HELLO-=[1,2]LOGIN.*;PRIV 0
HELP-=[1,2]HELP.*;PRIV 30000
IFL-=[1,2]RMSIFL.TSK;0
IND-ENT=[1,2]INDENT.SAV;30000
LBR-=[1,2]LBR.TSK;0
LIBR-=[1,2]LIBR.SAV;8208
LIN-K=[1,2]LINK.SAV;8208
LOG-IN=[1,2]LOGIN.*;PRIV 0
LOOK-=[1,2]PIP.SAV;PRIV 30000
MACR-O=[1,2]MACRO.SAV;8216
MAC-=[1,2]MAC.TSK;0
MAK-E=[1,2]TECO.TEC;0
MOU-NT=[1,2]UMOUNT.*;PRIV 30000
MU-NG=[1,2]TECO.TEC;0
PAT-=[1,2]PAT.TSK;0
PIP-=[1,2]PIP.SAV;8208
PL-EASE=[1,2]PLEASE.*;PRIV 30000
QU-EUE=[1,2]QUE.*;PRIV 30000
RMC-=[1,2]RMCOPY.BAC;PRIV 30000
RPG-=[1,2]RPG.TSK;PRIV 30000
ASN-=[1,2]RPGASN.TSK;PRIV 30000
ESP-=[1,2]RPGESP.TSK;PRIV 30000
RST-=[1,2]RMSRST.TSK;PRIV 0
TTY-SET=[1,2]TTYSET.*;PRIV 30000
SRT-=[1,2]SORT.TSK;0
SU-BMIT=[1,2]QUE.*;PRIV 30000
SW-ITCH=[1,2]SWITCH.*;PRIV 30000
SY-STAT=[1,2]SYSTAT.*;PRIV 30000
TE-CO-[1,2]TECO.TEC;0
TKB-=[1,2]TKB.TSK;0
TY-PE=[1,2]TYPE.TEC;8
UT-ILTY=[1,2]UTILTY.*;30000
VT50-=[1,2]VT50PY.*;0
VT52-=[1,2]VT52PY.*;0
COMTKB-=[1,2]COMTKB.BAC;0
CLAS-=DB2:[103,0]DASCTL.TSK;0
FRED-=DB2:[104,0]DASCTL.TSK;0
CHAINS-=DB2:[104,0]CHAINS.TSK;0
CLABAT-=DB2:[103,0]CLABAT.TSK;0
SPS-=DB2:[103,0]SPSSEL.TSK;0
SEND-=[1,9]SEND.TSK;PRIV 30000
MAIL-=[1,9]MAIL.TSK;PRIV 30000
NIP-=[1,2]NIP.*;PRIV 30000
TLK-=[1,2]TLK.*;PRIV 30000
NET-=[1,2]NET.*;PRIV 30000
NCP-=[1,2]NCP.*;30000
NFT-=[1,2]NFT.TSK;0
VT-EDIT=[1,2]VTEDIT.TEC;PRIV 0

READY
```

Anonymous

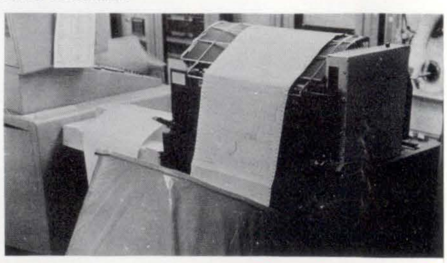
If it's not it ought to be! See "CALLER.BAS" in the December 1981 issue and eliminate all this and save 98 small buffers. Did you enjoy seeing "the picture" last issue?

DO YOU REMEMBER THIS?

(Photo contest, RSTS Professional, Vol. 3, #2, p. 93.)

Photo contests appear in the RSTS Professional occasionally and readers have until publication of the next issue to submit their answers. We may, from time to time, limit the number of correct answers eligible to receive prizes.

Because we have not received a correct answer — this contest continues.



. . . continued on page 95

Better than Datatrieve!

- Easier to understand
- Much more powerful
- Better manuals
- Better support

Not Limited to RMS-11!

- Can access most file types and DBMS
- Include your own file routines
- Mix file types in a single report

THE

ULTIMATE

REPORT WRITER

GRS

GENERALIZED REPORTING SYSTEM
QUERY LANGUAGE and REPORT WRITER

for PDP-11/RSTS systems

from

etc ENTERPRISE
TECHNOLOGY
CORPORATION

Call or write for complete details:

305 Madison Avenue • New York, NY 10165
(212) 972-1860

Incredibly Flexible!

- Complete formatting flexibility
- Powerful built-in calculations
- User-specified calculations
- Include your own special routines

Easy to use!

- Usable by non-programmers
- English query-language
- Very user-friendly
- Nonprocedural language

CIRCLE 4 ON READER CARD

The RSTS/E Benchmarks Part II

By Richard A. Marino, Data Processing Design, Inc.

In the last article I briefly defined benchmarking. Now we will look at one particular type of benchmarking — component benchmarking. This type of benchmarking compares a benchmark environment by measuring a single component. A component may be a program, a job stream, or even a single action. The purpose of a component benchmark is to measure differences in the benchmark environment and from this information determine characteristics of both the environment and the component.

Let's start with an example. One part of building a RSTS/E Version 7 system is the assembly of various files that are part of the monitor. One of these is the assembly of the file TTDVR.MAC (the terminal driver). The lines in the SYSGEN.CTL file which perform this task looks something like this:

```
R MACRO *TTDVR,TTDVR/C=COMMON,KERNEL,CONFIG,CHECK,KBDEF,TTDVR
```

This assembly requires no user interaction and is a good measure of the performance of a cpu and disk since the activity requires both extensive processing and plenty of i/o. For our test we have eliminated the listing part of the assembly and are using the command file in figure 1. If you want to duplicate this component benchmark use this command file and modify a CONFIG.MAC from any previous sysgen using the information in figure 2.

The purpose of this component benchmark is to evaluate the effect of cpu and disk combinations on this assembly. In order to provide the widest possible range of environments, we performed the assembly on both RT-11 and RSTS/E systems. This is very straight-forward since the assembly on RSTS occurs under the RT-11 run time system using a standard version of the RT-11 Macro assembler. The various environments are shown in figure 3. Since in all cases the environment was a single job environment, the amount of memory was not particularly relevant to this benchmark. Note, however, that while the amount of total physical memory does not impact this component benchmark significantly (a fact confirmed independently) this is not always the case, and one should be careful when benchmarking across operating systems.

The results are shown in figures 3 and 4. The elapsed times progressed about as one would expect given the differences in the environments. Some conclusions can be drawn concerning this particular component (the assembly of TTDVR.MAC):

1. An 11/23 with RX02's is faster than an 11/03 with RL01.
2. Using an RX02 in combination with an RL01 is not much better than just using the RX02.
3. An 11/44 is only slightly faster with RK07s than with an RL02.
4. An 11/70 with an RP06 is not much faster than an 11/44 with RK07.

What value are these conclusions? If you were planning for example to upgrade your system and a significant part of your system load were programmers doing assemblies, this type of benchmark might help you make your selection of new hardware. In this case you would certainly consider an 11/23 RL01 based system over an 11/34 RL01 system (less than 10% difference). Further you would probably choose the less expensive 11/44 RL02 system over the RK07, RM02, or 11/70 system.

This particular component benchmark was intentionally limited in its scope. You would need some further work to establish that a multi-user environment would perform the same.

Our second example of a component benchmark deals with much less hardware but a more complicated problem. The screen editing component of two of DPD's products, the WORD-11 word processing system (for documents) and the WAFE editor (for text files) are essentially the same. Each is a full-screen editor with keypad and keyboard functions. Each also is a complete editor not a pipeline editor (like TECO, or EDIT). This means that one can backup to the top of the file at any time. This one feature significantly complicates the editor since it must keep track of text it has passed and how to display it. The display format I should mention is changable throughout the document or text file by using 'rulers'.

Both of these products are available on a wide range of systems (since 1977 on RSTS/E for WORD-11). However the smallest hardware configuration has typically been an 11/34 with some 11/23's just now starting to be used. Written in a portable version of MACRO-11, the software itself is transportable, the question we needed to answer was what would the performance be on very small systems.

We converted the editor to run under RT-11 Version 4 and then set out to measure its performance on four very small benchmark environments:

1. 11/03, dual RX02
2. 11/23, dual RX02
3. 11/03, RL01 (equivalent)
4. 11/23, RL01 (equivalent)

In contrast to the example of the macro assembly, the editor is an interactive program and has many different features and operations. We decided to measure several typical operations:

1. Entering the Editor (Startup)
2. Display one screen (Move and display 7000 characters)
3. Move 10 pages (Move 35000 characters, display 7000)
4. Finish the Edit (Write the output file)

RSTS users!

Oregon Software Sort-1-Plus

“… I would like to single out two of the many products that I have personally bought and paid for. The first is OMSI Sort Version 1.6. I have used OMSI Sort since its original release as a macro replacement for SQWIK and MQWIK, the old DEC type-1 header sort. It was the best software buy I ever made, yielding staggering reductions in run time. Version 1.6 contains the final additions to a fine product – the macro rendition of XQWIK and OQWIK, the extract and reordering portions of the sort. Amazingly fast and truly flexible. Bravo!”

Software Techniques DISKIT

“The second entrant to my ‘Hall of Fame’ is a package from Software Techniques called DISKIT.
… Using DISKIT, I created 130 accounts and fully extended their centred UFD’s in three minutes and forty seconds (a job that used to take four to eight hours). I then copied the full contents of a 300 Mb RM05 equivalent to this new ‘well structured’ disk in 45 minutes, optimising clustersize and contiguity in the process. I (for once) was speechless… The documentation is excellent. In short, this package is the ‘final solution’ to structured disks, eliminating all of the time and complexity and reducing the job to one of the simplicity of a SAVRES.”

Editor Dave Mallery
reprinted from March 1981
RSTS Professional magazine

Recommended by RSTS Professional Now available from RTP.

* Is a complete replacement for $SORT – no other software required
* Sorts in excess of 32767 records
* Handles records across block boundaries
* Up to 50 times faster than $SORT
* Costs £600-00

* **DSU** A disk backup utility for creating optimum structures.
* **DIR** A complete replacement for $DIRECT that runs twelve times faster
* **OPEN** An ‘open files’ display program (‘control F’)
* **LIBRARY** For BASIC PLUS 2 users a library of disk handling routines
* Costs £1250-00

Call Ed Patient or Steve Collins

RTP
real time products

1 Paul Street, London EC2A 4JJ. Telephone 01-588 0667

OMSI, Sort-1-Plus are trademarks of Oregon Software, Portland, Oregon. DEC, RSTS are trademarks of Digital Equipment Corp., Maynard, Mass. DISKIT, DSU are trademarks of Software Techniques, Inc., Los Alamitos, Ca.

CIRCLE 5 ON READER CARD

The results shown in figures 5 and 6, were interesting. Some conclusions are:

1. Invoking a program is, understandably, very disk speed dependent. The size of the editor (120 blocks) makes its startup time on floppy disks very slow (23 seconds).
2. Display speed is cpu intensive. More so than expected. Disk speed plays only a minor part in how fast one can move in a document or file by small amounts.
3. Finishing an edit is disk bound.

What do these conclusions mean? Primarily, if one was engaged in editing a few documents, but each edit consumed a long time, a faster processor is better than a faster disk. For frequent, small edits, a faster disk makes a lot of difference.

From this component benchmark we arrived at some answers, but since we were surprised at the cpu intenseness of displaying characters we decided to also look at the cost of printing. For the word processing system the print spooler does not simply copy a file to an output device. It must do much of the complicated work and in fact determines spacing, line wrapping, page breaks, proportional spacing, footnotes, and many other things itself. We did a short and simple component test of the spooler on the same configurations:

1. Start the spooler
2. Scan past 32000 characters without printing
3. Print one page

The results are shown in figures 6 and 7.

Here we were not very surprised. Startup was again very disk bound (the spooler is approximately the same size as the editor). Scanning past characters involves all the work of printing except the actual physical printing (since the spooler must still count lines, etc.) and this time was comparable to that of the editor, given that the editor must do extra work to enable the user to backup in a document. Printing was as expected somewhat cpu intensive. Except for program startup, printing is essentially cpu bound.

This implies several things. First, on a small floppy disk system, it is much better to print one document of 50 pages than 5 of ten pages, but if you are going to print 5 of ten pages, try to print them all at once and only invoke the spooler once. Finally, although the spooler is cpu intensive, by itself on an 11/03 it probably can still drive a 1200 baud printer continuously. Only printing to a fast printer, multiple devices or providing simultaneous editing and printing would the faster processor really help.

These examples of component benchmarks demonstrate what simple elapsed time benchmarks can do if they are well planned and well executed. Often times it is not the physical activity of benchmarking that is difficult but the planning and the interpretation of the results. Hopefully these examples have given you some ideas for benchmarks on your system.

Component benchmarks though are only part of the picture. In future articles we will examine program instrumentation, a method to measure specific program performance, operating system options and how they affect peformance, combined and system benchmarks which are methods to measure the performance of multiple components, and hardware contributions to performance.

This is a partial listing of the

UNIQUE
RSTS UTILITIES
ENHANCE
DEC® SYSTEMS

CALOUT permits user to dial up a second CPU and exchange data files of any kind

CONTRL provides remote interactive training and user support. Complete log file of session may be kept.

DOC makes 4 logical terminals out of 1 physical terminal. Complete log file of each job session may be kept.

IMAGE allows DOC log file to be played back intelligently, one frame at a time.

MONITR records entire dialog of selected job-session in a secured log file. This is invisible and unintrusive to user being monitored.

LOCK simplifies and generalizes menu creation. Limits user to authorized menu. Records tasks run.

• No installation, simple to use • Clean, certified
• Inexpensive, low overhead

Please send me details on
□ CALOUT □ DOC □ MONITR
□ CONTRL □ IMAGE □ LOCK
Name ____________
Phone ( ) ____________
Company ____________
Address ____________
City ____________
or call Janet (617)275-6642

CLYDE
digital systems
Box 348
Bedford, MA 01730

CIRCLE 6 ON READER CARD

```
; TERMINAL TYPES

KL11      #   1.    ;# OF KL11 TYPE INTERFACES
                     (INCLUDING CONSOLE)

PK11      #   4.    ;# OF PSEUDO-KEYBOARDS

DH11      =   2.    ;# OF DH11 MULTIPLEXERS
DM11BB    =   0     ;1 IF DM11BB'S; ELSE 0
DHONL     =   16.   ;NUMBER OF LINES ENABLED
DH1NL     =   16.   ;NUMBER OF LINES ENABLED

DZ11      =   2.    ;# OF DZ11 MULTIPLEXERS
DZ11AB    =   1     ;1 IF DZ11A/B'S; ELSE 0
DZONL     =   8.    ;NUMBER OF LINES ENABLED
DZ1NL     =   8.    ;NUMBER OF LINES ENABLED

; TERMINAL OPTIONS

MULTTY    =   1     ;1 IF MULTI-TERMINAL OPTION; ELSE 0
EKOCTL    =   1     ;1 IF ECHO CONTROL OPTION; ELSE 0

; DISKS

CNT.DB    =   2.    ;# OF RP04/5/6'S; ELSE 0

; OTHER DEVICES

TM02      =   1.    ;# OF TU16/TE16/TU45/TU77'S; ELSE 0

LP11      =   2.    ;# OF LINE PRINTERS; ELSE 0

; SYSTEM PARAMETERS

JOBMAX    =   48.   ;MAXIMUM # OF JOBS

LOGICL    =   50.   ;# OF SYSTEM WIDE LOGICALS

QSTATS    =   1     ;1 FOR QUEUE STATISTICS
JSTATS    =   1     ;1 FOR JOB STATISTICS
DSTATS    =   1     ;1 FOR DISK STATISTICS
```

FIGURE 1

11/03, RX02 Single Floppy
11/03, RX02, "RL01" Winchester
11/03, "RL01" Winchester

11/23, RX02 Single Floppy
11/23, RX02, "RL01" Winchester
11/23, "RL01" Winchester

11/34, RX01, RL02
11/34, RL02
11/34, RL02 (RSTS/E)

11/44, RL01
11/44, RL02
11/44, RL02 (RSTS/E)
11/44, RK07 (RSTS/E)

11/70, RP06 (RSTS/E)

Unless otherwise indicated, the operating system is RT-11 V4. The "RL01" Winchester is the Data Systems Design DSD 880. The macro assembler from RT-11 Version 4 is being used.

FIGURE 2. Environments

CONFIG.MAC file used in the sample macro assembly component benchmark. It includes only the non-zero values in that file.

Processor	Disk(s)	Elapsed Seconds
11/03	RX02	403
11/03	RX02 + RL01	386
11/03	RL01	262
11/23	RX02	243
11/23	RX02 + RL01	228
11/23	RL01	143
11/34	RX01	278
11/34	RX01 + RL02	250
11/34	RL02	131
11/34 RSTS/E	RL02	140
11/44	RL02	79
11/44 RSTS/E	RL02	83
11/44 RSTS/E	RK07	82
11/70 RSTS/E	RP06	68

FIGURE 3. Macro-11 Assembly (TTDVR)

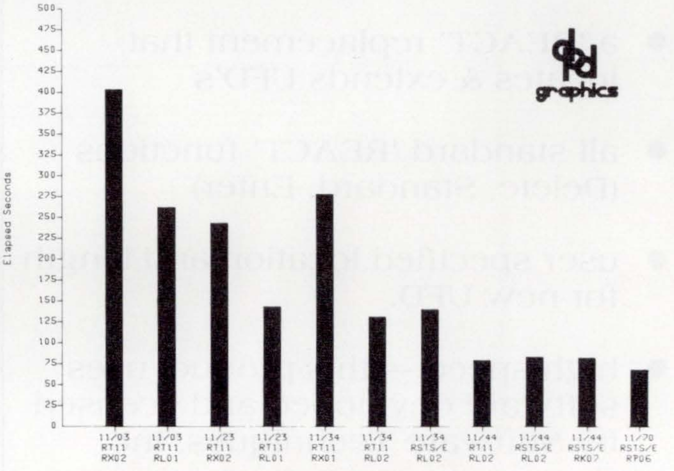


FIGURE 4. Macro-11 Assembly Time (TTDVR)

Editor

	Elapsed Time in Seconds			
Component Activity	11/03 RX02*	11/23 RX02*	11/03 RL01**	11/23 RL01**
Enter Editor (Startup)	24	22	6	5
Move/Display 7000 characters	78	43	71	36
Move 35000 characters	107	71	81	43
File Document (End)	27	25	5	5

*RX02 was Digital Dual RX02 drive on VT103 using both drives.
**RL01 was DSD 880 Winchester RL01/2 equivalent on VT103, single drive.

FIGURE 5. WORD-11/WAFE Component Benchmark

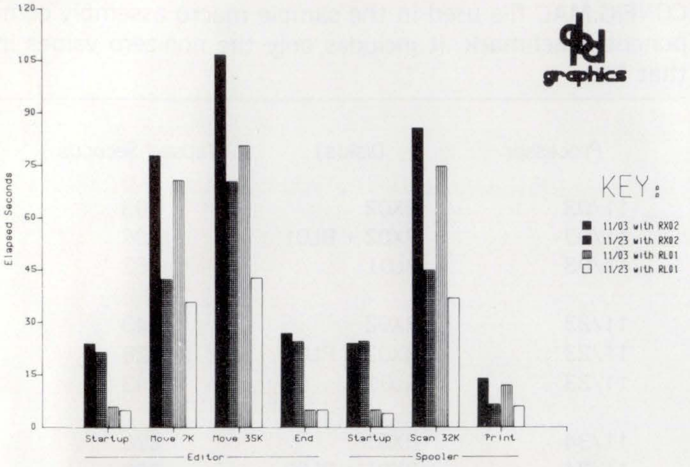


FIGURE 6. WORD-11 Component Benchmark

Structured disks for all!

Announcing

REACT2.TSK

- a 'REACT' replacement that locates & extends UFD's
- all standard 'REACT' functions (Delete, Standard, Enter)
- user specified location and length for new UFD.
- high speed — this product uses software developed and licensed by Software Techniques, Inc.
- distributed as an RSX or BP2 task on 9 track 800/1600 tape

Introductory Price: $200.00
Single CPU License

Nationwide Data Dialog
70 James Way
Southampton, PA 18966
(215) 364-2800

Call For Quick Service

CIRCLE 7 ON READER CARD

Spooler

	Elapsed Time in Seconds			
Component Activity	11/03 RX02*	11/23 RX02*	11/03 RL01**	11/23 RL01**
Enter Spooler (Startup)	24	25	5	4
Scan 32000 characters	86	45	75	37
Print one page	14	7	12	6

*RX02 was Digital Dual RX02 drive on VT103 using both drives.
**RL01 was DSD 880 Winchester RL01/2 equivalent on VT103, single drive.

FIGURE 7. WORD-11 Component Benchmark ♥

MY CHIPS ARE STALE, I WAS UP AND DOWN ALL NIGHT...
32769 %
I THINK I'm GOING TO SLIT MY RSTS
32769%
DB 81.

IT'S AN 11/40 BORN AGAIN

. . . from the editors

What other kind of computer could the RSTS professional have but a RSTS system. But what to get? The menu looks like Howard Johnson's ice cream selection: 11/23, 11/24, 11/34, 11/35, 11/40, 11/44, 11/45, 11/50, 11/55, 11/60, 11/70, Magnum (Able computer) or ?? At Toronto last year the RSTS people had a paper tape based 11/20 running RSTS??

We wanted something small, small enough to be affordable. We wanted something big, big enough to hold lots of data and be used by many users. We wanted something new, new enough to have the new big reliable disks. We wanted something up to date, up to date enough to have all its memory on a single board and be semi-conductor so it would be fast. We wanted something that would be industry compatible, so it could have 800/1600 tape and a single board controller.

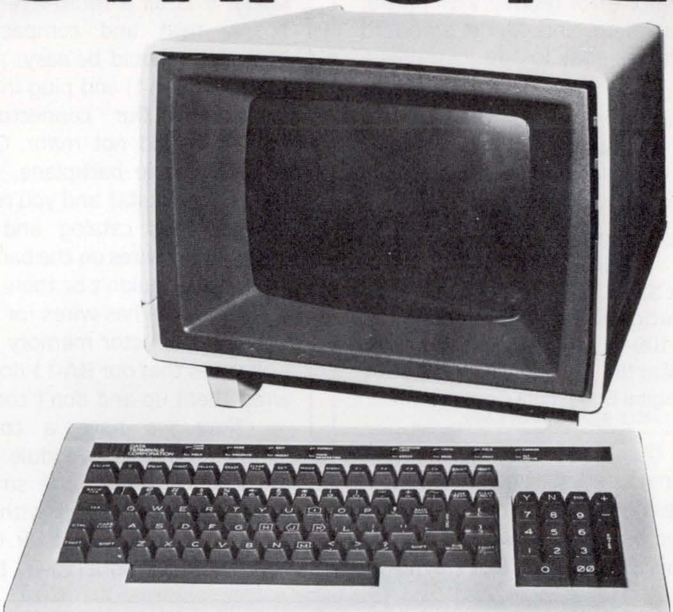
We discovered that all of what we wanted was available new, at lower prices than the old original equipment except for one thing, the processor. It seemed that old processors were readily available at very attractive prices. 11/40s were about half the price of the newer 11/34 yet seemed to offer the same power. The 11/40 looks like an original DEC computer; lots of lights and switches, it sort of has the appearance of an 11/45 or an 11/70. A classic.

We hunted for a while and learned a lot. In order to run RSTS the processor must have: memory management (standard on the 34), a clock, a stack limit register (KJ-11, ours ran without one just fine for some time), the EIS board for the 11/40 (11/34 standard again). You will also require memory, DL-11 console interface, and some type of disk. Of course, in order to connect terminals you will need a multiplexer (DH-11), a tape for backup (TU-10 type) and a line printer for high speed output.

Newman Computer exchange had a bare 11/40 with EIS but no memory management for sale and we bought it. Maybe we're picky, but it seemed awfully hard to deal with them; lots of phone calls never were returned and they seemed more distant that their Michigan location should have been. It was close to 6 weeks from the time we decided to buy that computer and when it was delivered; too long we thought. Newman gives a 30 day warranty and they check it out carefully before it goes out the door and into the moving van. They also included the prints and some manuals with it, but this can vary from machine to machine.

We tested their warranty because when we powered up there was no 5 volt supply to run the processor. It took a $460 DEC service call to repair the faulty regulator. Newman stood behind their machine and paid the bill, no sweat. All in all, they were distant, honest, snappish or short with us sometimes, slow, but in the end delivered what was promised and stood behind what they sold. We would buy from them again if they had the right piece at the right price. Why don't they advertise? Maybe Al Newman knows? If we all tell him, he might get the message.

We now had this wonderful 11/40 with 16KW of DEC memory up and running a toggled in memory diagnostic 24 hours a day. What next? Let me give you a small anatomy lesson on PDP-11's. The processor and all other components plug into a six slot high backplane. Those that use all six slots are called HEX high modules, four slotters are QUAD height modules, two slots and you are a DUAL height module, while the lowly one slot is a SINGLE height module. These backplanes come in a variety of configurations. There is a processor backplane, a memory backplane, a DH-11 backplane, other controller backplanes, and last the standard UNIBUS backplane that comes in 4 wide or 9 wide varieties to hold 4 or 9 modules. Well, our 11/40 had the processor backplane with the processor in it. The processor backplane also holds the memory management board, the stack limit board, the DL-11 console interface board, the KW-11L clock board, the EIS board, and the optional FIS (floating instruction set board,



SAVE YOUR COMPUTER

The PROTERM 80 saves CPU time by "debugging" on-line without the text editor. Now you can run Basic on-line the way Basic was meant to be used. The PROTERM 80 interacts directly with Basic and permits immediate syntax checking and on-line debugging that you have never had before. PROTERM 80's 48 user-defined key functions (programmable from the keyboard or downloadable) can save thousands of hours in programming time as well as CPU time. With sales of over 4,000 CRT terminals, our quality and reliability has proved second-to-none. DTI offers as standard, a one-year warranty. For data entry as you have never before seen, write to DTI for more details about the PROTERM 80.

dti

DATA TERMINALS CORP.
45 West Street
Medfield, Mass. 02052 U.S.A.
Tel. (617) 359-4188
TELEX 924-325

THE PROTERM 80

CIRCLE 8 ON READER CARD

Still $200! MORE FEATURES!

$200. RSTS/E* WORD PROCESSOR CBEDIT.BAS

Basic-Plus* program with VDT input, window edit and document save. Add, locate, change, replace, delete, block move and copy merge, etc. Crash and operator error recovery. Supports DEC, Hazletine and Mime standard VDT's. Others easy to add.

Fully formatted output (margins, justify, center, underscore, super-sub-script, readers, page numbers, etc.), to typewriter, line printer or disk. Bidirectional driver for Diablo-Xerox 16-17.

User's manual and source code included.
9-Track $200. RKO5 $260. ppd
T. F. Hudgins & Associates, Inc.
P.O. Box 10946, Houston, Texas 77018
Woods Martin 713/682-3651
*TM Digital Equipment Corporation
CIRCLE 9 ON READER CARD

which is on the 11/40 not supported with the 4 word math package and it, therefore, is almost useless). The only other backplane in the computer was a memory backplane capable of holding 32KW of old DEC memory.

No other backplane (DD-11C for 4 modules, DD-11D for 9 modules) was in the computer! Fortunately these are available from many used sources, new from DEC or new from several other suppliers. We opted for a new 9 module backplane from Able computer (see inside back cover). While the DEC backplane is a wire-wrap nightmare, the Able people have managed to do it with only one wire wrap (tell you about that one later); it is all a multi-layer printed circuit board, neat and compact. Installing a backplane should be easy; just mount it in the box (BA-11) and plug in the two power connectors. Our connectors, male and female, would not mate. Old style BA-11 and new style backplane. Order old style connectors, install and you're ok. Thank you DEC supplies catalog and 800 number! OOPS, extra wires on the backplane that the prints say shouldn't be there. They are right; the backplane has wires for battery backup of Semi-Conductor memory and some +20 Volt wires that our BA-11 doesn't have. Just wrap them up and don't connect them.

Now we have a computer, small memory, and a 9 module backplane. We decided to remove the small bulky DEC memory and replace it with a single board 256KB memory. We took CDC because it was available right then, but today it is available from most of the memory suppliers. It is parity memory, but it requires an external 7850 parity controller. Some of the new ones have all of this right on the board itself. The memory comes with good instructions on installation and configuration so in it went. On went our toggle in diagnostic and we were running our tests on 32K of MOS memory in a new backplane. In order to access more that 32KW on an 11/40 you must have the memory management board. We shopped around again and found one with the COMPUCOM people in Atlanta, GA. Nice people. Shipped it immediately. While installation of this board is easy (it goes in the processor slot assigned for it), there are options on about 3 other processor boards that have to be changed. This will require some soldering of wire straps on these boards, all indicated in the memory management manual and the 11/40 processor manual as well as on the prints themselves. Once these changes were made and a new diagnostic toggled in, we were running diagnostics on the entire 256KB of memory. We later found that the KJ-11 should have been included with the memory management and the folks at Compucom shipped it right out.

Next issue: The disk, The magtape, The multiplexers, The printer.

The RSTS PROFESSIONAL recognizes that a problem does exist in our industry with illegal use of software. With this issue we begin a series on these problems — and solicit your comments and suggestions for this feature.

Have You Stolen Any Good Software Lately?

Richard A. Marino, Data Processing Design, Inc.

You are probably a thief! If not, you probably harbor one or are an accomplice to one. I used to have high expectations, but I no longer believe in honesty — when it comes to software. Almost all of us, have, at one time or another, been a SOFTLIFTER.

Is the theft of software that pronounced? BELIEVE IT! It's not always the kind of theft you might think. There certainly are cases of intentional, for-profit, thievery. Unfortunately vendors are finding that former employees who claim they are not stealing software but just re-inventing it find little support from the judiciary. But most software theft is less pronounced and rather innocent. It's the casual borrowing of software from another site to 'try it out'. Or running software you've purchased for one system on three. And it's not just the small companies — one of our users, a multi-million dollar institution, just got caught running WORD-11 on two unlicensed systems.

This type of theft is hard to track. People do not fink. I don't even expect my best friends and most faithful customers to tell me when they suspect somebody is running DPD's software illegally. This type of theft is unpolicable — but it is damaging. When a company like ours or any other large, successful software house puts decades of work in developing and documenting a product, it deserves the benefits. This casual thievery in the long run hurts the user community as well. The investment in designing and building a new product along with the costs of documentation and support are more difficult to justify if one can't protect the investment by stopping the theft of the software.

The intangible nature of software and the difficulty of protecting it from clever as well as casual thievery weighs heavy on the minds of software producers. Some go to great effort to protect the software, sometimes including complex security routines in their software. But there are no perfect solutions.

What will be the result of this continuing illegal activity? There will certainly be more and more lawsuits. Despite the three-year wait we face to get a trial date and the possibility that there will be no monetary reward for winning our case, like all software producers we must pursue cases of obvious theft to their conclusion in court. Higher priced software may also be the result, to cover some of these new unexpected costs of the marketplace. Perhaps there will be fewer products coming to market and less innovation from the small software house unwilling to risk the callousness of the open marketplace. The problem is growing. It is racking the microcomputer marketplace and is threatening those of us producing software for the business minicomputer market as well. It is unfortunate we face this problem — and believe me we face it together. Innocent theft is still theft and it is not a victimless crime.

"Don't buy a PDP 11"

VAX, PDP & DEC are trade marks of Digital Equipment Corporation

HAMILTON RENTALS
6 Pearl Court, Allendale, N.J. 07401

PLEASE RUSH ME A QUOTATION BASED ON THE FOLLOWING SYSTEM: CIRCLE *ONE* ONLY IN EACH LINE

Processor	Minc	11/03	11/23	11/24	11/34	11/44	VAX 11/750
Disks	RX02	RL01	RL02	RK07	RM02	RM80	
Operating System	RT11	RSX11M	RSTS/E	VAX/VMS	CTS		
Software	RABBIT RSTS UTILITIES	RABBIT VAX UTILITIES	WORD PROCESSING	OTHER			
Rental	6 Months	12 Months	18 Months				

Name ______________________

Position ______________________

Company ______________________

Address ______________________

City ______________ **State** ______________ **Zip** ________

Telephone Number ______________________

RS

"...With departmental budget cuts, high interest rates, hardware prices falling back, an upgrade down the road..."

"But I've always bought PDP11s. They're the best."

"I didn't say don't USE a PDP11...just don't BUY one!"

"Huh?"

"HAMILTON will rent you the latest in DEC systems—popular 11/23s, hot new 11/24s and 11/44s, and all on a simple no-deposit monthly payment basis. Great for fitting into tight budgets. You'll get prompt delivery, the systems configured the way YOU want them, AND they'll bundle in your choice from HAMILTON's extensive software and utilities library."

"That sounds great, but don't these rental schemes lock you in for 5 years?"

"No, not if you rent from HAMILTON. If you know you're never going to change so much as a terminal for 3-5 years, then go talk to your bank about a long term lease. But, if you want the flexibility of upgrading either all or any part of your system when you need to, then go talk to HAMILTON about rental. They'll listen."

"That's really interesting. I didn't know I had that kind of rental alternative."

"And what's more, you can talk to HAMILTON about their rental-with-purchase-option plan which gives you the right to buy the system when you're ready."

"But will DEC maintain my HAMILTON rental systems?"

"DEC maintenance is already INCLUDED in your monthly rentals. Plus, HAMILTON handles all the shipping and guarantees you a really smooth installation."

"HAMILTON certainly has a comprehensive rental program."

"Of course. They have more experience in renting DEC systems than anyone else, with thousands of installations in Europe and North America."

"You know, if they hadn't limited my department budget this year, I could have gone straight to a VAX 11/750. Hey. I don't suppose HAMILTON rents....."

"11/750s? Of course! Call them now...
Toll Free 800-631-0298
In New Jersey 201-327-1444
In Canada 416-251-1166
or complete the coupon for an individual rental quotation on a system of your choice."

HAMILTON RENTALS

Pearl Court, Allendale, N.J. 07401
One Penn Plaza, New York, N.Y. 10119
415 Horner Ave., Toronto, M8W4W3

LONDON PARIS DÜSSELDORF CALGARY

CIRCLE 10 ON READER CARD

VERSION 2.1 NOW AVAILABLE:

QUE-11 — V2.1

ONE JOB SPOOLER FOR RSTS/E CONTROLS ALL SPOOLING

REPLACES:

- SPLIDL, SPLRUN . . . etc.
- BATIDL, BATRUN . . . etc.
- QUEMAN, QUMRUN . . . etc.
- OPSER, OPSRUN . . . etc.
- ATPRO
- ATPK

QUE-11:

- DEC QUE Compatible
- One job controls all spooling
- Saves small buffers and job slots
- Spawns jobs as needed
- Handles line printer and keyboard spooling
- Controls as many BATCH JOBS as pseudo-keyboards
- Full parameter replacement in QUE calls
- "DO" command replaces indirect processors
- QUEMAN SYS call supported
- Program deliveries — NOW
- Only $995 single cpu license
- Trial Version $100

For more information contact:

On Track Systems, Inc.
P.O Box 245
Ambler, PA 19002-0245
Phone: 215/542-7133

CIRCLE 11 ON READER CARD

Software Product Description

Product Name: LOCK-11 Version 2.0

Description:

Lock-11 is a security superstructure built upon the standard RSTS password structure that provides the following extensions:

- Absolute control of system access by keyboard. Manager may limit any keyboard to certain accounts or groups of accounts and control time of access.
- Auto-login (with or without password) and chain with specified core common contents by KB.
- Real time system surveillance. Manager specifies a list of alarm keyboards which log all infractions and probes as they happen. Opser is not required.
- Password knowledge is no longer carte blanche system access. System detects unauthorized use of passwords. Privileged passwords don't work on non-privileged keyboards. Non-privileged passwords work only on specified keyboards.
- Manager may establish special priority/burst settings by KB.
- Manager specifies a list of console keyboards from which security file editor may operate.

Minimum Hardware/Software Required:

Any valid RSTS/E system running Version 7.0 or later. Any version of RSX emulation is needed.

Support: See License Agreement

Installation: User Installed

Ordering Information:

Available on 9 track 800 or 1600 BPI tape. Multiple CPU discount schedule:

First license	0% discount
Second thru Third license	50% discount
Fourth thru Twentieth license	70% discount

Licensed users desiring source code for internal use only must execute a separate Program Sources License Agreement. Sources are available at ten times the initial license fee.

License Fee:

Single CPU license: $850.00. Annual maintenance at 12% of current list price.

Contact:

Dave Mallery
Nationwide Data Dialog
215—364-2800

CIRCLE 12 ON READER CARD

LD1: [1,3] EXTEND.BAS

By Scott Banks and Nationwide Data Dialog

```
        EXTEND
10      !                                                               &
        !       EXTEND.BAS                                              &
        !                                                               &
        !       WE TAKE AN INPUT FILE (DEFAULT .BAS) AND CREATE AN      &
        ! OUTPUT FILE (DEFAULT .BES) THAT WILL BE IN EXTEND MODE.       &
        ! IF THE FIRST LINE OF THE PROGRAM IS NOT LINE 1, WE WILL ADD   &
        ! A LINE 1 WITH AN EXTEND STATEMENT.  OTHERWISE, YOU MUST ADD   &
        ! 'EXTEND' YOURSELF.                                            &
        !                                                               &
        !       A NOTE ON COMMENTS.  BASIC-PLUS, UPON SEEING A          &
        ! LINE-FEED AFTER AN EXCLAMATION POINT, TAKES ALL FOLLOWING     &
        ! TEXT AS COMMENTS.  WE WILL ADD THE '!' IF NECCESSARY.         &
        ! HOWEVER, FOR ANY LINE THAT CONTAINS AN '!', WE MUST ASSUME    &
        ! THAT IT IS FOR COMMENT PURPOSES.  IF YOU HAVE AN '!' IN       &
        ! QUOTES (A PROGRAM CHAIN TO [1,3] PERHAPS), YOU MUST EXAMINE   &
        ! THIS AND REPAIR IT YOURSELF IF NEED BE.                       &
        !                                                               &
        !       PROGRAMS, OR PROGRAM SECTIONS THAT PREVIOUSLY WORKED    &
        ! IN EXTEND MODE ARE SELDOM A PROBLEM.  THERE MAY BE SOME       &
        ! COMPLICATIONS IN CONVERTING CODE THAT WAS ORIGINALLY WRITTEN  &
        ! FOR NOEXTEND MODE.  IN EXTEND MODE, UNLIKE NOEXTEND MODE,     &
        ! SPACES OFTEN TAKE ON GREAT SIGNIFICANCE.                      &
        !                                                               &
        !       BASIC-PLUS KEYWORDS MUST NOT CONTAIN SPACES IN EXTEND:  &
        !                                                               &
        !               999     F N E N D               (NOEXTEND)      &
        !               999     FNEND                   (EXTEND)        &
        !                                                               &
        !       SPACES MUST SEPARATE KEYWORDS:                          &
        !                                                               &
        !               1020    GOTO1090                                &
        !               1020    GOTO 1090                               &
        !                                                               &
        !       AND WATCH LSETS AND RSETS LIKE THIS:                    &
        !                                                               &
        !               4320    LSETA$ = 'ZZ'                           &
        !               4320    LSET A$ = 'ZZ'                          &
        !                                                               &
        !       FINALLY, ONCE A PROGRAM HAS BEEN CONVERTED AND WHATEVER &
        ! YOUR PROBLEMS, ALL EDITING CAN THEN BE DONE WITH TECO.        &
        ! A BARGAIN AT TWICE THE PRICE!                                 &
        !
90        GOSUB 19600
200       F.BAS% = 1%                                                   &
        \ F.BES% = 2%                                                   &
        !
1000      GOSUB 2000                            ! OPEN INPUT FILE       &
        !
1040      GOSUB 3000                            ! OPEN OUTPUT FILE      &
        !
1200      GOSUB 3200                            ! BUILD OUTPUT FILE     &
        !
1600      GOSUB 2800                            ! CLOSE INPUT FILE      &
        !
1620      GOSUB 3800                            ! CLOSE OUTPUT FILE     &
        !
1880      GOTO 1000                                                     &
        !
2000    !       OPEN INPUT FILE                                         &
        !
2020      PRINT #KB%, 'Input file? ';           ! EOF -> 32767          &
        \ INPUT LINE #KB%, Z$                                           &
        \ F.BAS$ = CVT$$(Z$,255%-64%)                                   &
        \ GOTO 32767 UNLESS LEN(F.BAS$)                                 &
        \ F.BAS$ = F.BAS$ + '.BAS'                                      &
                        UNLESS INSTR(1%,F.BAS$,'.')                     &
        !
2100      OPEN F.BAS$ FOR INPUT AS FILE F.BAS%,                         &
                                RECORDSIZE 4096%,                       &
                                MODE 8192%                              &
        !
2190      RETURN                                                        &
        !
2800    !       CLOSE INPUT FILE                                        &
        !
2820      CLOSE F.BAS%                                                  &
        !
2990      RETURN                                                        &
        !
3000    !       OPEN OUTPUT FILE                                        &
        !
3020      PRINT #KB%, 'Output file? ';                                  &
        \ INPUT LINE #KB%, Z$                                           &
        \ F.BES$ = CVT$$(Z$,255%-64%)                                   &
        \ IF    F.BES$ = ''                                             &
          THEN  Z% = INSTR(1%,F.BAS$+'.' , '.')                         &
        \       F.BES$ = LEFT(F.BAS$,Z%-1%)                             &
        !
3040      F.BES$ = F.BES$ + '.BES'                                      &
                        UNLESS INSTR(1%,F.BES$,'.')                     &
        !
3100      OPEN F.BES$ FOR OUTPUT AS FILE F.BES%                         &
        !
3190      RETURN                                                        &
        !
3200    !       CONVERT .BAS PROGRAM TO .BES                            &
        !
3220      BAS.COM% = 0%                         ! NOT IN COMMENT MODE   &
        !
3240      INPUT LINE #F.BAS%, BAS.LIN$                                  &
        \ Z$ = CVT$$(BAS.LIN$,-1%)                                      &
        \ Z% = ASCII(RIGHT(Z$,2%)) AND 127%                             &
        \ IF    LEFT(Z$,1%) <> '1'                                      &
          OR    LEFT(Z$,1%) =  '1'                                      &
          AND   Z% >= 48% AND Z% <= 57%                                 &
          THEN  PRINT #F.BES%, '1'; CHR$(9%); '  EXTEND'                &
        !
3280      GOTO 3320                                                     &
        !
3300      INPUT LINE #F.BAS%, BAS.LIN$          ! EOF -> 3390           &
        !
3320      BAS.LIN$ = CHR$(9%) + '!' + CVT$$(BAS.LIN$,8%)                &
                        UNLESS LEFT(CVT$$(BAS.LIN$,8%),1%) = '!'        &
                        IF BAS.COM%                                     &
        !
3340      BAS.EOL% = ASCII(RIGHT(BAS.LIN$,LEN(BAS.LIN$)))               &
        \ IF    BAS.EOL%                                                &
          THEN  BAS.LIN$ = CHR$(9%) + '!'                               &
                                IF LEN(BAS.LIN$) <= 2%                  &
        \       PRINT #F.BES%, CVT$$(BAS.LIN$,4%)                       &
        \       BAS.COM% = 0%                                           &
        \       GOTO 3300                                               &
        !
3360      BAS.COM% = INSTR(1%,BAS.LIN$,'!')                             &
        ! BAS.COM% = LEFT(CVT$$(BAS.LIN$,8%),1%) = '!'                  &
        \ PRINT #F.BES%, CVT$$(BAS.LIN$,4%); CHR$(9%); '&'              &
        \ GOTO 3300                                                     &
        !
3390      RETURN                                                        &
        !
3800    !       CLOSE OUTPUT FILE                                       &
        !
3820      CLOSE F.BES%                                                  &
        !
3990      RETURN                                                        &
        !
19600   !> ------ S T A N D A R D   I N I T I A L I Z A T I O N ------  &
          ON ERROR GOTO 19800                                           &
        \ KB%=12%                                                       &
        \ OPEN 'KB:' AS FILE KB%, MODE 4%                               &
        \ RETURN                                                        &
        !
19800   !> ------ L O C A L   E R R O R   T R A P S ----------------    &
          E%=ERR                                                        &
        !
19810     RESUME 3390 IF ERL=3300% IF E%=11%            ! SOURCE EOF
19820     RESUME 32767 IF ERL=2020% IF E%=11%           ! INPUT EOF
19990     GOTO 32000                                                    &
        !
20000   !> ------ S T A N D A R D   R O U T I N E S ----------------    &
        !
32000   !> ------ S T A N D A R D   E R R O R   T R A P S ----------    &
        !
32500     ON ERROR GOTO 0                                               &
        \ STOP                                                          &
        !
32560   DEF FNE$(E%)=CVT$$(RIGHT(SYS(CHR$(6%)+CHR$(9%)+CHR$(E%)),3%),5%) &
        !
32767     END
```

Word Processing* RSTS/E

*Word-11 by
Data Processing Design, Inc.
181 W. Orangethorp Avenue
Placentia, CA 92670

On Track Systems Provides:

- Sales
- Service
- Installation
- Demonstrations
- Training
- Consulting

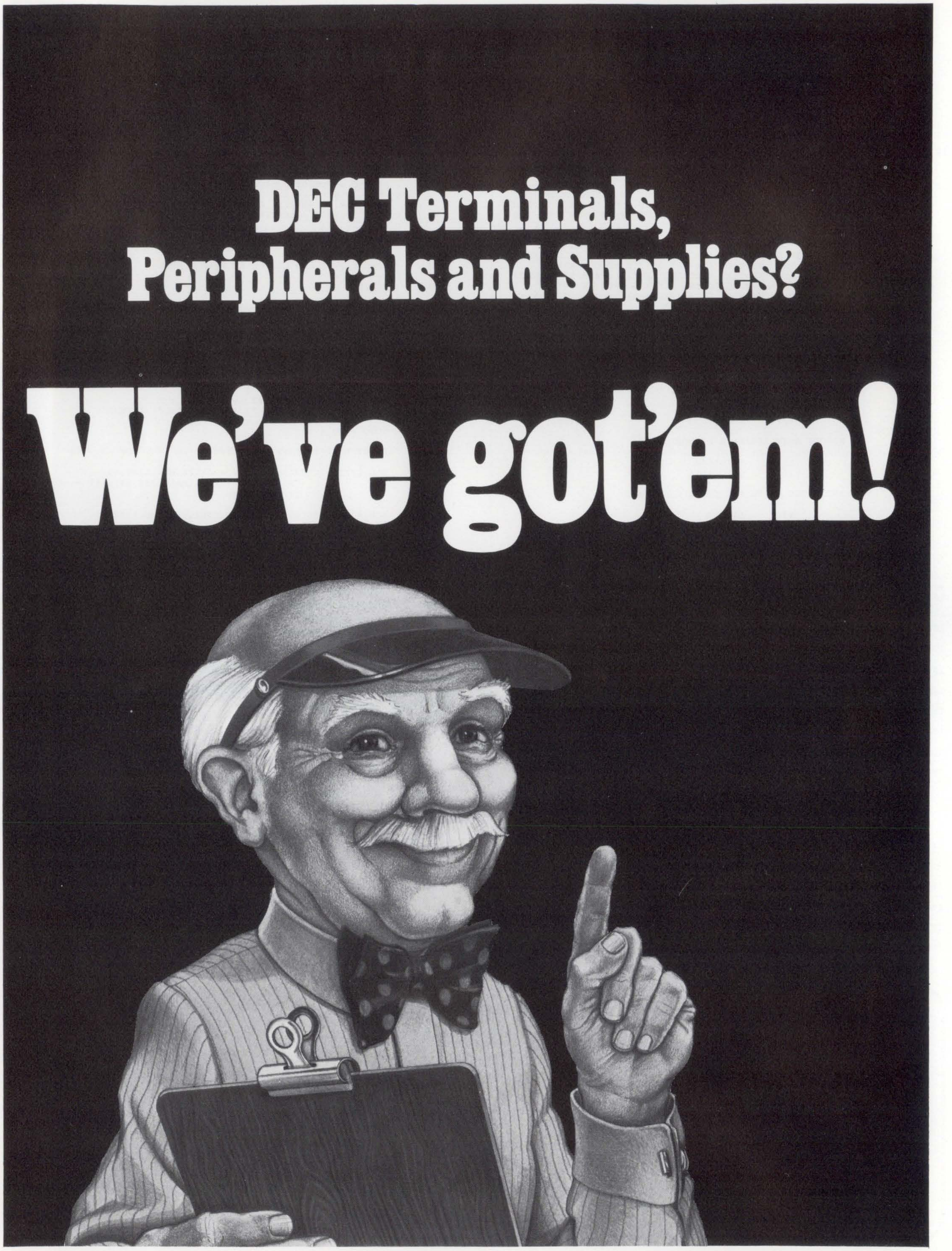
At your convenience!

At your office!

On Track Systems, Inc.

**P.O. Box 245
Ambler, PA 19002-0245
(215) 542-7133**

CIRCLE 13 ON READER CARD


DEC Terminals,
Peripherals and Supplies?
We've got'em!

And we'll get 'em on their way to you right away —
usually within 24 hours. Exactly what you wanted.
At the right price. In time. On time. Every time.
Anywhere in the continental U.S.A.
You can count on us.
Because we're The Suppliers.

DEC Peripherals

BA11KE
DD11CK/DK
DH11AD
DL11E
DZ11-A,B,C,D,E,F
FP11-A
MS11LB
MS11LD
MSV11DD
DLV11J
RM05-AA
RK07EA
RK711EA
RL02-AK
RL211-AK
RL01-AK
RL11-AK
RLV21
RLV11
RX211-BA
RXV21-BA
TJE16-EA
TWE16-EA
TWU45-AA
KDF11-AD,HD,HF,HH,HK
DR11C
MR11EA
VT103-AA,BA

Call us at (904) 434-1022

DEC Terminals

LA 34	DECwriter IV
LA 36	DECwriter II
LA 38	DECwriter IV
LA 120	DECwriter III
LA 180	DECprinter I
VT 50	DECscope CRT
VT 52	DECscope CRT
VT 100	DECscope CRT
VT 103	DECscope CRT
VT 132	DECscope CRT

Call us at (904) 434-1022

DEC Supplies

Disk Cartridges and Packs
Diskettes
Diskette Storage Units
Mag Tape
Ribbons
Paper
Forms Caddy
Anti-Static Rugs
Print wheels
Terminal/Printer Stands
Forms Bursters
and Decollators
Check Signers
and a whole lot more

**Call us for our complete catalogue:
1-800-874-9748.
In Florida, call
(904) 434-1022.**

COMPUTER
&TERMINAL
EXCHANGE

We're The Suppliers

114 East Gregory Street Pensacola, FL. 32501

DEC is a registered trademark
of Digital Equipment Corporation

CIRCLE 14 ON READER CARD

A WORD ABOUT THE AUTHOR . . .
Rudy Bazelmans is a Software Analyst at Sykes Datatronics Inc. where he designs language processors and helps maintain a PDP-11/70. His address at Sykes is 375 Orchard Street, Rochester, New York 14606.

ARE MACROS WORTH USING?

By Rudy Bazelmans, Sykes Datatronics Inc.

ABSTRACT

Often programmers fail to utilize the tools which are available to increase productivity. One powerful tool for the programmer is macros. This paper discusses the advantages and disadvantages of using macros and addresses the question of whether to use a macro preprocessor or a macro assembler.

Introduction

It is often necessary for programmers to repetitiously program particular sequences of code. Examples of these repeated sequences would be:

- Standard parameter passing and subroutine linkage.
- Stack and register manipulation.
- Test and branch sequences.
- I/O routines and their calls.
- Error checking and aborting.
- Multiple byte arithmetic operations.

Many of these items are particularly error prone and require a certain amount of thought each time the routine is copied. Macros can offer a solution to some of these problems because they possess the conciseness of assembly language along with the ease of use of a high level language.

Macro instructions are one line abbreviations for a group of instructions. The programmer initially defines the group of instructions which make up the macro. Then subsequent calls to this macro will be expanded in-line. Macro instructions look very similiar to ordinary assembly language instructions. The parameters (or operands) passed on the macro call are used by the macro processor to customize each call.

Although macros have been used by programmers to simplify the programming of assembly language, they have also played an important part in the development of high level languages. SNOBOL for example, was initially implemented using macros. Fortran and PL/I have been revitalized thru the use of a structured macro package called Ratfor (RATional FORtran). Editors such as TECO (VTEDIT and EMACS) are incredibly powerful, this is partly due to their macro capability. Another area of computers which is heavily dependent on macros is system generation. On many Digital Equipment and Data General computers the system monitors are customized by having the user answer a number of questions and using these answers to include or exclude certain elements of the monitor.

Purpose

The main purposes of using macros are:

1. As an organizational tool to aid in the layout and design of a program or system.
2. As an abbreviation mechanism to help simplify the coding and understanding of programs.
3. As an error checking tool to aid in writing bug free code.
4. To aid in the correction of errors and changing programs.

Advantages of Using Macros

1. After a macro is developed, only one line of code is needed to perform a repetitive task. The time required to develop a macro is usually only as long as the time needed to program the sequence of code the first time and check it when it is used the second time.
2. Using macros reduces the number of bugs within a program because macros relieve the programmer of much of the housekeeping. A well designed macro will perform defaulting, error checking and local optimization. This reduces the amount of time which a programmer spends thinking about repetitive tasks.
   A. When certain parameters are not specified in the macro call, the macro can use default values in the expansion. For example, if no output device is specified, the macro can assume the output device is the terminal and generate the appropriate code.
   B. The macro can do error checking by flagging conflicting or missing parameters.
   C. If desired, local optimization can be accomplished automatically. For example, jumps can be replaced with branches and adds replaced with increments.

3. Macro calls usually consist of only one line of source code. As a result:
   A. The size of the listing is reduced, therefore reducing paper costs, output and processor time.
   B. Programs which contain macro calls are often easier to read and understand than programs which do not use macros.
      1) Macro calls hide confusing details which make programs hard to read.
      2) For programs which must execute very quickly, calls to subroutines can be replaced with macros instructions to insert the desired code in-line. This preserves the readability and ease of programming characteristic of subroutines, but eliminates the overhead of subroutine linkage.
   C. The program entry time is reduced which frees up processor and terminal time.
4. Using macros reduces the number of bugs within a program because once a macro is debugged, its internals do not have to be rechecked.
5. Although macro definitions must be stored on disk, space is conserved because libraries of macros can be shared among users.
   A. Macros which are common to all programs can be stored in the system library and referenced by everyone.
   B. Macros which are common to all the programs within a particular product line can be stored in the product library.
   C. Any macros which are specific to a particular program will have to be stored within that program, but disk space will still be conserved.

Disadvantages of Using Macros

1. Macros could have the effect of complicating programming if the programmer carries them too far. For example, the programmer may have too many parameters on the macro call making them difficult to read. Or he might try to handle too many special cases and as a result, increase the chance of errors.
2. In order to handle many cases, poorly designed macros could generate unnecessary code. An example of this would be a macro which unnecessarily saves all the registers.
3. In non real-time applications, users could inappropriately use macros to generate code in-line when the code should actually be made into a subroutine in order to conserve memory. Macros which have many lines of code which are non-variable are an example of this programming flaw.
4. There is some machine overhead involved in defining and referencing macros. The language processor (macro preprocessor or macro assembler) must initially read in the definition of each macro, then for each macro call, the actual parameters are substituted for the formal parameters found in the definition. There is no overhead on the resulting object code, only in the time spent translating the source program.

Advantages of Using a Macro Assembler Over a Preprocessor

1. The macro assembler has more flexibility because it can use the information contained within the symbol table. It is also possible to use EQUs and SETs to count loops and nesting levels.
2. Macros can be expanded without making additional passes through the input file and no intermediate file needs to be generated.
3. Many routines needed to expand the macros already exist in the assembler, such as: read a line, test the statement type, convert routines, etc.

Disadvantages of Using a Macro Assembler Over a Preprocessor

1. Designing a good macro assembler is complicated and will take more time than a macro preprocessor.
2. The macro assembler will be larger than an ordinary assembler, and since the code, symbol table and macros must all reside in memory at one time, the programmer will be limited in the number of symbols and macros which may be used. In order to program large applications using a macro assembler, a Linker is usually required. Table 1 shows the estimated amount of memory available for symbols, macros or both, for each laguage processor. The memory sizes are based on a job size of 28KW. As shown in the table, using a preprocessor and an assembler offers the user approximately 2¾ the amount of memory for symbols and macros.

TABLE 1

language	memory for		
processor	macros	symbols	both
preprocessor	24kw	—	—
assembler	—	20kw	—
macro-assembler	—	—	16kw

3. Since no temporary file is generated during the first pass of the assembler, all macro calls must be re-expanded during pass two.

Advantages of Using a Preprocessor Over a Macro Assembler

1. A preprocessor allows the programmer to use the power of macros to help program both high and low level languages. Besides being used with most assembly languages, macros have been successfully used with high level languages like Basic, Fortran and C. Macros have also been used with several editors like TECO and EDIT.

2. Macro languages can be fairly complicated, and a preprocessor allows the same macro syntax to be used with any assembler or compiler. This reduces the training time for employees.
3. The amount of memory for macros within a preprocessor would be about twice as much as within a macro assembler and the amount of memory available for symbols within an assembler would be about 60% more than the memory available within a macro assembler.
4. A preprocessor would be easier to design than a macro assembler and would take less time to implement.
5. A preprocessor expands the macros once whereas the macro assembler must expand them twice, thus reducing processing time.

Disadvantages of Using a Preprocessor Over a Macro Assembler

1. Many of the routines which are in the assembler must be duplicated within the preprocessor.
2. A preprocessor must make a pass through the source file.
3. A preprocessor must generate a temporary file for use as an input file for the assembler.

Structured Macros

A group from Duke University in North Carolina developed a set of structured macros early in 1974. These macros were implemented in order to provide PASCAL-like control structures for the PDP-11 assembly language. This package of macros called BIOMAC is perhaps the best known package of structured macros available. Below are the comments made by the authors of BIOMAC about their macro package:

1. There were two important reasons for implementing BIOMAC:
   A. To eliminate tedious test-and-branch code sequences.
   B. To provide a block structure capability to reduce the errors caused by the complexity of transfer instructions.

   The important macros which are available are: IF...THEN...ELSE, CASE, WHILE, REPEAT and FOR.
2. BIOMAC provides both the benefits of the assembly language (efficiency and direct control) as well as the ease of programming provided by a high level language.
3. The primary benefits of structured macros are in increased programmer productivity and error reduction.
4. Unlike good compilers, good macro processors are incapable of making global optimizations and they can not enforce variable scope rules.
5. BIOMAC adds approximately one jump instruction for each block (a block is the basic unit of a structured program). It is estimated that there is an overhead of about 5-10% in program size and speed.
6. Structured macros are useful when high level languages are not available for a particular machine or when these high level languages obscure the useful features of the machine in use.
7. Assembly languages don't have to be difficult.

General Comments

During a discussion concerning the overhead of high level macros in a book entitled Software Tools, Brian W. Kernighan and P. J. Plauger said; "We feel strongly that this [overhead] is absolutely immaterial in comparison to the benefits of this [macro] preprocessor. Even though a single compilation may cost twice as much, our experience has been that many fewer compilations are needed, because the code works sooner. If you can find a bug in a few minutes instead of a few hours it pays for a lot of compiling. The other saving is in the much reduced cost of modifying a program after it's been in service for a while." Kernighan and Plauger also warn us by saying; "Beware of becoming too clever with macros. In principle, [a macro language] is capable of performing any computing task, but it is all too easy to write unreadable macros that cause more trouble than they save work." Then at a later point in the book, "...whenever you can, let the machine do the work, for that is the ultimate purpose of building tools."

References

Calingaert, Peter. **Assemblers, Compilers and Program Translation.** Potomac, MD: Computer Science Press, Inc., 1979, chapter 4, pp. 73-98.

Donovan, John J. **System Programming.** New York: McGraw-Hill Book Co., 1972, chapter 4, pp. 111-148.

Eckhouse, Richard H. and L. Robert Morris. **Minicomputer Systems, Organization, Programming and Application (PDP-11).**2nd ed. Englewood Cliffs, NJ: Prentice-Hall, Inc., 1979, pp. 112-123.

Herman-Giddens, G.S., C.B. Warren, R.C. Barr and M.S. Spach. "BIOMAC: Block Structured Programming Using PDP-11 Assembly Language." **Software-Practice and Experience,** John Wiley Sons, Ltd., 1975, Vol. 5, pp. 359-374.

Kernighan, Brian W. and P.J. Plauger. **Software Tools.** Reading, MA: Addison-Wesley, 1976.

PDP-11 Macro-11 Language Reference Manual (AA-5075A-TC). Maynard, MA: Digital Equipment Corp., chapter 7, pp. 7-1 to 7-8.

Struble, George W. **Assembly Language/Programming the IBM System/360 and 370.** 2nd ed. Reading, MA: Addison-Wesley Publishing Co., 1975, chapter 16, pp. 375-389.

♥

DILOG NUMBER 1 FOR DEC 11*

With LSI 11/PDP 11 Software Compatible Disc/Tape Controllers Offering Single Board Low Power μP Based Design and Low Cost... Plus Many Other Good Reasons!

The reasons start with DILOG'S (Distributed Logic Corp's.) full time engineering and design staff. *Not outside suppliers.* That means when you contact DILOG for product selection or after sale service, you'll get "first hand" assistance... along with years of experience manufacturing μP based controllers that interface with DEC 11 CPUs.

The intelligent products you'll discuss all utilize common proprietary architecture and DILOG automated design techniques—products with exceptional reliability and cost efficiency... mostly available from stock. And when you plug a DILOG controller into your DEC CPU it's ready-to-run because it's *fully operating system software compatible.*

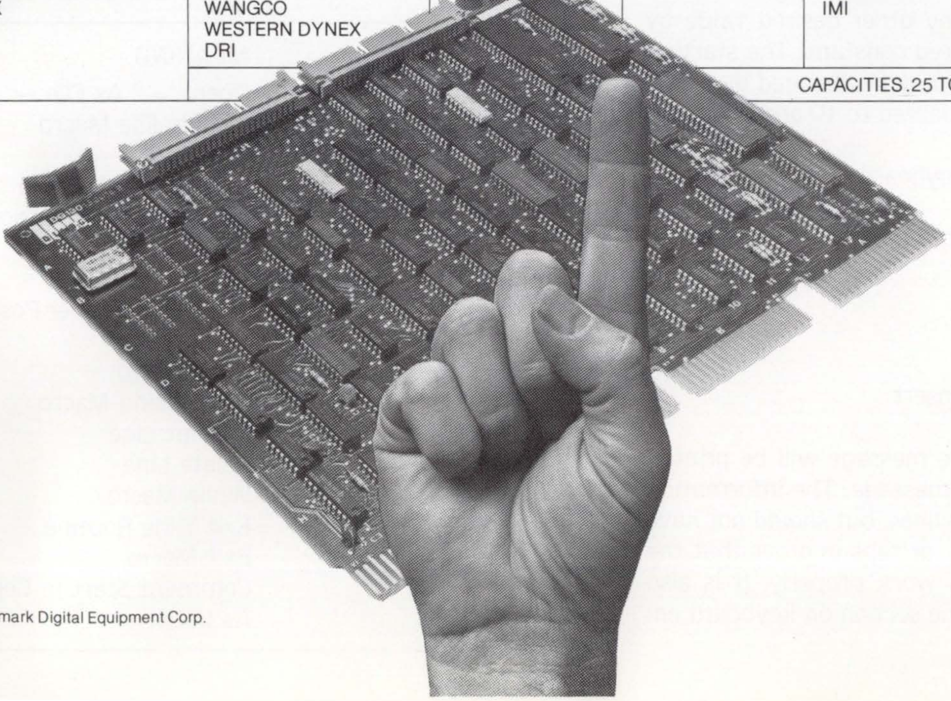
These high performance data storage interface products also feature • minimum bus/space requirements • up to 60% less power • 10 to 50% lower cost • automatic self-test... and numerous other features for easy system integration.

Consult the DILOG/disc-tape compatibility table for your needs. Then ask for detailed data on existing, or future products from DILOG... #1 in single board DEC 11 compatible disc/tape controllers.

Distributed Logic Corp., 12800-G Garden Grove Blvd., Garden Grove, CA 92643, Phone: (714) 534-8950 • TELEX: 681 399 DILOG GGVE

DISC/TAPE DRIVE MANUFACTURER COMPATIBILITY CHART

MAGNETIC TAPE	DISC					
½″ REEL-TO-REEL STD. & STREAMER	2315/5440/RK05 CARTRIDGE CLASS	CMD CARTRIDGE MODULE	SMD STORAGE MODULE	WINCHESTER 5¼″, 8″ OR 14″	¼″ TAPE CARTRIDGE	FLOPPY DISC DRIVE
AMPEX	AMPEX	AMPEX	AMPEX	BASF	DEI	BASF
CIPHER	CAELUS	CONTROL DATA	CENTURY DATA	CONTROL DATA	KENNEDY	DECITEK
CONTROL DATA	CENTURY DATA		CONTROL DATA	FUJITSU	PEREX	MICROPOLIS
DIGI-DATA	CONTROL DATA		BALL COMPUTER	KENNEDY	QUANTEX	PERTEC
KENNEDY	DEC		MITSUBISHI	MEMOREX		REMEX
MICRODATA	DIABLO			PRIAM		SHUGART
PERTEC	IOMEC			SHUGART		
TANDBERG DATA (IDT)	MICRODATA			SEAGATE		
WANGCO	PERTEC			QUANTUM		
TDX	WANGCO			IMI		
	WESTERN DYNEX					
	DRI					
	CAPACITIES .25 TO 300 MB					



DISTRIBUTED
LOGIC CORP.
DILOG

*Trademark Digital Equipment Corp.

CIRCLE 15 ON READER CARD

PROGRAM DEVELOPMENT SYSTEM

By David Whip, Maryland National Bank, Baltimore, Maryland

PDS is a set of teco macros designed to aid in the development of structured Basic+ and Basic+2 programs. Some of the features included are : "shorthand" entry of often used phrases "OPEN", "PRINT USING", etc.; automatic indentation of FOR-NEXT and WHILE-NEXT loops; user entered "shorthand" entry; file include for often needed code such as MAP or COMMONs; automatic line numbering; and automatic alignment of comments.

It is assumed in the following discussion that the user is familiar with teco at, at least, a novice level and understands pointer manipulation and search commands. PDS may be included automatically by renaming PDS.TEC to TECO.INI and entering teco as usual, or, after entering teco do EI PDS$$. The pointer position will not be changed by the EI command which may safely be used at any time. Several Q registers are used by PDS (see Table 1.) and should not be used for other purposes.

TABLE 1. Q Registers Reserved by PDS

NUMERIC
A,B,C,I,L,M,P,S,T
ALPHA
B,C,D,E,F,G,I,N,P,S,T,U,W

Once PDS has been loaded it is possible to change 4 preset values. The position at which comments will be aligned is kept in Q register C and is initialized to 50. This value may be changed by entering 30UC at which time newly entered comments would be aligned at column 30. The indentation constant is loaded in Q register I. This is the number of spaces which will indent each FOR-NEXT and WHILE-NEXT loop. The indentation feature may be turned off by entering 0UI, or set to any other desired value by entering NUI where N is the desired constant. The starting line number is preset to 100 and may be changed by entering NUL; the line increment is initialized to 10 and is located in Q register M.

If desired the file include or keyboard entry macros may be invoked at this time.

To use the file include macro type MI$$

```
*MI$$
File to include:
FOO.BAS
Loaded . . . Use CTRL B To Insert
```

If the file is found the above message will be printed otherwise there will be an error message. The information to be inserted may be on multiple lines, but should not have any leading line numbers, spaces, or tabs in order that the automatic indention feature will work properly. It is also legal to use the tilde insert (see the section on keyboard entry macros).

The keyboard entry macro is called by entering ME$$

```
*ME$$
Enter Macro (End With CTRL Z) :
For ~ = 1 to N% \
  Print Date$(1000% + ~) \
Next ~
↑Z

*
```

The tilde (~) is used as an escape character in the macro definitions. Its purpose is to allow the user to type in information and have the macro continue after the entry is made. In the above case when the macro is invoked <For > will be printed and the keyboard moniter will be called recursively. The user may type in as many characters as desired followed by <CTRL Z> which will cause < = 1 to N% \ <CR><LF> Print Date$(1000% + > to be printed and characters will again be accepted from the keyboard. At any point the user may abort the macro by typing <esc>.

Either of the two macros may be used multiple times from the teco command level during a session, of course only the last macro entered or file read will be available to the user. Any indentation included in the macro must be implicitly typed into the macro since the indentation Q register is not updated during the macro invocation.

PDS is started or restarted by typing MS$$ from the teco command level. After entering the MS command you will be in the keyboard monitor loop. All characters typed at the keyboard are trapped and processed before being echoed at the terminal. Certain characters have special meaning to PDS as shown in Table 2.

TABLE 2. Control Characters Recognized by PDS

CHARACTER	MEANING
CTRL A	Open "~" As File ~
CTRL B	Include File Macro
CTRL C	As Usual
CTRL D	User Entered Macro
CTRL E	Auto Line Numbering
CTRL F	For Macro
CTRL L	List Program
CTRL I <TAB>	Indents To Proper Position (uses spaces)
CTRL N	Next Macro
CTRL P	Print Using Macro
CTRL R	Reprint Line
CTRL U	Delete Line
CTRL W	While Macro
CTRL Z	Exit Tilde Routine
<ESC>	Exit Macro
<!>	Comment Start In Col C%
<DEL>	As Usual

The very nature of the shorthand approach makes it difficult to show the method of program entry, because the control characters are replaced by their associated strings.

```
<CTRL E><CTRL W>A< >B !<CR>
<TAB><CTRL P><ESC>A !<CR>
<TAB>A=A+1 \<CR>
<TAB><CTRL N><CR>
<CTRL E>END<CR>
```

FIGURE 1. Program As Entered At The Keyboard

```
100     While A< >B \
            Print A \
            A=A+1 \
        Next
110     END
```

FIGURE 2. Program As Typed At The Keyboard

Assume that MS$$ has been entered. Enter a line number <TAB>, or <CTRL E>, and <CTRL W>. <While> will print and the bell will sound. This is the indication that the keyboard monitor has exited from the inner loop associated with the while macro. Enter the condition associated with the while, now type <CR> and another <TAB>. Notice that automatic indentation has taken place. Type in some statements either with or without line numbers. In all cases the first <TAB> on a line will maintain the proper indentation. When you wish to end the loop enter the <TAB> at the beginning of a line, followed by <CTRL N>. As you see the effect is a <CR> with no <LF> followed by an extent to the proper position under the associated While. The same effect will be seen with the <CTRL F> or For Macro and <CTRL N>.

At any time you may exit from the PDS macro by typing <ESC> (except from the tilde entry macro). An example is the use of <CTRL P>. The terminal will type <Print> and pause. If you wish a plain print statement type <ESC>, the bell will ring, and you will still be in the keyboard monitor. To enter a <Print Using> statement you would type <CTRL Z>, Using> would print and the bell would ring. On the other hand, if you wished a <Print #1% Using> simply type #1% <CTRL Z> at the pause after <Print>. After exiting from PDS you may use any teco command you wish including dot moving. If MS is entered at the beginning of a line and a tab is typed, the correct indentation will be established. If the dot is in the middle of a line, the line up to the dot will be printed upon entry.

There are several restrictions on the use of PDS. The major restrictions deal with the indent feature. <CTRL F> and <CTRL W> must be used immediately after an initial <TAB> to have an indentation effect. If they are used at any other position on the line they are assumed to be statement modifiers and the indentation register is not adjusted. A line including <CTRL N> must have the <CTRL N> immediately after a leading <TAB> since its position is physically changed. If a line is deleted (<CTRL U> from within PDS) which contains <CTRL F>, <CTRL W> or <CTRL N> the indentation register will have to be adjusted. The method is to insert the opposite control character and then delete it. For example, <TAB><CTRL W><CTRL U> could be adjusted by <TAB><CTRL N><CTRL U>.

Clearly there is more that can be done with a system like PDS. For example it might be useful to develop a set of macros for Fortran or Runoff. Please let me know of any enhancements that you find useful in your applications. The basis of the keyboard monitor can be found in "Structured Programming In Teco" by Jacquie Stafsudd (RSTS Professional, Vol 2, Nbr 2) and "BLDCTL: Control File Processor" by David Spencer (RSTS Professional, Vol 3, Nbr 1). Anyone desiring further information on the type of text processing covered in this article should read Proceedings of the ACM SIGPLAN SIGOA Symposium On Text Manipulation Portland, Oregon June 8-10, 1981 published by the ACM.

```
! ----------------------------------!
!PDS.TEC V1.1 22-APR-81             !
!                                   !
!COPYRIGHT (C) 1981                 !
!DAVID WHIP                         !
!                                   !
! CHANGES TO V1.0 INCLUDING:        !
!    TRAP CTRL C                    !
!    SAVE ET CONTEXT                !
!    CTRL U NOW WORKS FOR SCREENS   !
!    FIX BUG IN <TAB> MACRO         !
!    FIX BUG IN CTRL N              !
!    FIX BUG IN AUTO LN             !
!                                   !
! RENAME TECO.INI OR                !
! EIPDS$$                           !
!                                   !
! ME - ENTER MACRO                  !
! MI - FILE INCLUDE                 !
! MS - START PDS                    !
!                                   !
! CTRL A - OPEN                     !
! CTRL B - FILE INCLUDE             !
! CTRL C - AS USUAL                 !
! CTRL D - USER MACRO               !
! CTRL E - AUTO LINE NBR            !
! CTRL F - FOR                      !
! <TAB>  - 1ST TAB INDENTS TO       !
!          PROPER POSITION IN LINE!
! CTRL L - LIST PROGRAM             !
! CTRL N - NEXT                     !
! CTRL P - PRINT USING              !
! CTRL R - AS USUAL                 !
! CTRL U - AS USUAL                 !
! CTRL W - WHILE                    !
! CTRL Z - EXIT TILDE<~> ROUTINE    !
! <ESC>  - EXIT MS                  !
! <EX.PT>- COMMENT START IN COL C%!
! <DEL>  - AS USUAL                 !
!                                   !
! Q REGISTERS                       !
! B$ - MACRO PROCESSOR                 A% - STRING INSERT POSITION      !
! C$ - OPEN MACRO                      B% - CURRENT CHARACTER FROM KB   !
! D$ - USER MACRO                      C% - INDENT COL FOR COMMENTS     !
! E$ - ENTER MACRO FROM KB             S% - SCRATCH                     !
! F$ - FOR MACRO                       T% - COLUMN TO INDENT TO - 7     !
! G$ - KEYBOARD MONITOR                I% - POSITIONS TO INDENT         !
! I$ - ENTER MACRO FROM FILE           L% - LINE NUMBER TO ENTER        !
! N$ - NEXT MACRO                      M% - LINE NUMBER INCREMENT       !
! P$ - PRINT USING MACRO            !
! S$ - START PDS                    !
! T$ - TAB AND INDENT HANDLER       !
! U$ - USER FILE MACRO              !
! W$ - WHILE MACRO                  !
! ----------------------------------!

@^UG
ETUO [0                               ! SAVE CONTEXT!
ET#8#32768ET                          ! TRAP CTRL C & TURN OFF ECHO!
<
 ^TUB

 QB-  1"= .UA GO MB F<          '   ! OPEN   !
 QB-  2"= .UA GU MB F<          '   ! FILE   !
 QB-  3"= .US :QS/^A/"S QS,.K   '
          QSJ <:DD;> QCET      '  ! CTRL C !
 QB-  4"= .UA GD MB F<          '   ! USER   !
 QB-  5"= 0^Q"= QL\ QL:=        '
          -1A"D QL \UL QL+QMLL  '
          MT F<                 '   ! AUTO LN!
 QB-  6"= 0^Q+QT+7"= QI%T       '
          .UA GF MB F<          '   ! FOR    !
 QB-  9"= MT F<                 '   ! TAB    !
 QB- 12"= 2< 13^T 10^T >
          @EW// :G* 2< 13^T 10^T >
          0,.T @^A/<.>/ .,ZT
          2< 13^T 10^T >
          OT F<                 '   ! LIST   !
 QB- 14"= QT-QIUT QI< 8^T > -QIC
          .UA GN MB F<          '   ! NEXT   !
 QB- 16"= .UA GP MB F<          '   ! PRINT  !
 QB- 18"= QR^T 13^T 10^T OT F< '   ! CTRL R !
 QB- 21"= ET&2"G 13^T -(0^Y)< 32^T > 13^T
               |
                QB^T 13^T 10^T'
          0K F<                              '    ! CTRL U !
QB- 23"= 0^.+.T+7"= QI%T                     '
          .UA QW MB F<                       '    ! WHILE  !
QB- 26"= 0;                                  '    ! CTRL Z !
QB- 27"= 0;                                  '    ! <ESC>  !
QB- 33"= -1A-32"N 0^QUS
          QC+QS"L 13^T 10^T QUS
          13@I// 10@I//                      '
                    .C+QS< 32^T 32@I// > '
          QB^T QB@I// F<                     '    ! COMMENT!
QB-127"= 0^Q"= 7^T                                ! <DEL>  !
                 |
                  R ET&2"= 0A^T
                          |
                           0A-32"L 2US
                                  |
                                   1US '
                           QS< 8^T 32^T 8^T > '
                  D '
            F< '
 QB@I// QB^T                                      ! CHAR   !
>
]0 QOET                                           ! RESTORE CONTEXT!


@^UB
1@I//
QAJ
<
 0A-^^~"= D MG                               '    ! TILDE !
 0A-   1"= D C;                              '    ! END   !
 QB- 27"= .US :@S/^A/"S QS,.K'0;  '               ! <ESC> !
 -1A-10"= MT                                 '    ! <LF>  !
 0A^T C                                           ! CHAR  !
> 7^T


@^UE
@^A/Enter Macro (End With CTRL Z) : / 13^T 10^T
```

```
.UP J MG 0,.XD 0,.K QPJ


@^UI
@^A/File To Include: /
J @I/:ER/ MG 27@I// 0XU 0K MLLS
13^T 10^T
QS"S ZUS A QS,ZXU QS,ZK
     @^A/Loaded . . . Use CTRL B To Insert/
    |
     @^A/?Can't Find File Or Account/ '
13^T 10^T '


@^US
Z-."N .UA 0L                                 ! AT EOF? _ NO !
\ ^E <0A-32"N 0; ' C >                       ! SKIP NBRS AND SPACES!
-(0^Q)-7UT ::@S/NEXT/"S QI%T '
QT"L 0UT            '                        ! SET T TO CRNT POS-7 OR ZERO IF T<0!
QAJ 0T '                                     ! RESET DOT CALL KBMON!
MG


@^UT
C^QUS (7*((-QS)/7)+7)+QS<  32^T 32@I// >
7+(0^Q)"= QT<32^T 32@I// > '


@^UW%While %
@^UF%For %
@^UN%Next %
@^UC%Open "~" As File ~ %
@^UP%Print ~ Using %
 50UC                                        ! COMMENT COL <- 50 !
  3UI                                        ! POSITIONS TO INDENT <- 3 !
100UL                                        ! START LINE NBR <- 100 !
 10UM                                        ! LINE NBR INCREMENT <- 10 !
@EI//
```

♥

CREATE.BAS

By James Swanson, Area Two Educational Computer Center, Mason City, Iowa

```
1       EXTEND
2       !&
        !               C R E A T E . B A S
        !&
!       CREATE.BAS will create a user account on-line.  It will create it  &
! as contiguous and extend it to its full length.  The UFD will be placed  &
! at or near the center of the pack or at any location indicated by the    &
! floating point variable "CENTR" at line 1000 which contains the block    &
! number to place the UFD at.                                              &
!       This program has been tested for only a short while but it has     &
! created accounts that have survived $ONLCLN as well as the REFRESH       &
! option in INIT.                                                          &
!&

3       !       PROGRAM NAME:   CREATE.BAS      &
        !       AUTHOR:         JIM SWANSON     &
        !       DATE WRITTEN:   16-JUN-81       &
        !       VERSION:        1.1             &

8       !  &
        !&
        !&
        !               C O P Y R I G H T  &
        !&
        !&
   !            Copyright (C) 1981 by  &
   !    Area Two Educational Computer Center, Mason City, IA  &
   !&
   !&
   !    This software is furnished free of charge to members of the
   !    North American DECUS organization and may be copied only
   !    with the  inclusion  of  the  above  copyright  notice.   This
   !    software  or  any  other copies thereof may not be provided or
   !    otherwise made available to any other person.  No title to and
   !    ownership of the software is hereby transferred.  &
   !&
   !    The information in this software is subject to change  without
   !    notice and should not be construed as a commitment by The Area
   !    Two Computer Center.  &
   !&
   !    ATECC assumes  no  responsibility for the use  or reliability  &
   !    of its software.         &

10      ON ERROR GOTO 19000     &
                !  STANDARD ERROR TRAP  &

150     !  1000                 ### CENTER OF DISK SOMEWHERE AFTER 61536 ### &

400     !&
        !&
                ! USER VARIABLES        &
        !&
        !       VARIABLE        EXPLANATION     &

450     ! ACCNT$                ACCOUNT IN FILE FORMAT FOR TESTING &
        ! C%(30)                FOR CHANGE-TO'S &
        ! CENTR                 LEFT OF CENTER OF DISK TO PLACE DIRECTORIES &
        ! DEV$                  DEVICE TO CREATE ACCOUNT ON &
        ! E$                    RSTS ERROR MESSAGE &
        ! FIL$                  NAME OF FILE OT BE OPENED IN MFD &
        ! M%(3585,7)            MFD VIRTUAL ARRAY &
        ! M.PTR%                POINTER INTO THE MFD ARRAY &
        ! M.UAA%                POINTER TO MFD'S ACCOUNTING ENTRY &
        ! M.UAR%                POINTER TO MFD'S UAR &
        ! MAXCLU%               LARGEST POSSIBLE UFD CLUSTER SIZE &
        ! MFDCLU%               M%(31%,0%)MFD CLUSTER SIZE &
        ! PAKCLU%               PACK CLUSTER SIZE &
        ! PAS1$                 FIRST 3 CHARACTERS OF PASSWORD &
        ! PAS2$                 SECOND THREE CHARACTERS OF PASSWORD &
        ! PASS$                 PASSWORD &
        ! PPN$                  PROJECT-PROGRAMMER NUMBER FORM: "(#,#)" &
        ! PPN%                  INTERGER VALUE FOR THE PPN &
        ! QU%                   QUOTA FOR THE NEW ACCOUNT &
        ! RAD.PAS1%             RAD-50 VALUE FOR FIRST 3 CHAR. OF PASSWORD &
        ! RAD.PAS2%             RAD-50 VALUE FOR SECOND 3 CHAR. OF PASSWORD &
        ! RAD.UFD%              RAD-50 VALUE FOR "UFD" &
        ! STA%                  VALUE OF THE MFD'S STATUS WORD FOR THIS UFD &
        ! U%(3585,7)            UFD VIRTUAL ARRAY &
        ! U.PTR%                POINTER TO THE NEW UFD FILE &
        ! UAR%                  STARTING DCN OF THE NEW UFD &
        ! UFDCLU%               UFD CLUSTER SIZE TO CREATE &

800     !&
        !&
                ! USER FUNCTIONS        &
        !&
        !       FUNCTION        DESCRIPTION     &

820     ! FNERR%( )             PRINTS A STANDARD ERROR MESSAGE &
        ! FNL%( )               DECIPHERS THE DIRECTORY LINK WORDS &
900     !&
        !&
        !       D I M E N T I O N   S T A T E M E N T S         &
        !&
        !&

902     DIM #1%, M%(3583%,7%)   &
        \ DIM #2%, U%(3583%,7%) &
                !> M%(3585,7) = MFD VIRTUAL ARRAY       &
                !> U%(3585,7) = UFD VIRTUAL ARRAY       &

903     DIM C%(30%)     &
                !> C%(30) = FOR CHANGE-TO'S     &

1000    MAXCLU% = 16%   &
        \ CENTR = 61536.        &
                !> MAXCLU% = LARGEST POSSIBLE UFD CLUSTER SIZE  &
                !> CENTR = CENTER OF DISK TO PLACE DIRECTORIES (BLOCK#) &
                !### MAXIMUM UFD CLUSTERSIZE SET AT 16 ####      &
                !### CENTER OF DISK SOMEWHERE AFTER BLOCK 61536 ###     &

1010    NULLBUF$ = STRING$(512%,0%)     &
                !> NULLBUF$ = 512 BYTES OF CHR$(0%)     &

1020    PRINT 'CREATE - Create and extend user accounts.'       &

2000    !&
        !&
        !       M A I N   P R O G R A M         &
        !&
        !&

2010    PRINT 'PPN to create';  &
        \ INPUT LINE S.PPN$     &
        \ PRINT '  ';PPN$       IF      ASCII(S.PPN$) = 61% !"="        &
        \ S.PPN$ = PPN$         IF      ASCII(S.PPN$) = 61% !"="        &
        \ PPN$ = CVT$$(S.PPN$,-1%)      &
        \ GOTO 32767    UNLESS  LEN(PPN$)       &
        \ PPN$ = '('+PPN$       UNLESS  INSTR(1%,PPN$,'(') > 0% &
        \ PPN$ = PPN$ + ')'     UNLESS  INSTR(1%,PPN$,')') > 0% &
        \ INPUT 'Device';DEV$   &
        \ DEV$ = 'SY:'  UNLESS  LEN(DEV$)       &
        \ DEV$ = DEV$ + ':'     UNLESS  INSTR(1%,DEV$,':') > 0% &
                !> S.PPN$ = PROJECT-PROGRAMMER # SAVED FOR "=" RESPONSE &
                !> PPN$ = PROJECT-PROGRAMMER NUMBER FORM: "(#,#)"       &
                !> DEV$ = DEVICE TO CREATE ACCOUNT ON   &

2012    INPUT 'Password';PASS$  &
        \ PAS1$ = MID(PASS$,1%,3%)      &
        \ PAS2$ = MID(PASS$,4%,3%)      &
                !> PASS$ = PASSWORD     &
                !> PAS1$ = FIRST 3 CHARACTERS OF PASSWORD       &
                !> PAS2$ = SECOND THREE CHARACTERS OF PASSWORD  &
                !  GET THE NECESSARY DATA       &

2015    FIL$ = 'UFD'+PAS1$+'.'+PAS2$    &
        \ ACCNT$ = DEV$+PPN$+FIL$       &
                !> FIL$ = NAME OF FILE OT BE OPENED IN MFD      &
                !> ACCNT$ = ACCOUNT IN FILE FORMAT FOR TESTING  &

2020    CHANGE SYS(CHR$(6%)+CHR$(-10%)+ACCNT$) TO C%    &
        \ RAD.UFD% = C%(7%)+SWAP%(C%(8%))       &
        \ RAD.PAS1% = C%(9%)+SWAP%(C%(10%))     &
        \ RAD.PAS2% = C%(11%)+SWAP%(C%(12%))    &
        \ PPN% = C%(5%)+SWAP%(C%(6%))   &
        \ IF    LEN(PASS$) = 0% &
        THEN    GOTO 2012       IF      C%(23%) = ASCII('S')    &
                                AND     C%(24%) = ASCII('Y')    &
        ELSE    IF      SWAP%(C%(28%)) < 0%     &
                THEN    PRINT 'Invalid password or account.'    &
                        \ GOTO 2000     &
                !> RAD.UFD% = RAD-50 VALUE FOR "UFD"    &
                !> RAD.PAS1% = RAD-50 VALUE FOR FIRST 3 CHAR. OF PASSWORD &
                !> RAD.PAS2% = RAD-50 VALUE FOR SECOND 3 CHAR. OF PASSWORD &
```

MENU/AUTHORIZATION PROCESSOR SYSTEM

M APS is a system to generate Application Menus and provide system security by controlling user access.

- Unlimited number of menu levels • Menu entry points may vary between users
- Three levels of user authorization privileges • "Keywords" to jump between menus
- Variable arguments passed between menu and program • Menus are all file driven
- And more . . .

Written in MACRO for RSTS/E, RT11, RSX11M, And VAX.

CALL OR WRITE: (414) 784-8250

McHUGH, FREEMAN AND ASSOCIATES, INC.
1135 LEGION DRIVE • ELM GROVE, WISCONSIN 53122

CIRCLE 57 ON READER CARD

```
                !> PPN% = INTERGER VALUE FOR THE PPN      &
                !  GET THE RAD 50 VALUES OF THE STUFF     &
                !  VERIFY THE VALIDITY OF THE DATA RETURNED        &

2040    ON ERROR GOTO 2043          &
        \ OPEN DEV$+PPN$ FOR INPUT AS FILE #2%  &
        \ PRINT 'This account already exists.'  &
        \ CLOSE #2%     &
        \ GOTO 2000     &

2043    !&
        !&
        !       C R E A T E    T H E    U F D         &
        !&
        !&
        RESUME 2045     &

2045    ON ERROR GOTO 19000        &
        \ OPEN DEV$+'[1,1]' FOR INPUT AS FILE #1%         &
        \ MFDCLU% = M%(31%,0%)  &
        \ PAKCLU% = M%(0%,4%)   &
                !> MFDCLU% = M%(31%,0%) MFD CLUSTER SIZE           &
                !> PAKCLU% = M%(0%,4%) PACK CLUSTER SIZE           &

2050    INPUT 'Account cluster size';UFDCLU%     &
        \ IF    (UFDCLU%*1./PAKCLU% <> UFDCLU%/PAKCLU%) &
        OR      (UFDCLU% < PAKCLU%)      &
        OR      (UFDCLU% > MAXCLU%)      &
        THEN    PRINT 'Illegal cluster size.'   &
                \ GOTO 2050      &
                !> UFDCLU% = UFD CLUSTER SIZE TO CREATE &

2055    UFDSIZ% = UFDCLU% * 7%  &
        \ UFDSIZ$ = '/SI:'+NUM1$(UFDSIZ%)         &
        \ UFDCLU$ = '/CL:'+NUM1$(UFDCLU%)         &
        \ LOC$ = '/PO:'+NUM1$(INT(CENTR/PAKCLU%))         &
        \ OPEN DEV$+'[1,1]'+FIL$+LOC$+UFDSIZ$+UFDCLU$ FOR OUTPUT AS FILE #2% &
                                ,MODE 16%        &
                !  OPEN THE ACCOUNT AS A FILE   &

2057    OPEN DEV$+'[1,1]' FOR INPUT AS FILE #1% &
        \ M.PTR% = FNL%(M%(0%,0%),MFDCLU%)        &
        \ UNTIL M.PTR% = 0%      &
        \ GOTO 2060     IF      M%(M.PTR%,4%) AND 64%   &
        \ GOTO 2060     IF      M%(M.PTR%,1%) <> RAD.UFD%        &
        \ GOTO 2060     IF      M%(M.PTR%,2%) <> RAD.PAS1%       &
        \ GOTO 2060     IF      M%(M.PTR%,3%) <> RAD.PAS2%       &
        \ GOTO 2063     &
                !  FIND THE UFD IN THE MFD       &
                !> M.PTR% = POINTER INTO THE MFD ARRAY  &

2060    M.PTR% = FNL%(M%(M.PTR%,0%),MFDCLU%)      &
        \ NEXT  &
        \ PRINT 'OOPS!!,  file not found again during reset search.'    &
        \ GOTO 32767    &
                !  SHOULD NEVER GET HERE        &

2063    !&
        !&
        !       B U I L D   U P   T H E   N E W   U F D   AND   M A K E &
        !&
        !         T H E   M F D   S E E   I T   A S   A   U F D   &
        !&
        !&

2064    PRINT 'Extending...';   &

2065    M.UAR% = FNL%(M%(M.PTR%,7%),MFDCLU%)     &
        \ FIELD #2%, 512% AS Z$ &
        \ FDCM$ = CVT%$(SWAP%(UFDCLU%)) &
        \ FDCM$ = FDCM$ + CVT%$(SWAP%(M%(M.UAR%,P%)))    &
                        FOR P% = 1% TO 7%        &
        \ RSET Z$ = NULLBUF$+FDCM$       &
        \ PUT #2%, RECORD X%    &
                FOR X% = 1% TO UFDSIZ%  &
                !> M.UAR% = POINTER TO MFD'S UAR        &
                !> U.PTR% = POINTER TO THE NEW UFD FILE &
                !  EMPTY THE ENTRIES IN THE NEW UFD     &
                !  SET UP THE UDCM IN EVERY 32ND ENTRY OF THE NEW UFD.  &

2067     STA% = SWAP%(60%) OR (16%+32%+64%)      &
         \ UAR% = M%(M.UAR%,1%)  &
         \ U%(0%,1%) = -1%       &
         \ U%(0%,6%) = PPN%      &
         \ U%(0%,7%) = RAD.UFD%  &
                !> STA% = VALUE OF THE MFD'S STATUS WORD FOR THIS UFD   &
                !> UAR% = STARTING DCN OF THE NEW UFD   &
                !  GET THE MFD'S UAR TO FIND THE CLUSTER NUMBERS OF THIS &
                !   NEW UFD.    &
                !  PUT THE PPN INTO THE NAME ENTRY OF THE MFD   &
                !  SET THE MFD ENTRY BIT (64) IN THE MFD STATUS WORD.   &
                !  SET UP THE LABEL ENTRY FOR THE NEW UFD       &

2070    PRINT IF POS(0%)        &
        \ INPUT 'Quota';QU%     &
        \ GOTO 2070     IF      QU% < 0%        &
        \ CLOSE #2%     &
        \ OPEN DEV$+'[1,1]' AS FILE #1%, MODE 16384%    &
        \ M.UAA% = FNL%(M%(M.PTR%,6%),MFDCLU%)  &
        \ M%(M.UAA%,0%) = 1%    &
        \ M%(M.UAA%,X%) = 0% FOR X% = 1% TO 5%  &
        \ M%(M.UAA%,6%) = QU%   &
        \ M%(M.UAA%,7%) = UFDCLU%       &
                !> M.UAA% = POINTER TO MFD'S ACCOUNTING ENTRY   &
                !> QU% = QUOTA FOR THE NEW ACCOUNT      &
                !  SET UP THE MFD'S ACCOUNTING ENTRY STUFF      &
                !   ZERO EVERYTHING AND PUT THE QUOTA AND CLUSTER SIZE. &
                !   AND PUT A "1" IN THE LINK ENTRY TO INSURE "IN USE". &

2075    M%(M.PTR%,1%) = PPN%    &
        \ M%(M.PTR%,4%) = STA%  &
        \ M%(M.PTR%,7%) = UAR%  &
        \ CLOSE #1%     &
                !  PUT THE PPN, STATUS WORD, AND UFD'S FIRST DCN        &
                !   INTO THE MFD ENTRY  &
                !  CLOSE UP THE MFD & FINISH UP.        &

2090    PRINT '  Account ';DEV$;PPN$;' placed at DCN# ';NUM1$(UAR%);'.' &

2100    !&
        !&
        !       HERE A ROUTINE TO MAKE AN ENTRY INTO AN "ACCT.SYS" OR OTHER &
        !       FILE CAN BE MADE.       &
        !&
        !&

4000    GOTO    2000    &

19000   !&
        !&
        !       E R R O R    T R A P S          &
        !&
        !&

19005   E$ = RIGHT(SYS(CHR$(6%)+CHR$(9%)+CHR$(ERR)),3%) &
                !> E$ = RSTS ERROR MESSAGE      &

19020   IF      ERL = 2020%     &
        THEN    PRINT "Invalid information, can't create."      &
                \ RESUME 2000%  &
                ! AND ERROR IN FILE-NAME-STRING-SCAN HERE       &

19980   IF      ERR > 49%       &
        AND     ERR < 53%       &
        THEN    PRINT E$        &
                \ RESUME        &
                !  BAD NUMBERS ON USER INPUT HANDLED HERE       &

19990   Z% = FNERR%('MAIN',ERR,ERL,'Aborting...')       &
        \ RESUME 32767  &

22000   !&
        !&
        !       D E F I N E     F U N C T I O N S       &
        !&
        !&

22002   DEF FNL%(L%,CLU%)       &
                =(((L% AND 3584%) /512%) *CLU%  &
                +(SWAP%(L% AND -4096%) /16%)) *32%      &
                +((L% AND 496%) /16%)   &
                !> FNL%( ) = DECIPHERS THE DIRECTORY LINK WORDS &

22060   DEF FNERR%(F.$,E%,EL%,M$)       &
        \ PRINT CHR$(7%);RIGHT(SYS(CHR$(6%)+CHR$(9%)+CHR$(E%)),3%);' in ';F.$  &
        \ PRINT '       at line';EL%    &
        \ PRINT M$      IF      LEN(M$) &
        \ FNEND &
                !> FNERR%( ) = PRINTS A STANDARD ERROR MESSAGE  &

32/67   END
```

SO YOUR DISK PACK IS IRREVOCABLY CORRUPT

By Mark Diebert, E.R. Squibb & Sons, Inc., East Brunswick, N.J.

```
10      EXTEND                                                           &
                ! This program supports extend mode                      &
   !******************************************************************** &
   !*****                                                          ***** &
   !*****  Neither the author of this program, nor the author's    ***** &
   !*****  employer warrant that this program will perform any     ***** &
   !*****  function whatsoever.  Any person(s) using any form of   ***** &
   !*****  this program upon any computer system or computer       ***** &
   !*****  related devices do so at their own risk.                ***** &
   !*****                                                          ***** &
   !*****  This program is intended solely for the purpose of      ***** &
   !*****  illustrating various data manipulation techniques as    ***** &
   !*****  an adjunct to the DECUS Symposium workshop entitled:    ***** &
   !*****           So Your "Disk Pack Is Irrevocably Corrupt".    ***** &
   !*****                                                          ***** &
   !******************************************************************** &
                                                                         &

90      ON ERROR GOTO 19000                                              &
                ! Set the standard error trap                            &

900     DIM Mfd.label%(7%),                                              &
                Mfd.fdcm%(3%,7%),Tmp.arayl%(7%),                         &
                Dir.field$(31%,7%)=2%                                    &
                ! Dim the in-core arrays for the MFD label entry,        &
                !  MFD FDCM, temporary print array                       &
                ! Dim the string array that will field the directory entries &

1000    PRINT "The function of this program is to demonstrate ";         &
                "simple RSTS/E directory handling."                      &
        \ PRINT "There are two read options available:"                  &
        \ GOSUB 11000                                                    &
        \ PRINT                                                          &
        \ GOSUB 11100                                                    &
        \ PRINT "Output to <KB:>";                                       &
        \ INPUT LINE Outfil$                                             &
        \ Outfil$=CVT$$(Outfil$,38%)                                     &
        \ Outfil$="KB:" UNLESS LEN(Outfil$)                              &
        \ OPEN Outfil$ AS FILE 12%                                       &
                ! Print the header                                       &
                ! GOSUB to print the various options                     &
                ! GOSUB to print the write instructions                  &
                ! Route the output                                       &

1010    INPUT "1-Read  2-Write a blockette  0-End";Code%                 &
        \ GOTO 32700 UNLESS Code%                                        &
        \ Input "Which disk";Disk$                                       &
        \ GOSUB 10000                                                    &
        \ ON Code% GOSUB 1050,3000 IF (Code%>0%) AND (Code%<3%)          &
        \ GOTO 1010                                                      &
                ! Get a macro option (read/write/end)                    &
                ! Get the disk name                                      &
                ! GOSUB to set up the pack information                   &
                ! GOSUB to the proper routine if the option is in range &
                ! Get another option                                     &

1050    PRINT                                                            &
        \ INPUT "Read option";Task%                                      &
        \ ON Task% GOSUB 1100,1200                                       &
                IF Task%>0% AND Task%<3%                                 &
        \ RETURN UNLESS Task%                                            &
        \ GOSUB 11000 IF Task%>2% OR Task%<0%                            &
        \ GOTO 1050                                                      &
                ! Get a read option                                      &
                ! GOSUB to the proper read routine if the option is in range &
                ! Return to macro option if no read option was entered   &
                ! GOSUB to the help text if the read option was out of range &
                ! Get another read option                                &
                                                                         &
                                                                         &

1100    INPUT "Enter link";Link.next%                                    &
        \ GOSUB 11200                                                    &
        \ PRINT "Link:";NUM1$(Link.next%);"  UFD Cluster:";              &
                NUM1$(Clus.next%);"  Block offset:";NUM1$(Block.next%); &
                "  Entry offset:";NUM1$(Entry.next%)                     &
        \ PRINT                                                          &
        \ RETURN                                                         &
                ! Get the link to be parsed                              &
                ! GOSUB to break apart the link                          &
                ! Print the components of the link                       &
                ! Return                                                 &

1200    Proceed%=0%                                                      &
        \ INPUT "DCN of blockette";Dcn.find%                             &
        \ PRINT "Block offset into DCN (0 to ";NUM1$(Pcs%-1%);")";       &
        \ INPUT Temp%                                                    &
        \ INPUT "Blockette position in the block (0 to 31)";Entry.find% &
        \ GOTO 1200 IF Entry.find%>31% OR Entry.find%<0% OR Temp%<0% OR &
                Temp%>(Pcs%-1%)                                          &
        \ GET #1%,BLOCK Dcn.find%                                        &
        \ Temp%=512%*Temp%                                               &
        \ GOSUB 11300                                                    &
        \ Tmp.arayl%(I%)=SWAP%(CVT$%(Dir.field$(Entry.find%,I%)))        &
                FOR I%=0% TO 7%                                          &
        \ PRINT #12%, "Word";TAB(10%);"Value";TAB(20%);"Lo Byte";TAB(30%); &
                "Hi Byte";TAB(40%);"RAD50"                               &
        \ PRINT #12%, NUM1$(I%);TAB(10%);NUM1$(Tmp.arayl%(I%));          &
                TAB(20%);NUM1$(Tmp.arayl%(I%) AND 255%);TAB(30%);        &
                NUM1$(SWAP%(Tmp.arayl%(I%)) AND 255%);TAB(40%);          &
                RAD$(Tmp.arayl%(I%)) FOR I%=0% TO 7%                     &
        \ PRINT #12%,                                                    &
        \ Proceed%=-1%                                                   &
        \ RETURN                                                         &
                ! Reset the "OK to proceed" switch (for write)           &
                ! Routine to dump a blockette                            &
                ! Input the DCN of the blockette                         &
                ! Input the block number within the DCN                  &
                ! Input the position in the block                        &
                ! Get the device cluster from the disk                   &
                ! Compute the offset into the buffer                     &
                ! GOSUB to field the buffer                              &
                ! Load the print array within the blockette              &
                ! Dump the blockette                                     &
                ! Set the "OK to proceed" switch (for writes)            &
                ! Return                                                 &
                                                                         &
                                                                         &

3000    PRINT                                                            &
        \ PRINT "You are now in the routine that allows the user to write" &
        \ PRINT "onto the disk. Are you sure that you want to do this"; &
        \ INPUT Temp$                                                    &
        \ RETURN UNLESS ASCII(CVT$$(Temp$,32%))=89%                      &
        \ GOSUB 1200                                                     &
        \ RETURN UNLESS Proceed%                                         &
        \ FOR I%=0% TO 7%                                                &
                \ PRINT "Enter word ";NUM1$(I%);                         &
                \ INPUT Tmp.arayl%(I%)                                   &
        \ NEXT I%                                                        &
        \ INPUT "Ok to write (Y/N)";Temp$                                &
        \ RETURN UNLESS ASCII(CVT$$(Temp$,32%))=89%                      &
        \ LSET Dir.field$(Entry.find%,I%)=CVT%$(SWAP%(Tmp.arayl%(I%)))   &
                FOR I%=0% TO 7%                                          &
        \ PUT #1%,BLOCK Dcn.find%                                        &
        \ PRINT "Blockette written."                                     &
        \ RETURN                                                         &
                ! Prompt for and get verification that we want to be here &
                ! Return unless we want to stay                          &
                ! REMEMBER: Non-file structured access required to write !! &
                ! GOSUB to get the blockette that we want to write to    &
                ! Enter the seven words of the blockette                 &
                ! Verify that we want to write the blockette             &
                ! Return unless we want to write                         &
                ! Place the blockette into the buffer                    &
                ! Put the record out to disk                             &
                ! Print the verification message                         &
                ! Return                                                 &
                                                                         &
                                                                         &

10000   OPEN Disk$ AS FILE 1%                                            &
        \ GOSUB 11300                                                    &
        \ GET #1%,BLOCK 1%                                               &
        \ Mfd.label%(I%)=SWAP%(CVT$%(Dir.field$(0%,I%))) FOR I%=0% TO 7% &
        \ Pcs%=Mfd.label%(4%)                                            &
        \ Mfd.fdcm%(0%,I%)=SWAP%(CVT$%(Dir.field$(31%,I%))) FOR I%=0% TO 7% &
        \ PRINT "Danger -- The link to the first MFD name entry is 0."  &
                UNLESS Mfd.label%(0%)                                    &
        \ PRINT "Danger -- Word 1 of the MFD label is not -1."           &
                UNLESS Mfd.label%(1%)=-1%                                &
        \ PRINT "The pack cluster size is";Pcs%                          &
        \ OPEN Disk$ AS FILE 1%, RECORDSIZE 512%*Pcs%                    &
                IF (Pcs%<17%) AND (Pcs%>0%) AND (((Pcs% AND 1%)=1%) XOR &
                        ((Pcs% AND 2%)=2%) XOR ((Pcs% AND 4%)=4%) XOR    &
                        ((Pcs% AND 8%)=8%) XOR ((Pcs% AND 16%)=16%))     &
        \ GET #1%,BLOCK 1%                                               &
        \ FOR I2%=1% TO 3%                                               &
                \ Temp%=512%*I2%                                         &
                \ GOSUB 11300                                            &
                \ Mfd.fdcm%(I2%,I%)=SWAP%(CVT$%(Dir.field$(31%,I%)))     &
                        FOR I%=0% TO 7%                                  &
        \ NEXT I2%                                                       &
        \ PRINT "The FDCM in block";I%;"differs from the FDCM in block";I1% &
                IF Mfd.fdcm%(I%-1%,I2%)<>Mfd.fdcm%(I1%-1%,I2%)           &
                        FOR I2%=0% TO 7% FOR I1%=I%+1% TO 4% FOR I%=1% TO 4% &
        \ RETURN                                                         &
                ! Subroutine to set up the pack information:             &
                ! Open the disk on file channel 1                        &
                ! GOSUB to field the buffer into 16 byte arrays          &
                ! Get the first block of the disk                        &
                ! Build the MFD label array                              &
                ! Initialize the pack cluster size variable              &
                ! Get the first MFD file directory cluster map           &
                ! Print some messages if the label is messed up          &
                ! Print the pack cluster size                            &
                ! Reopen the pack using a buffer large enough to hold a pack &
                !  cluster                                               &
                ! Get the first pack cluster                             &
                ! Loop through blocks 2,3, and 4 and get the FDCM's from each &
                !  of those blocks for comparison purposes               &
                !  (we expect a minimum pack cluster size of 4)          &
                ! Print a message if any of the FDCM's are different     &
                ! Return                                                 &
                                                                         &
                                                                         &

11000   PRINT TAB(10%);"1-Print the components of a specific link"       &
        \ PRINT TAB(10%);"2-Dump the contents of a specified blockette" &
        \ RETURN                                                         &
                ! Subroutine to print READ help message                  &
                ! Return                                                 &

11100   PRINT "The write option of this program is limited to the";      &
                " building of one blockette"                             &
        \ PRINT "at any given time.  The user must ";                    &
                "know, in advance, the entire contents of the"           &
        \ PRINT "blockette to be built, and must be able to identify the"; &
                " DCN, block offset,"                                    &
        \ PRINT "offset for the position off the blockette to be built." &
        \ PRINT                                                          &
        \ RETURN                                                         &
                ! Subroutine to print the WRITE help message             &
                ! Return                                                 &

11200   Clus.next%=(SWAP%(Link.next%) AND 14%)/2%                        &
        \ Block.next%=(SWAP%(Link.next%) AND 240%)/16%                   &
        \ Entry.next%=(Link.next% AND 496%)/16%                          &
        \ RETURN                                                         &
                ! Subroutine to break apart the next link:               &
                !       Clus.next% is the next UFD cluster needed        &
                !       Block.next% is the block offset within that cluster &
                !       Entry.next% is the entry offset within that block &
                ! Return                                                 &

11300   FIELD #1%,Temp%+(I%*16%+I1%+I1%) AS Temp$,                       &
                2% AS Dir.field$(I%,I1%) FOR I1%=0% TO 7% FOR I%=0% TO 31% &
        \ Temp%=0%                                                       &
        \ RETURN                                                         &
                ! Offset into the buffer Temp% blocks and field as 32,   &
                !  16-byte arrays                                        &
                ! Return                                                 &
                                                                         &
                                                                         &

19000   IF ERR=11 THEN                                                   &
                        RESUME 32700 IF ERL=1010                         &
                        \ RESUME 19999                                   &
                ! If the error is "End of file" then                     &
                !       Resume @ program end if we are looking for a task &
                !       Resume @ RETURN                                  &

19990   ON ERROR GOTO 0                                                  &
                ! Abort end all else                                     &

19999   RETURN                                                           &
                ! Convenient return for error handler                    &

32700   CLOSE I% FOR I%=1% TO 12%                                        &
                ! Prepare to end                                         &

32767   END                                                              &
                ! The end                                                &
```

DMS 500 USERS

DO YOU
LOVE basic plus?
LOVE multiple keys?
LOVE segmented keys?
HATE reorganizing files?
HATE compiling and tkbing?
HATE recoding old programs?

KEEP THE LOVE IN YOUR LIFE!!!!

from the Chainbreakers* who brought you **DBQ***
now **RMQ** lets you
Continue using Basic Plus
and have

- Multiple keys
- Segmented keys
- Automatic file extension (No more **REORG**anizing)
- Logical record locking
- Journaling
- Power fail protection
- Automatic logical record split into variables

AND

ALL OUR PRODUCTS PROVIDE
A PAINLESS plug compatible
path to **VAX/VMS**

* ***DATA BOSS 2 USERS***

DBQ provides online
ENTER – REMOVE – REPCHN
DIRECTLY from your Basic Program

See RSTS Pro. June/81 pg. 69

For more information call our computer
from a 300 baud (103/113 type coupler)
at (305) 634-7402

OR WRITE/CALL

SYSTEMS ASSOCIATES, INC.
P.O. Box 450429
Miami, Florida 33145

(305)447-2695
(305)634-8114

Basic Plus is a trademark of
Digital Equipment Corporation

Data Boss/2 is a licensed product of Florida Computer, Inc.
99 N.W. 183rd Street, Suite 230, North Miami, Fla. 33169

CIRCLE 16 ON READER CARD

FIP's Alignment Algorithm

By Paul R. Laba, Computer Services, Le Moyne College

Introduction

This article is the result of a Software Performance Report I recently submitted to DEC seeking clarification on the alignment algorithm used by FIP in the placement of files or UFDs on RSTS disks. It is assumed that the reader is familiar with the usual notations and definitions concerning RSTS disk directories; some brief definitions are presented here for completeness, although more detailed explanations can be found in several past articles of the RSTS PROFESSIONAL.

Block
256 contiguous words of data.

Cluster
A collection of one or more contiguous blocks. The number of blocks in a cluster is always a power of 2 in the range 1 — 256.

Disk Cluster (DC)
The primary storage unit on a RSTS disk. A DC is the smallest allocatable unit of disk data.

Disk Clustersize (DCS)
The number of blocks in each disk cluster. This value is based on the disk type (RM02, RL01, etc.), such that every disk cluster can be addressed with a single 16-bit word. This value is hardware dependent and cannot be altered.

Disk Cluster Number (DCN)
The address of each disk cluster on a disk pack, starting at 0 and increasing to the maximum disk cluster. DCN 0 always contains the disk's boot block; DCN 1 always addresses the start of the MFD (Master File Directory).

Pack Cluster (PC)
A collection of one or more contiguous disk clusters. The pack cluster serves as the minimum unit of storage for all data structures on a RSTS disk. Pack clusters can be thought of as logical extensions of disk clusters. The Storage Allocation Table (SAT) on a RSTS disk maps each pack cluster (not disk cluster!) with a single bit, indicating whether the pack cluster is allocated (1) or free (0).

Pack Clustersize (PCS)
The number of blocks assigned to each pack cluster, defined when a disk pack is initialized (via DSKINT). The pack clustersize must always be greater than or equal to the disk clustersize.

Pack Cluster Number (PCN)
The address of each pack cluster on a disk pack, starting at 0 and increasing to the maximum pack cluster. PCN 0 is aligned with DCN 1, with the effect that **no** pack cluster includes DCN 0 (unnecessary, since DCN 0 is always allocated).

File Cluster (FC)
A collection of one or more pack clusters. File clusters can be thought of as logical extensions of pack clusters, since they serve as the minimal storage unit for a specific file.

File Clustersize (FCS)
The number of blocks contained in a file cluster. This value is defined when the file is created, and must always be greater than or equal to the pack clustersize. The maximum clustersize for a file is 256.

The Alignment Algorithm

It is common knowledge that FIP aligns file clusters on pack cluster boundaries. This is necessary, since each bit in a disk's SAT represents a pack cluster. What is not commonly known is that FIP goes one step further in aligning file clusters; it aligns them on **file** cluster boundaries! The alignment algorithm, formally stated:

Let ROOT = starting DCN of a file cluster,
FCS = File Clustersize,
DCS = Disk Clustersize.

Then, for FCS > DCS,

ROOT modulo (FCS/DCS) = 1.

The algorithm restricts "where" on a disk a file's clusters may be placed. For example, a file with FCS = 8 must have all of its clusters aligned on a disk with DCS = 1, at DCNs in the set {1,9,17,25,...,8n + 1}; no cluster of the file can ever be placed at any DCN not in the set. If the FCS is 64, then the alignment set becomes {1,65,129,...,128n + 1}. For a disk with DCS = 4, a file with clustersize 64 will have its clusters aligned in the DCN set {1,17,33,...16n + 1}, where 16 = 64/4.

Less formally, the effect of the algorithm is for FIP to alter the pack clustersize to match the clustersize of a file

DEC*

TRADE IN and SAVE!

AMERICAN will credit your account with the full trade in price for your old DEC equipment to be used as payment for:

NEW DEC .... 11/23, 11/44, VAX 11-750 or 11-780
NEW DEC .... Disk or Tape Drives or other Peripherals
NEW DEC .... Terminals
USED DEC .... CPU's, memory, Peripherals or Terminals
DEC SOFTWARE .... DRS,MODEL,PAC I,II,ILS,PACS,P-STAT

All models of DEC 11, VAX, Peripherals & Terminals in Stock
Ready to Ship!

CALL *617-437-1100*
For Our Latest Listing of DEC CPUs & Peripherals.

AMERICAN USED COMPUTER
P.O. Box 68, Kenmore Station, Boston, Massachusetts 02215

Leaders in DEC Hardware Since 1968

*Registered Trademark of Digital Equipment Corporation

CIRCLE 17 ON READER CARD

being created or extended. For example, when a file with clustersize 256 is created or extended, FIP logically segments the disk into pack clusters of size 256 in order to align the file clusters. Note from the algorithm that the placement of a file's cluster is completely determined by the FCS and DCS. This is a significant change from the general notion that the pack clustersize plays the dominant role in FIP's determination of a file cluster's placement on a RSTS disk; instead the file clustersize itself plays a more important part. (Of course, the PCS defines the minimum FCS, so that the pack clustersize does have some control over a file cluster's alignment).

Files with clustersize equal to the disk clustersize can have their clusters placed at **any** available disk cluster. However, this is impossible for disks having a pack clustersize larger than the disk clustersize, since the minimum clustersize for a file is the pack clustersize.

Experimentation

In order to confirm this phenomenon, a small BASIC-PLUS program was written to scan a selected UFD, passing the DCNs of files with various clustersizes through the above algorithm. The file cluster alignment was confirmed in every case. Furthermore, the results were verified for the root DCN of all UFDs on the disk. (This is not surprising, since UFDs are nothing more than files with clustersize limited to 16).

While the alignment algorithm is itself interesting, the implications of the algorithm are quite surprising. Consider the following scenario.

A freshly DSKINTed RF-11 fixed-head disk shows the following attributes:

Disk Clustersize (DCS):	1
Pack Clustersize (PCS):	1
Allocated DCNs:	{0,1,5-9}
Non-allocated DCNs:	{2-4,10-1023}

Note that 1,014 contiguous blocks exist, starting at DCN 10. I now enter the following command to create a file:

```
PIP DF0:DUMMY.FIL/MO:16/CL:128/SI:896=NL:
```

My wish is to create a contiguous, 896-block file with clustersize 128. Since 1,014 contiguous blocks are available, the file creation should succeed. Instead, I get the dreaded ?NO ROOM FOR USER ON DEVICE message. The reason: since FIP must align the file clusters on a **file** cluster boundary, the file must begin at DCN 129, leaving only 895 contiguous, non-allocated blocks. If I re-enter the above command, changing the file clustersize to 64 (/CL:64), then the creation succeeds. Analysis shows that the file starts at DCN 65, with enough contiguous blocks to hold the entire file (DCNs 65-960).

Another implication of the algorithm appeared when I attempted to build a UFD on a non-file-structured disk (via my own utility). Concerned only with **pack** cluster alignment, I created the UFD, only to receive the message, ?BAD DIRECTORY FOR DEVICE when trying to read it later. Once the UFD was moved to a cluster that satisfied the alignment algorithm, the problem disappeared. Point of the story? FIP not only utilizes the algorithm in creating or extending files or UFDs; it expects and requires that such alignment is always maintained.

Rationale

While becoming more aware of the algorithm's effects, I was still puzzled by its purpose. Why should FIP **require** that all file clusters start on file cluster boundaries, rather than pack cluster boundaries? Clearly, disk fragmentation is not reduced by the algorithm. It can be argued that fragmentation actually increases (consider the above example on a large, almost-full disk pack). Furthermore, files thought to be "back-to-back" on a disk may actually have small "gaps" between them, where the second file could not be aligned on the pack cluster immediately following the first. DEC provided the not-so-obvious reason for the algorithm:

> "We always start files on **file** cluster boundaries to guarantee that the entire file cluster (or UFD cluster) will never cross a SAT block boundary."

Without the algorithm, files with clustersize greater than the pack clustersize (requiring two or more bits in the SAT per file cluster) could end up spanning a SAT block boundary in order to map a single file cluster. This would be undesirable, since it would increase SAT disk access overhead. DEC concludes their explanation saying, "We have no intention of changing this algorithm."

A Bug in FIP

The SPR submitted also pointed out a bug in FIP having to do with (you guessed it) the alignment algorithm for placed files. One of the features added to RSTS version 7.0 was the ability to place files starting at specified disk cluster numbers, via the /POSITION switch. (When will they provide us the ability to place UFDs?). Documentation on the matter indicates that a placed file will be located at or after the specified DCN, with a bit set in the file's directory marking it as placed (an "L" appears in directory listings). Continuing with the DF0:#disk scenario from above, I attempt to place a 16-block contiguous file at DCN 970, with a file clustersize of 16. The following command is issued:

```
PIP DF0:TEST.FIL/MO:16/CL:16/SI:16/PO:970=NL:
```

Since all DCNs from 961 through 1023 are non-allocated, the file creation succeeds. However, a directory listing shows the file's starting DCN as 961, and the file is **not** marked as placed. As expected, the alignment algorithm would not permit the file to be placed at DCN 970 (since 970 modulo 16 < > 1), but why not place the file at the next higher aligned DCN (977)? DEC described the bug this way:

> "The problem results from the fact that the routine that allocates the blocks out of the SAT rounds the requested DCN down to the PCN on the file cluster boundary. Later, when the file is created, the code checks to see if the DCN where the file was created is greater than or equal to the DCN requested by the user. Because of rounding, this test fails and the file is not marked as being placed."

In the above example, if any DCN from 961 to 976 had already been allocated, the file placement would have succeeded, with the file starting at DCN 977, and the "placed" bit being set.

DEC further noted that the problem is difficult to repair and would not be corrected until the next RSTS release (will there **ever** be a next release?). Note that, even with the bug, the placement of files still works, if you are willing to accept "works" to mean that files end up somewhere near (before or after) your specified DCN, and the "placed" bit may sometimes not get set. ♥

RSTS/E USERS

SYS CALL LIBRARY

FOR

RT-11 & RSX EMULATORS

$500/CPU

AVAILABLE 9 TRACK TAPE ONLY

CONTACT:

C. MUSUMECI
NATIONAL PUBLIC RADIO
2025 M ST., N.W.
WASHINGTON, D.C. 20036
202-822-2644

CIRCLE 19 ON READER CARD

LOOK at the "tear-out" cards in this issue.

There's a "HOT to COLD" form for rating our articles. Let us know what is most and least interesting to you.

There's a FREE gift for you. Bring in new subscribers and collect rewards. See "BOUNTY HUNTERS" card.

There's a READERS INQUIRY card for your convenience.

Thanks to a friend at the DECUS meeting.

DEC SALE

PDP11 CPU'S	PDP11 MEMORY	PDP11 OPTIONS
11/04-LC 11/34A-DE 11/34A-LE	MS11-LB MJ11-BE MK11-BE	DU11-DA FP11-A FP11-C

MSI MIDWEST SYSTEMS, INC.
1118 East Highway 13, Burnsville, Minnesota 55337
(612) 894-4020

©1981 MIDWEST SYSTEMS, INC. All rights reserved.
CIRCLE 18 ON READER CARD

RSTS PROFESSIONAL

Box 361 • Ft. Washington, PA 19034 • (215) 542-7008

□ PAYMENT ENCLOSED for one year's subscription (4 issues).
US 3rd class, $25 / Canada & US 1st class: $35
All other countries (air mail): $40, payable in US dollars.

□ BILL ME for one year's subscription:
□ US 3rd class / □ Canada or US 1st class / □ Other foreign.

Please send **Back Issues** circled:

□ $10 per issue enclosed.

□ Bill me for $12.50 per issue.

Send me a RSTS PRO Tee Shirt — $6.95
□S □M □L □XL

Vol. 1, #1	Vol. 2, #2	Vol. 2, #4	Vol. 3, #2
Vol. 2, #1	Vol. 2, #3	Vol. 3, #1	Vol. 3, #3

Name ____________________

Address ____________________

____________________ Suite ________

City ____________ State ________ Zip ________

Telephone (   ) ____________ Country ____________

FREE CLASSIFIED AD WITH SUBSCRIPTION!!
Your first 12 words are absolutely **FREE**, only $1.00 per word thereafter. Use the space provided below.

____

____

DEC USERS: Consider the Alternatives.

We're System Industries. And we've become the world's largest supplier of high-performance DEC*-compatible disk storage systems. For a lot of good reasons. Our broad product line includes data base hardware for almost every DEC minicomputer you can name—including the "supermini" VAX-11/780*.

You can buy your peripherals from DEC. Or you can buy them from us. But before you decide, consider the alternatives. Then, you'll realize there's only *one* viable alternative—*SYSTEM INDUSTRIES.*

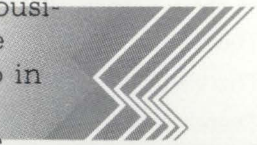
1 You get more reliability.

Reliability. We've always done much more than talk about it. We've delivered. Case in point, our Add-On, Non-Stop Reliability (ANSR) systems. They provide redundant data paths between each computer and each disk drive, so if a computer fails, users can continue to access data on any drive using an alternate data path.

Another reliable alternative we offer is our superior 160 Mbyte, SMD-compatible Winchester storage drives. These drives have already delivered more than 10,000 hours MTBF. That's twice the reliability that you'd get with an RM03.

We've been achieving optimum disk storage system reliability for 10 years. By constantly evaluating drive technology advances, developing new disk controllers, and supporting all such activities with proper user software, System Industries has earned its reputation for unmatched reliability.

2 You get faster delivery.



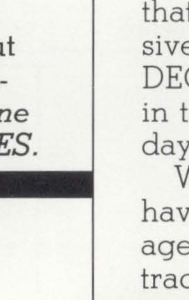
We've built our entire business around fast, effective service. And the first step in that procedure is responsive delivery. We provide DEC users with equivalent storage systems in typically 30 days ARO, instead of 180 days.

With System Industries, you'll never have to wait six months for the disk storage system you want. We'll deliver an attractive alternative—in a calendar month.

3 You save more money.

We've made it a habit to save DEC users big money on disk storage systems for more than a decade now.

For example, two RM03 storage units with a combined capacity of 160 Mbytes can cost you $50,000. While three of our 160 Mbyte Winchester disk storage drives in support of a DEC VAX-11/780 can give you 480 Mbytes for $42,500.

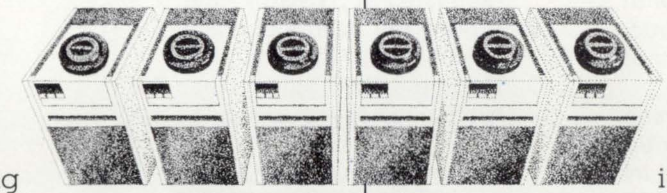


We can show you how to start from scratch and configure a high-performance disk storage system for your new CPU. Or how to add to your present disk storage. Or how to trade in your RPOX/RMOX drives for one of our superior disk systems. Whichever alternative you decide upon, you can always count on 100% software transparency, and big dollar savings.

4 You get a lot more flexibility.

Flexibility has been one of our key customer benefits during the past decade of supporting DEC users in the field. We've simply eliminated the problem of limited choice.

You tell us what you need. We'll supply the alternatives. And they can include such unique features as a data base shared by multiple CPU's for nonstop operation... or our reliable, ultra-flexible Winchester-SMD combos running off the same controller.



Our disk controllers are manufactured on a standard basis, but they are designed with flexibility that permits us to configure unique data storage systems that provide specific solutions to a wide variety of customer application requirements. You name it, we can probably come up with a highly versatile arrangement to take care of all your disk storage system needs.

5 You can save more floor space.

Our approach has always been to give our customers the greatest storage value in the smallest floor space possible.

For example, you might choose our fully loaded, 19-inch rack-mounted configuration housing our disk controller and three 160 Mbyte Winchester disk storage drives. This gives you 480 Mbytes of ultra-reliable storage in a space normally taken up by one 80 Mbyte RM03. To get 480 Mbytes of storage from DEC, you'd need six 80 Mbyte RM03s standing in a row.

Or you might want to select our newest Winchester-SMD combo that packs 900 Mbytes into the same floor space normally occupied by a 300 Mbyte RM04/05 from DEC.

6 You get stronger sales & service.

Uptime is an important part of our business. That's why we maintain service centers throughout the U.S. and Europe. We offer four-hour response to U.S. customers within a 50-mile radius of nine major cities, and to European customers contacting our wholly-owned subsidiary near London.

We've delivered and installed 15,000 systems over the past 10 years. Our service organization has doubled in size every year since 1977. And we cover all major metropolitan areas with 24 hours/day, seven days/week contract agreements, providing on-site maintenance as a standard option. Our field service engineers are experts at maintaining the complex data base systems we attach to DEC computers.

For more information about the *only* viable alternative to DEC data storage systems, contact the System Industries office nearest you.

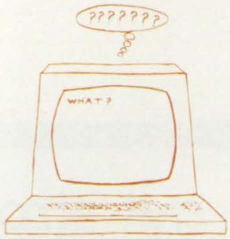
*DEC and VAX-11/780 are registered trademarks of Digital Equipment Corporation.

System Industries

United States: 525 Oakmead Parkway, P.O. Box 425, Sunnyvale, CA 94086, (408) 732-1650, Telex 346-459.
Europe: System Industries (Europe), System House, Guildford Road, Woking, Surrey, GU22 7QQ, England, (048 62) 5077, Telex 859124.
California (714) 754-6555, (213) 557-0384; **Colorado** (303) 986-1591; **Georgia** (404) 955-2252; **Illinois** (312) 948-9330; **Massachusetts** (617) 695-4022; **New Jersey** (201) 839-8650; **New York** (516) 482-6082; **New York Metro** (212) 785-9300; **Ohio** (513) 771-0075; **Seattle** (206) 451-8791; **Texas** (713) 497-7224; **Washington, D.C.** (703) 734-9700, **West Germany** (06102) 5464/5; **Sweden** 08-63 62 74

CIRCLE 20 ON READER CARD

DEAR RSTS MAN:



DEAR RSTS MAN:

What solutions have other users found to make disk garbage unreadable? Our students sometimes erase a file without meaning to and then search around through disk garbage until they find it and then pull it back into their account. Last week, while looking for a paper he had erased by accident, a student stumbled across an instructor's grade book and pulled that along with his paper into his account. The instructor whose grade book it was, was not exactly thrilled.

Cordially, Gordon G. Henderson
Director, Computing Center
Earlham College, Richmond IN

Dear Mr. Henderson: *Confer RSTS Programming Manual, p. 1-8. Modify your grade book program & other sensitive programs to use their normal protection codes +128. Other folks, not willing to endure the FIP overhead incurred zeroing files, use batch programs to open & zero files until they run out of space, then delete them all. Others suffer.*

—

DEAR RSTS MAN:

How does PIP.SAV work so quickly?

K. Wallewein
Calgary, AL 12K5B3

Dear K: *PIP.SAV is written in MACRO (RT11 version). That makes it about seven to ten times faster than B+2 and 20 to 100 times faster than B+ (on average). It also avoids SYS calls for directory lookup & uses huge buffers (if you let it) and swaps buffers when copying. The problem with writing in MACRO is that the difficulty of producing debugged code is directly proportional to the relative speed enhancements noted above.*

—

DEAR RSTS MAN:

I attach a solution to the problem raised by Wilf Forrow (Vol. 3, No. 1, p. 15). I hope this may be of use to your readers. I have not used it with an LA120 but it works with similar devices.

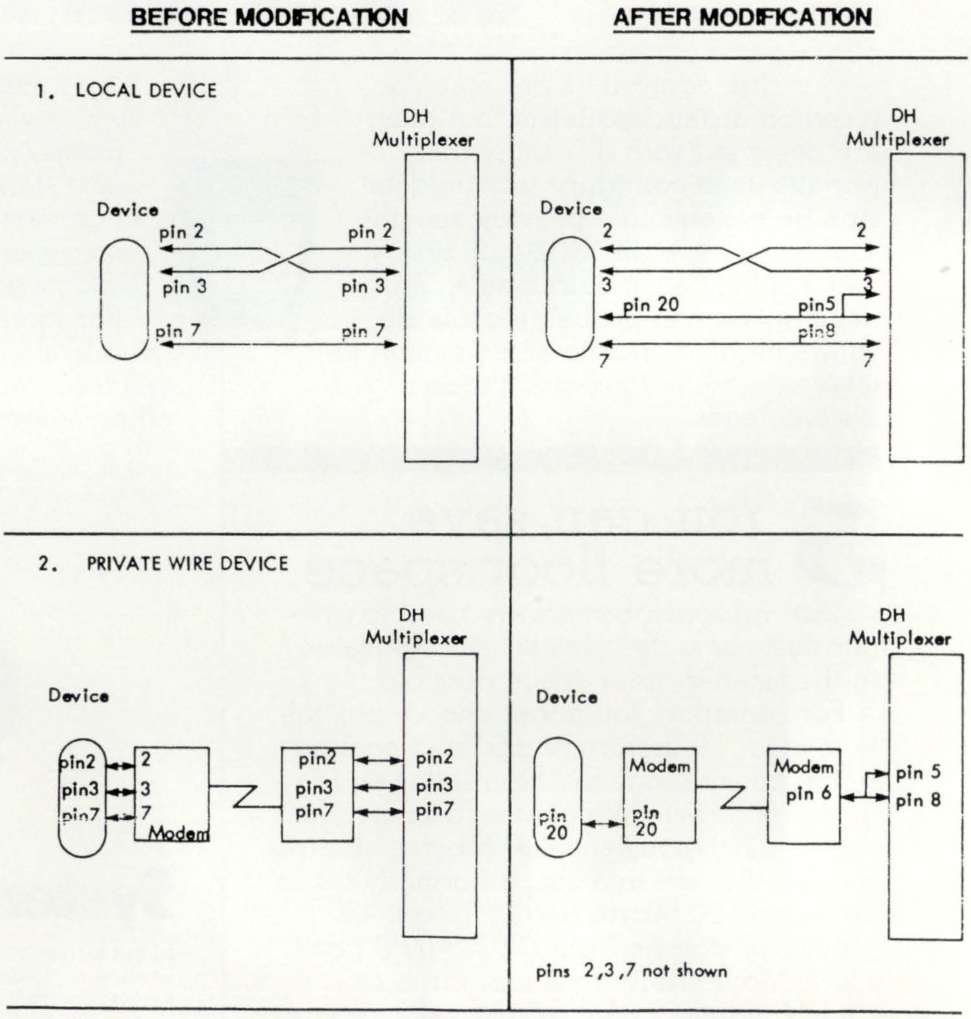
1. *Problem:* When outputting to a terminal device RSTS does not check whether the device is actually present and on-line. Data can be lost down a hole.
2. *Solution*

The RSTS/E operating system detects and acts on device status only for modem configured terminal devices.

A local device, or remote device using private wires must be made to look like a modem configured device. Then, the operating system must be informed that the device is "remote". The device itself must be made to set and unset the modem control signals via its V24 interface connection. So the steps taken are as follows:

A. Ensure that Modem Support is requested at SYSGEN time.
B. Rewire the device connections as shown in Figure 1.
C. Using the 'SET' option when the system is booted up change the relevant devices from "LOCAL" to "MODEM". This means RSTS will respond to the modem control signals.
D. Using TTYSET set the device characteristics, especially speed, using the /RING option. This means that every time the device fails, RSTS effectively "hangs the line up" and sets the device characteristics back to those set in the /RING option. If this option is not used the device is always set back to 110 baud.
E. When printing to the device, always open it in MODE 16% to prevent hibernation on device failure. Error 27 (I/O to detached keyboard) will always occur when the device is unavailable.

Yours faithfully, George T.A. May
Special Systems Division
Software Sciences Limited, UK



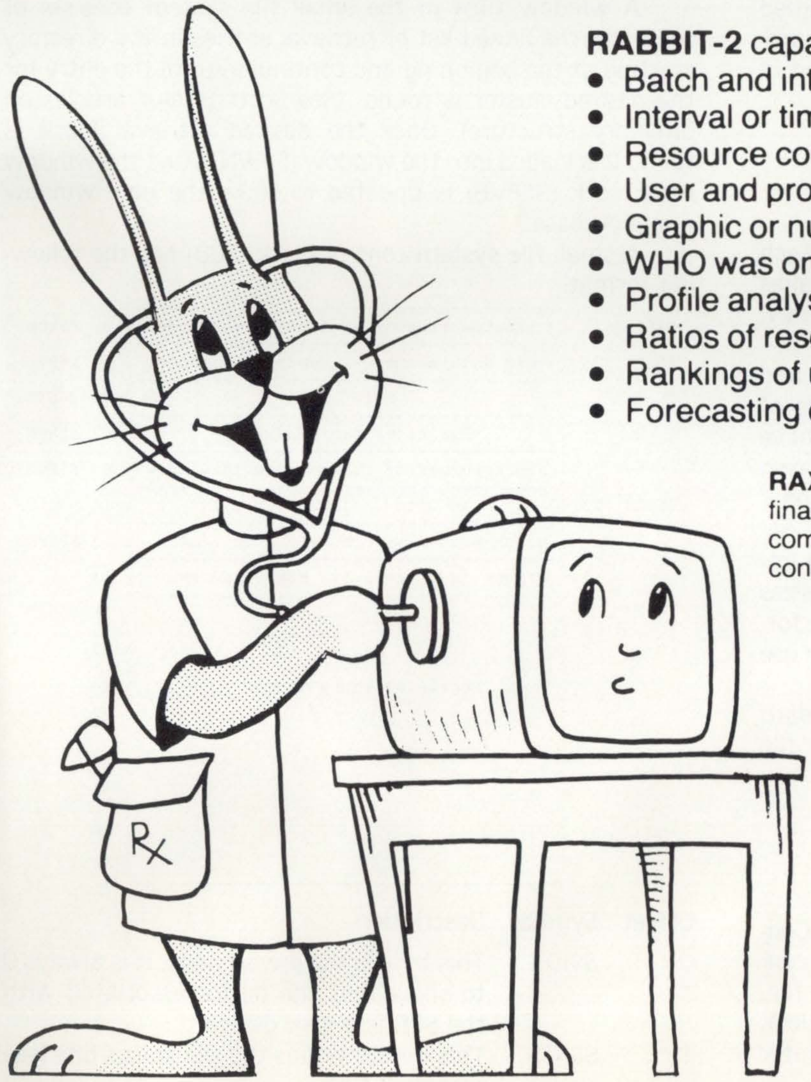
pin 2	TRANSMIT	pin 7	SIGNAL GROUND
pin 3	RECEIVE	pin 8	CARRIER DETECT
pin 5	CLEAR TO SEND	pin 20	DATA TERMINAL READY
pin 6	DATA SET READY		

SYSTEM PERFORMANCE ANALYSIS FOR VAX AND RSTS/E USERS

If your system is suffering from slow response, clogged I/O, reduced throughput, then put **RAXCO's 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.



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

RAXCO markets a complete line of operational support, financial planning and data management systems for DEC computing equipment. For a free catalog of these systems contact:

RAXCO INC.

3336 N. Flagler Drive
West Palm Beach,
Florida 33407
(305) 842-2115

CIRCLE 21 ON READER CARD

RSTS/E MONITOR INTERNALS Part 3

By Mike Mayfield, Northwest Digital Software, Box 2-743, Newport, WA 99156

The demand for information on RSTS internals has been so great that I have decided to write a complete book on RSTS monitor internals. It will include not only monitor tables but disk structures, writing device drivers, runtime systems, and resident libraries, FIP, file attributes and performance optimization. Look for more information on it here in the RSTS Pro.

This is the third in a series of four articles describing the internal structure of the RSTS/E V7.0 monitor tables. The articles to date have described job and memory control. This article will describe the structures involved in the device and file control. Both the large and small file systems will be discussed.

After reading this article you may want to go back and take another look at Francois Dubois' article entitled "JBSTAT" in Volume 3, Number 2 of the RSTS Pro. His program makes use of much of the information presented in this article.

3.0 FILE CONTROL

A disk is typically broken down into several files. Each file can be treated as if it were a separate disk. RSTS takes care of finding a place to put the file data and then retrieving that information when needed.

Since most RSTS systems are used in commercial and general purpose applications which make frequent use of disk files, improvements in the file control system can have a significant impact on system performance. For this reason DEC essentially rewrote the file control system for version 7 of RSTS.

The result was the large file system. As the name implies this new file control system allows large file sizes (larger than 65,535 blocks). It also includes many performance improvements, especially when the same file is in use by more than one person.

For purposes of this discussion we will call the standard file control system the small file system and the new file control system the large file system.

3.1 SCB — SMALL FILE CONTROL BLOCK

Each time a file is opened a structure called a Small Control Block (SCB) is allocated to the opener. This stucture contains all the information needed to access the opened file. This information includes the file size, current block number, pointers to seven clusters of the file and a pointer to the remaining cluster information in the directory.

When a file is accessed the file size (S$SIZ) is checked against the desired block number. If the desired block number is greater than the current file size an error is returned for GETs and an automatic file extension is done for PUTs (except in the case of contiguous files and update mode).

The window base block number (S$FVB) and the window information (S$WND) are used to map the logical block number supplied by the user into a physical block number on the disk. A window holds pointers to seven clusters of the file. If the desired logical block is contained in the seven clusters pointed to by the current window the physical block can be determined immediately. If it is not contained in the current window a window turn is performed.

A window turn in the small file system consists of following the linked list of retrieval entries in the directory starting at the beginning and continuing until the entry for the desired cluster is found. (See Scott Banks' articles on directory structure). Once the desired retrieval block is found it is loaded into the window (S$WND) and the window base block (S$FVB) is updated to show the new window mapping base.

A small file system control block (SCB) has the following format:

```
              ---------------------------------------
S$STS  1  |   Status flags    | Disk driver index |  0  S$IDX
          |---------------------------------------|
S$UNT  3  |  FIP unit number  | Pending transfers |  2  S$PT
          |---------------------------------------|
          |               File size               |  4  S$SIZ
          |---------------------------------------|
          |        Next block to read/write       |  6  S$NVB
          |---------------------------------------|
          | Block number of first block in window |  8  S$FVB
          |---------------------------------------|
S$FLG 13  |      Flag bits    | Cluster size - 1  | 10  S$CLUS
          |---------------------------------------|
          | Block number of UFD name entry (LSB)  | 12  S$UFND
          |---------------------------------------|
          | Offset in block/2 |   MSB of S$UFND   | 14
          |---------------------------------------|
          |                                       | 16  S$WND
          |                                       |
          |                                       | 18
          |                                       |
          |        Retrieval entry window         | 20
          |                                       |
          |                                       | 22
          |                                       |
          |                                       | 24
          |                                       |
          |                                       | 26
          |                                       |
          |                                       | 28
          |                                       |
          |                                       | 30
              ---------------------------------------
```

Offset	Symbol	Description
0	S$IDX	This byte is the driver index. It is always 0 to show that the device associated with the SCB is a disk device.
1	S$STS	This byte contains the file status bits (see section 3.1.1).

2 S$PT This byte contains a count of the number of transfers pending on this file by this user. When an I/O is requested this byte is set to 1. If the request requires more than one physical I/O transfer, this value is increased as necessary. An I/O request can require more than one transfer if it involves more than one block and the blocks either cross a cluster boundary or cross a cylinder boundary on a disk that doesn't do automatic cylinder movement (such as RL01/02).

3 S$UNT This byte contains the FIP unit number for the disk in use. Each disk unit on the system has a unique FIP unit number associated with it.

4 S$SIZ This word contains the current file size, in blocks. The file size may range from 0 to 65,535 blocks.

6 S$NVB This word contains the next block to read or write if a sequential I/O transfer is requested.

8 S$FVB This word contains the block number of the first block in the retrieval window (S$WND). The block number corresponds to the first block of the first cluster in the window. Knowing this block number and the file's cluster size you can determine which blocks are mapped by the current window.

10 S$CLUS This byte contains the file's cluster size minus one.

11 S$FLG This byte contains file status bits and locked block information (see 3.1.2).

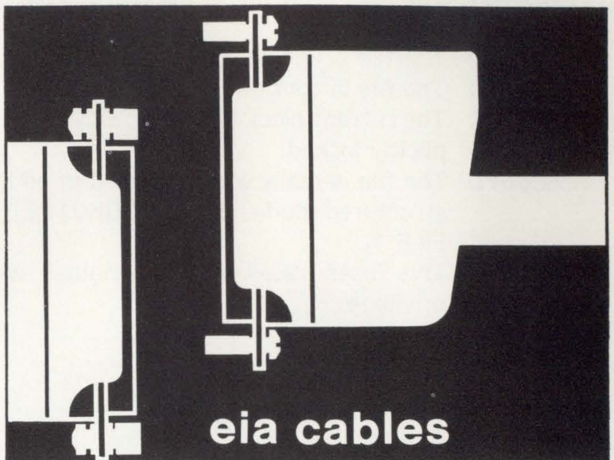
12 S$UFND These two words contain a pointer to the name entry in the UFD for this file. The first two bytes are the low order bytes of the block number of the UFD. The next byte is the high order byte of the block number. The last byte is the offset, divided by 2, into the block to the name entry.

16 S$WND The first word contains status bits (see section 3.1.3). The following seven words are the current retrieval window. Each word contains the device cluster number (DCN) of the first block of the associated cluster.

3.1.1 S$STS — Status Flags

The status flags show the current status and restrictions placed on the file in use. These bits correspond to conditions in effect at the time the file was opened (such as write protection) and conditions in effect as of the last I/O. The format of S$STS is as follows:

Bit	Symbol	Description
8	DDNFS	The disk is opened non-file structured. The remaining status bits apply as if the disk were opened file structured.
9	DDRLO	The file is read protected against the user.


eia cables

- Standard and extended length cables.
- Highest quality materials.
- Stringent quality control.
- Stocked lengths for immediate delivery.
- Custom lengths to order.
- Coax/twinaxial cables also available.
- Quantity pricing and fast delivery.

wireworks
WIREWORKS CORPORATION
Data Communications Division
380 Hillside Avenue, PO Box 3600
Hillside, New Jersey 07205
Tel: (201) 686-7400
Twx: 710-985-4675

CIRCLE 22 ON READER CARD

REPRINT POLICY

All content in this publication is copyrighted material.

All reprints must be purchased from M Systems, Inc. No other reprints are authorized.

All reprints shall contain both a cover and a subscription blank.

Price quotations available on request.

A RSTS INTERNALS MANUAL

by Mike Mayfield

More than 200 pages in a binder.

Published by
The RSTS PROFESSIONAL

Pre-publication Price
$75-includes binder

Order by December 15, 1981
to

M SYSTEMS
Box 361
Fort Washington, PA 19034-0361

10	DDWLO	The file is write protected against the user.
11	SC$UPD	The file is opened for update (mode 1).
12	SC$CTG	The file is contiguous.
13	SC$LCK	The current block (shown in S$NVB) is implicitly locked.
14	SC$UFD	The file is really a UFD opened in non-file structured mode (ie. OPEN "DK0:[1,2]" AS FILE 1).
15	SC$USE	This user received the original write privileges.

3.1.2 S$FLG — Flag Bits

S$FLG serves two purposes. First, it defines the number of blocks that are included in an implicit lock. Second, it contains status bits similar in purpose to S$STS. The format of S$FLG is as follows:

Bit	Symbol	Description
0-4	SC$LLK	These five bits specify the number of blocks that are included in the current implicit lock. This value may vary from 0 to 31.
5	SC$RR	The file is opened in read regardless mode (mode 4096).
6	SC$EXT	This SCB is followed immediately in memory by another small buffer's worth of information about explicitly locked blocks. The information in this extended portion of the SCB consists of a table of double-words where the first byte is the most significant byte of the block number of the first explicitly locked block, the second byte is the number of blocks included in the lock, the third and fourth bytes (as a word) are the least significant bytes of the block number of the first locked block.
7	SC$DLW	The file was either written into or extended. The file size and date last written need to be updated when the file is closed.

3.1.3 S$WND — Retrieval Window Flags

The first word of the retrieval window contains flag bits used for new version 7 features. The format of S$WND is as follows:

Bit	Symbol	Description
0	SC$SPU	The file is open in special update mode (mode 5).
1	SC$AEX	Always do a real extend on the file (mode 8).
2	SC$CHE	The file is open for user data caching (mode 256 or mode 2048)
3	SC$CSQ	The user data caching specified by SC$CHE is sequential (mode 2048).

3.2 FCB — LARGE FILE CONTROL BLOCK

With version 7 of RSTS came the optional large file support. If this option is taken at sysgen time the standard file processing code in FIP is replaced with a new improved version. This new file processor uses totally different control structures which allow many improvements in performance, especially when the same file is opened by more than one user.

With the new file system control structures, less disk overhead is required to open a file for the first user and no disk overhead is required for subsequent users of the same file. In addition, disk overhead associated with window turns (loading the retrieval window from the directory) is reduced, especially when the file is open more than once.

The structures used in the new file system are the File Control Block (FCB) and the Window Control Block (WCB). When a file is opened a list of FCBs for currently opened files is searched. If the desired file is not found in this list the directory is searched to find the information for the file. This information is loaded into an FCB and a WCB.

When the same file is opened simultaneously on another channel, or by another user, the information about the file is already in an FCB. No search of the directory, with its attendant disk accesses, is needed. All the information about the file is already contained in the existing FCB. A new WCB is allocated for the open and linked to the previous WCB or list of WCBs for this file. No disk accesses are required and FIP is released immediately.

As you can see, the overhead of opening a file when someone else already has it open is drastically reduced. This can be used to our advantage by writing a small program that opens the most commonly used files in read/regardless mode and then detaches and goes to sleep indefinitely. Both data files and commonly used programs can be opened in this way. Once opened, there will essentially be no overhead the next time any of these files are opened.

To allow the existing FCBs to be searched quickly they are kept in a linked list. Each disk unit has its own linked list. The base address of each list is kept in a table (FCBLST) and is accessed by the FIP unit number (FUN), times two, for the associated disk unit.

The window control blocks (WCBs) associated with each file are also kept in a linked list. The pointer to the base of this list is kept in the FCB for the file. The last WCB in the linked list contains a pointer back to the FCB.

The following diagram shows the linked list of FCBs and WCBs:

```
      FCBLST
     ---------
---|         |
|  |---------|
|  |    .    |   Indexed by FUN*2
|  |    .    |
|  |         |
|   ---------
|
|       FCB              FCB               FCB
|    ---------        ---------         ---------
--->|         | --->|         | - - ->|         |
    | File A  |     | File B  |       | File x  |
    |         |     |         |       |         |
     ---------        ---------         ---------
        |   ^           |    ^            |    ^
        |   |           |    |            |    |
        |   |           |    |            |     ------------
        v   |           v    |            v                |
     ---------        ---------        ---------      ---------
    |         |     |         |      |         |    |         |
    |   WCB   |     |   WCB   |      |   WCB   |---->|   WCB   |
    |         |     |         |      |         |    |         |
     ---------        ---------        ---------      ---------
```

RSTS/E programs: the long...

The format of a File Control Block (FCB) is as follows:

```
               --------------------------------------
               |  Link to next FCB on this FIP unit  | 0  F$LINK
               |-------------------------------------|
               | File ID (link to name entry in UFD) | 2  F$FID
               |-------------------------------------|
               |  PPN project #   | PPN programmer # | 4  F$PPN
               |-------------------------------------|
               |                                     | 6  F$NAM
               |         File name in RAD50          |
               |                                     | 8
               |-------------------------------------|
               |    File name extension in RAD50     | 10
               |-------------------------------------|
F$PROT 13      | Protection code  |    Status byte   | 12 F$STAT
               |-------------------------------------|
F$RCNT 15      | RR access count  | N/U access count | 14 F$ACNT
               |-------------------------------------|
               |                                     | 16 F$WFND
               |     FBB of first retrieval entry    |
               |                                     | 18
               |-------------------------------------|
               |                                     | 20 F$UFND
               |          FBB of name entry          |
               |                                     | 22
               |-------------------------------------|
F$SIZM 25      | File size (MSB)  | FIP unit number  | 24 F$UNT
               |-------------------------------------|
               |           File size (LSB)           | 26 F$SIZL
               |-------------------------------------|
               |          File cluster size          | 28 F$CLUS
               |-------------------------------------|
               | Pointer to first WCB for this file  | 30 F$WCB
               --------------------------------------
```

Offset	Symbol	Description
0	F$LINK	This word contains a pointer to the next FCB on this FIP unit. The end of the list is terminated by a forward link of 0.
2	F$FID	This word contains the file ID for this file. It is used when searching the FCB list during an open by file ID (only available in MACRO).
4	F$PPN	This word contains the PPN of the file. The programmer number is in the first byte, the project number in the second byte.
6	F$NAM	These three words contain the file name and extension in RAD50. The file name is in the first two words, the extension in the third word.
12	F$STAT	This byte contains file status bits (see section 3.2.1).
13	F$PROT	This byte contains the file's protection code.
14	F$ACNT	This byte contains a count of the number of times this file is currently open for normal or update access.
15	F$RCNT	This byte contains a count of the number of times this file is currently open for read/regardless access.
16	F$WFND	This double word is the FIP block/sub-block (FBB) of the first retrieval entry for this file in the directory. FBBs are described in section 3.2.2.
20	F$UFND	This double word is the FIP block/sub-block (FBB) of the name entry for this file in the directory. FBBs are described in section 3.2.2.
24	F$UNT	This byte contains the FIP unit number for the disk unit containing this file.
25	F$SIZM	This byte contains the most significant byte of the file size.
26	F$SIZL	This word contains the least significant bytes of the file size.
28	F$CLUS	This word contains the file's cluster size.
30	F$WCB	This word contains a pointer to the first window control block (WCB) for this file.

3.2.1 F$STAT — Status Flags

The status flags contained in F$STAT have the following for mat:

Bit	Symbol	Description
0	US.OUT	This file is a swapping disk that is not included in the SAT.
1	US.PLC	This file is placed starting at a specific disk block.
2	US.WRT	Write access has already been given out.
3	US.UPD	This file is open in update mode.
4	US.NOX	This file is contiguous. No extends are allowed.
5	US.NOK	This file may not be deleted or renamed.
6	US.UFD	This file is really a UFD opened in non-file structured mode (i.e.; OPEN "DKO:[1,2]"AS FILE 1).
7	US.DEL	This file has been deleted.

3.2.2 FBB — FIP Block/Sub-block

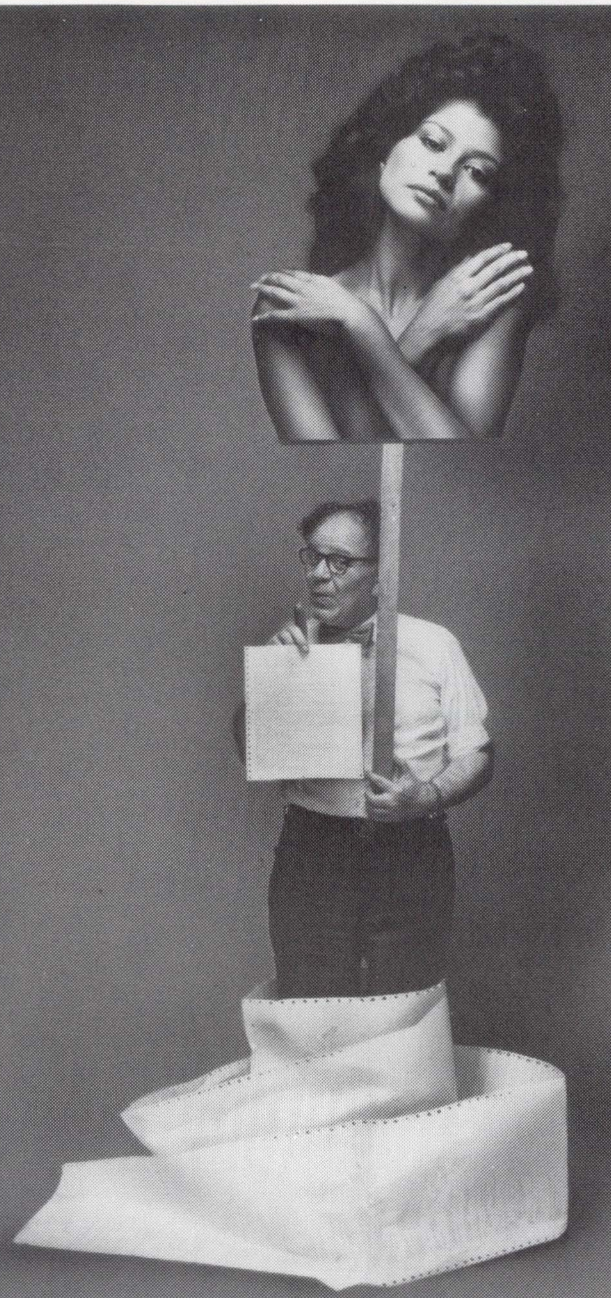
A FIP block/sub-block provides all the information necessary to immediately retrieve any entry in a directory if the FIP unit number is known. Given this information a directory entry can be accessed in one seek. The format of an FBB is as follows:

Offset	Description
0	This byte contains the offset within the directory block, divided by 2, to the desired directory entry.
1	This byte contains the block offset within the cluster to the desired block of the directory.
2	This word contains the device cluster number of the first block of the desired cluster.

3.3 WCB — WINDOW CONTROL BLOCK

The window control block (WCB) contains all the information needed by the monitor for an individual user of a file. Each simultaneous opening of a file generates a unique WCB. These WCBs are pointed to by the user's IOB (see section 1.4, Volume 3, Number 1).

When an I/O request is made for a file, the WCB associated with the specified user and channel number is used to access the retrieval window. If the desired retrieval information is not already present in the WCB a window turn is performed. Unlike the small file system, the large file system will start a search at the current position if the desired retrieval information is beyond the current position in the linked list of retrieval entries.



...and short of it.

Data Node is proud to announce that RSTS/E programs will now be quicker to write, faster to run and they won't hog your system.

With the installation of our Network Processing add-ons, you can expect CPU time, core memory and code reductions of up to 10 to 1.

Write us for details - the long and short ones.

data node

432 Toyama Drive, Sunnyvale, California 94086 • (408) 744-0561

CIRCLE 24 ON READER CARD

WHEN YOU NEED
DEC...

TERMINALS
• VT-100 • VT-103
• LA34 • LA120
INTERFACES
MODEMS
PERIPHERALS
LSI/11 MODULES
PDP 11/03
PDP 11/23
SYSTEMS
Standard & Custom
Software Packages
WORD PROCESSING

UNITRONIX
CORPORATION
(201) 231-9400
197 Meister Ave.
Somerville, NJ 08876
TELEX: 833184

CIRCLE 25 ON READER CARD

Turn your PDP-11 into a powerful word processing tool.

WP SATURN

With the WP Saturn you can add powerful word processing to your PDP-11, LSI-11, or Vax System, without the cost of a standalone system.

WP runs on RT-11/TSX, RSX, RSTS, IAS, VMS.

WP's ASCII Format interfaces to standard DEC files.

Assembler Language provides maximum speed and efficiency.

User Friendly Prompting helps you master word processing with minimal training.

Extensive List Processing: Forms, prompted data entry and sort/select; plus flexible merging of lists and documents.

For Information Call . . .

800-328-6145

Saturn Systems, Inc., 6875 Washington Avenue South, Suite #218, Minneapolis, MN 55435, **(612) 944-2452.**

CIRCLE 26 ON READER CARD

As the list of retreival entries is followed, the linked list of WCBs for this file is checked. If the desired retreival entry is already contained in a WCB, that WCB is used instead of following the link on disk. When the desired retrieval entry is found the information is copied into the current WCB's retrieval entry window.

The format of a window control block is as follows:

```
             ----------------------------------------
W$STS  1   |   Status flags    | Disk driver index |  0  W$IDX
           |---------------------------------------|
W$FLAG 3   |     Flag bits     |    Job number *2  |  2  W$JBNO
           |---------------------------------------|
W$NVBM 5   | Next block # (MSB)| Pending transfers |  4  W$PT
           |---------------------------------------|
           |       Next block number (LSB)         |  6  W$NVBL
           |---------------------------------------|
           |      Pointer to FCB at F$CLUS         |  8  W$FCB
           |---------------------------------------|
           |       Retrieval entry number          |  10 W$REN
           |---------------------------------------|
           |  Pointer to next WCB for this FCB     |  12 W$WCB
           |---------------------------------------|
           |                                       |  14 W$NXT
           |          FBB of next window           |
           |                                       |  16
           |---------------------------------------|
           |                                       |  18 W$WND
           |                                       |
           |                                       |  20
           |                                       |
           |        Retrieval entry window         |  22
           |                                       |
           |                                       |  24
           |                                       |
           |                                       |  26
           |                                       |
           |                                       |  28
           |                                       |
           |                                       |  30
           -----------------------------------------
```

Offset	Symbol	Description
0	W$IDX	This byte is the driver index. It is always 0 to show that the device associated with the WCB is a disk device.
1	W$STS	This byte contains the file status bits for this user (see section 3.3.1).
2	W$JBNO	This byte contains the job number, times 2, of the user that owns this WCB.
3	W$FLAG	This byte contains file status bits and locked block information (see section 3.3.2).
4	W$PT	This byte contains a count of the number of transfers pending on this file by this user. When an I/O is requested this byte is set to 1. If the request requires more than one physical I/O transfer this value is increased as necessary. An I/O request can require more than one transfer if it involves more than one block and the blocks either cross a cluster boundary or cross a cylinder boundary on a disk that doesn't do automatic cylinder movement (such as RL01/02).
5	W$NVBM	This byte contains the most significant byte of the next block number to use for sequential access.
6	W$NVBL	This word contains the least significant bytes of the next block number to use for sequential access.

8	W$FCB	This word contains a pointer to the file control block (FCB) at offset F$CLUS in the FCB.
10	W$REN	This word contains the current retrieval entry number. It identifies which retrieval entry is currently stored in W$WND of the WCB.
12	W$WCB	The low order bits of this word are file status bits. The high order bits are a pointer to the next WCB for this FCB. See section 3.3.3 for more information.
14	W$NXT	This double-word is the FIP block/sub-block (FBB) for the next retrieval entry in the directory. It allows immediate access to the next retrieval entry in the directory if sequential access is taking place.
18	W$WND	The following seven words are the current retrieval window. They contain the device cluster number of the first block of each cluster for this retrieval entry.

3.3.1 W$STS — Status Flags

The status flags in W$STS show the status and restrictions placed on the file for the current user. Bear in mind that each time a file is simultaneously opened a new WCB is allocated and each WCB has its own set of status flags. The format of W$STS is as follows:

Bit	Symbol	Description
8	DDNFS	The disk is opened non-file structured. The remaining status bits apply as if the disk were opened file structured.
9	DDRLO	The file is read protected against the user.
10	DDWLO	The file is write protected against the user.
11	WC$UPD	The file is open for update (mode 1).
12	WC$CTG	The file is contiguous.
13	WC$LCK	The current block of the file is implicitly locked.
14	WC$UFD	The file is really a UFD opened in non-file structured mode (i.e.; OPEN "DKO:[1,2]"AS FILE 1).
15	WC$USE	This WCB received the original write privileges.

3.3.2 W$FLAG — Flag Bits

W$FLG serves two purposes. First, it defines the number of blocks that are included in an implicit lock. Second, it contains status bits similar in purpose to W$STS. The format of W$FLG is as follows:

Bit	Symbol	Description
0-4	WC$LLK	These five bits specify the number of blocks that are included in the current implicit lock. This value may vary from 0 to 31.

Hardware Accessories For DEC Equipment Users

C-XX Overtemperature Protection System

Standard DEC PDP 11, VAX, and System 10-20 machines are NOT adequately protected from equipment damage due to high machine room temperatures. This unit provides aural warning signal and total system power shutdown with two customer adjustable temperature limits and approved interface to standard DEC AC power control system.

B-11 CPU Speedup for PDP 11/45 and 70

This timing generator modification allows 11/45 or 11/70 CPU instruction throughput rate to be increased by up to 15%.

Both products are attractively priced, install in a few minutes with no special tools or skills, and are allowed by DEC Field Service in contract systems. Call or write for further information.

Nassau Systems
P.O. Box 19329
Cincinnati, Ohio 45219
(513) 231-1283

DEC, VAX, and PDP are trademarks of Digital Equipment Corp.

CIRCLE 27 ON READER CARD

TERMINALS FROM TRANSNET

PURCHASE PLAN • 12-24 MONTH FULL OWNERSHIP PLAN • 36 MONTH LEASE PLAN

		PURCHASE	PER MONTH		
	DESCRIPTION	PRICE	12 MOS.	24 MOS.	36 MOS.
DEC	LA36 DECwriter II	$1,095	$105	$ 58	$ 40
	LA34 DECwriter IV	995	95	53	36
	LA34 DECwriter IV Forms Ctrl.	1,095	105	58	40
	LA120 DECwriter III KSR	2,295	220	122	83
	LA120 DECwriter III RO	2,095	200	112	75
	VT100 CRT DECscope	1,595	153	85	58
	VT132 CRT DECscope	1,995	190	106	72
TEXAS INSTRUMENTS	TI745 Portable Terminal	1,595	153	85	58
	TI765 Bubble Memory Terminal	2,595	249	138	93
	TI Insight 10 Terminal	945	90	53	34
	TI785 Portable KSR, 120 CPS	2,395	230	128	86
	TI787 Portable KSR, 120 CPS	2,845	273	152	102
	TI810 RO Printer	1,895	182	102	69
	TI820 KSR Printer	2,195	211	117	80
DATAMEDIA	DT80/1 CRT Terminal	1,695	162	90	61
	DT80/3 CRT Terminal	1,295	125	70	48
	DT80/5L APL 15" CRT	2,295	220	122	83
LEAR SIEGLER	ADM3A CRT Terminal	875	84	47	32
	ADM31CRT Terminal	1,450	139	78	53
	ADM42 CRT Terminal	2,195	211	117	79
HAZELTINE	1420 CRT Terminal	945	91	51	34
	1500 CRT Terminal	1,095	105	58	40
	1552 CRT Terminal	1,295	125	70	48
TELEVIDEO	920 CRT Terminal	895	86	48	32
	950 CRT Terminal	1,075	103	57	39
NEC SPINWRITER	Letter Quality, 55/15 RO	2,895	278	154	104
	Letter Quality, 55/25 KSR	3,295	316	175	119
QUME	Letter Quality KSR, 55 CPS	3,395	326	181	123
	Letter Quality RO, 55 CPS	2,895	278	154	104
CENTRONICS	730 Desk Top Printer	715	69	39	26
	737 W/P Desk Top Printer	895	86	48	32

FULL OWNERSHIP AFTER 12 OR 24 MONTHS • 10% PURCHASE OPTION AFTER 36 MONTHS

MICROCOMPUTERS
APPLE • COMMODORE • HP85 • DEC LSI 11

ACCESSORIES AND PERIPHERAL EQUIPMENT
ACOUSTIC COUPLERS • MODEMS • THERMAL PAPER • RIBBONS • INTERFACE MODULES • FLOPPY DISK UNITS

TRANSNET CORPORATION
1945 ROUTE 22 • UNION, N.J. 07083 • (201) 688-7800
TWX 710-985-5485

CIRCLE 28 ON READER CARD

5 WC$EXT This WCB is followed immediately in memory by another small buffer's worth of information about explicitly locked blocks. The information in this extended portion of the WCB consists of a table of double-words where the first byte is the most significant byte of the block number of the first explicitly locked block, the second byte is the number of blocks included in the lock, the third and fourth bytes (as a word) are the least significant bytes of the block number of the first locked block.

6 WC$DLW The file was either written into or extended. The file size and date last written need to be updated when the file is closed.

7 WC$NFC The file is opened non-file structured in cluster mode (mode 128 was not used).

3.3.3 W$WCB — File Flags and Link to WCB

Since the small buffers from which WCBs are allocated always lay on even boundaries of 32 bytes, the low order five bits of their address are always 0. These low order bits are used as extended flag bits. They are not actually part of the address of the next WCB and should be treated as zeroes when used in a pointer. The format of W$WCB is as follows:

Bit Symbol Description

0 WC$RR The file is open in read regardless mode (mode 4096).

1 WC$SPU The file is open in special update mode (mode 5).

2 WC$AEX Always do a real extend on the file (mode 8).

3 WC$CHE The file is open for user data caching (mode 256 or mode 2048)

4 WC$CSQ The user data caching specified by SC$CHE is sequential (mode 2048).

5-15 These bits, combined with low order bits of 0, are the address of the next WCB for this FCB.

4.0 DEVICE CONTROL

Device control breaks down into two parts: information needed by the monitor to perform logical operations (opens, closes, etc.) on the device and information needed by the device driver to interface with the device at the physical level.

While we will be talking about data structures that are used by device drivers, we will not be discussing how to actually write a device driver. Writing drivers is beyond the scope of this article but is covered in my book on RSTS monitor internals.

The logical operations available to a user are OPEN, CLOSE, GET, PUT and SPEC. Except for opening a device, the only structures required are the user's XRB (see the RSTS/E System Directives Manual) and the Device Data Block (DDB). Opening a device requires the data structures DEVNAM, DEVCNT, DEVPTR, UNTCNT and DEVTBL, which are used to verify the name and unit number specified for the device and associate it with the physical device information.

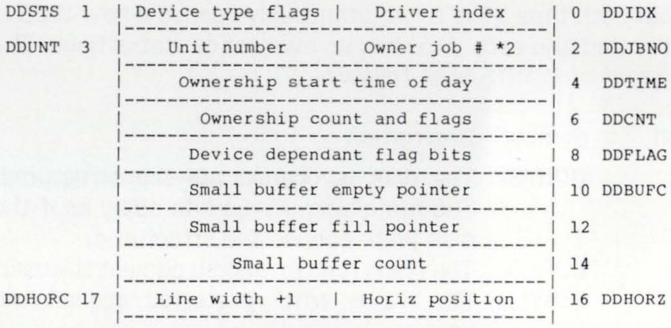
The physical operation of a driver uses information passed by the user in his XRB and device specific information kept in a structure called the Device Data Block (DDB). Portions of the information in the DDB are standardized between all devices and portions of it are specific to the needs of the particular devices.

4.1 DDB — DEVICE DATA BLOCKS

The structure used to control physical use of a device by a driver and to communicate between monitor routines and the driver is the Device Data Block (DDB). Every unit of every device has its own DDB associated with it.

A DDB contains information about device opens and assignment, small buffer usage and printer position, as well as other information specific to the particular device, such as controller status and temporary buffering.

Every DDB must contain a certain amount of specific information. This minimal DDB has the following format (the information starting at offset 8 is only required if character oriented output is to be performed):



```
              -----------------------------------
DDSTS   1    | Device type flags |   Driver index    | 0  DDIDX
             |-----------------------------------|
DDUNT   3    |   Unit number     |  Owner job # *2   | 2  DDJBNO
             |-----------------------------------|
             |     Ownership start time of day     | 4  DDTIME
             |-----------------------------------|
             |      Ownership count and flags      | 6  DDCNT
             |-----------------------------------|
             |     Device dependant flag bits      | 8  DDFLAG
             |-----------------------------------|
             |     Small buffer empty pointer      | 10 DDBUFC
             |-----------------------------------|
             |     Small buffer fill pointer       | 12
             |-----------------------------------|
             |         Small buffer count          | 14
             |-----------------------------------|
DDHORC 17    |  Line width +1    |  Horiz position   | 16 DDHORZ
             |-----------------------------------|
```

Offset Symbol Description

0 DDIDX This byte contains the driver index. This value is unique to each device type.

1 DDSTS This byte contains a set of bits that describe the characteristics of the device (see section 4.1.1).

2 DDJBNO This byte is the job number, times 2, of the job that has this device unit assigned, either explicitly by an ASSIGN or implicitly by an OPEN.

3 DDUNT This byte specifies the unit number of this device.

4 DDTIME This word contains the system time at the time the device was assigned. It is used to charge device time when the device is closed.

6 DDCNT This byte is the ownership count for this device unit. It is incremented each time the device is opened and decremented when it is closed.

7 This byte is a set of device flags (see section 4.1.2).

RSTS/E SOFTWARE PACKAGES

- **KDSS,** a multi-terminal key-to-disk data entry system. (Also available for RSX-11M.)
- **TAM,** a multi-terminal screen-handling facility for transaction-processing applications. (Also available for RSX-11M.)
- **FSORT3,** a very fast sort. Directly sorts RSTS/E files containing up to 16 million keys or records. Up to 70 times as fast as the RSTS-11 Sort package in CPU time.
- **SELECT,** a convenient, very quick package for extracting records that meet user-specified selection criteria.
- **BSC/DV,** a device driver for the DEC DV11 synchronous multiplexer that handles most bisynchronous protocols.
- **COLINK,** a package that links two RSTS/E systems together using DMC11s. Supports file transfers, virtual terminals, and across-the-link task communication.
- **DIALUP,** a package that uses an asynchronous terminal line to link a local RSTS/E system to a remote computer system. Supports file transfers, virtual terminals, and dial-out through a DN11.

(The performance-critical portions of the first five packages are implemented in assembly language for efficiency.)

Evans Griffiths & Hart, Inc.
55 Waltham Street
Lexington, Massachusetts 02173
(617) 861-0670

CIRCLE 29 ON READER CARD

8	DDFLAG	This word contains device specific flag bits and is not used by the monitor.
10	DDBUFC	The following three words contain small buffer control information. The first word is the pointer to where bytes are to be removed from the small buffer chain. The second word is the pointer to where bytes are to be inserted into the small buffer chain. The third word is the number of small buffers remaining in the allocation for this device. When this word reaches 0 the program may be stalled for small buffer wait.
16	DDHORZ	This byte is the current horizontal position of the "print head". A value of 0 is the right hand margin. A value equal to DDHORC is the left hand margin.
17	DDHORC	This byte is the device's line width, plus one.

4.1.1 DDSTS — Device Characteristics Flags

The device characteristics bits in DDSTS have the following meaning:

Bit	Symbol	Description
8	DDPRVO	Ownership of this device unit requires privileges.
9	DDRLO	Read privileges are never given for this device. This device is inherently write only. 10 DDWLO write privileges are never given for this device. This device is inherently read only.
11	DDNET	This device exists elsewhere in a DECNET network.
12		Unused.
13	DDAUX	This device can use KMC11 bridge blocks. Bridge blocks provide a means of communication between the computer and a KMC11 processor.
14	DDAUXA	This device is currently bridged to a KMC11.
15	DDSTAT	This bit is not used by the monitor but is cleared each time the device is closed. The driver can use this bit to automatically reset a condition when the device is closed.

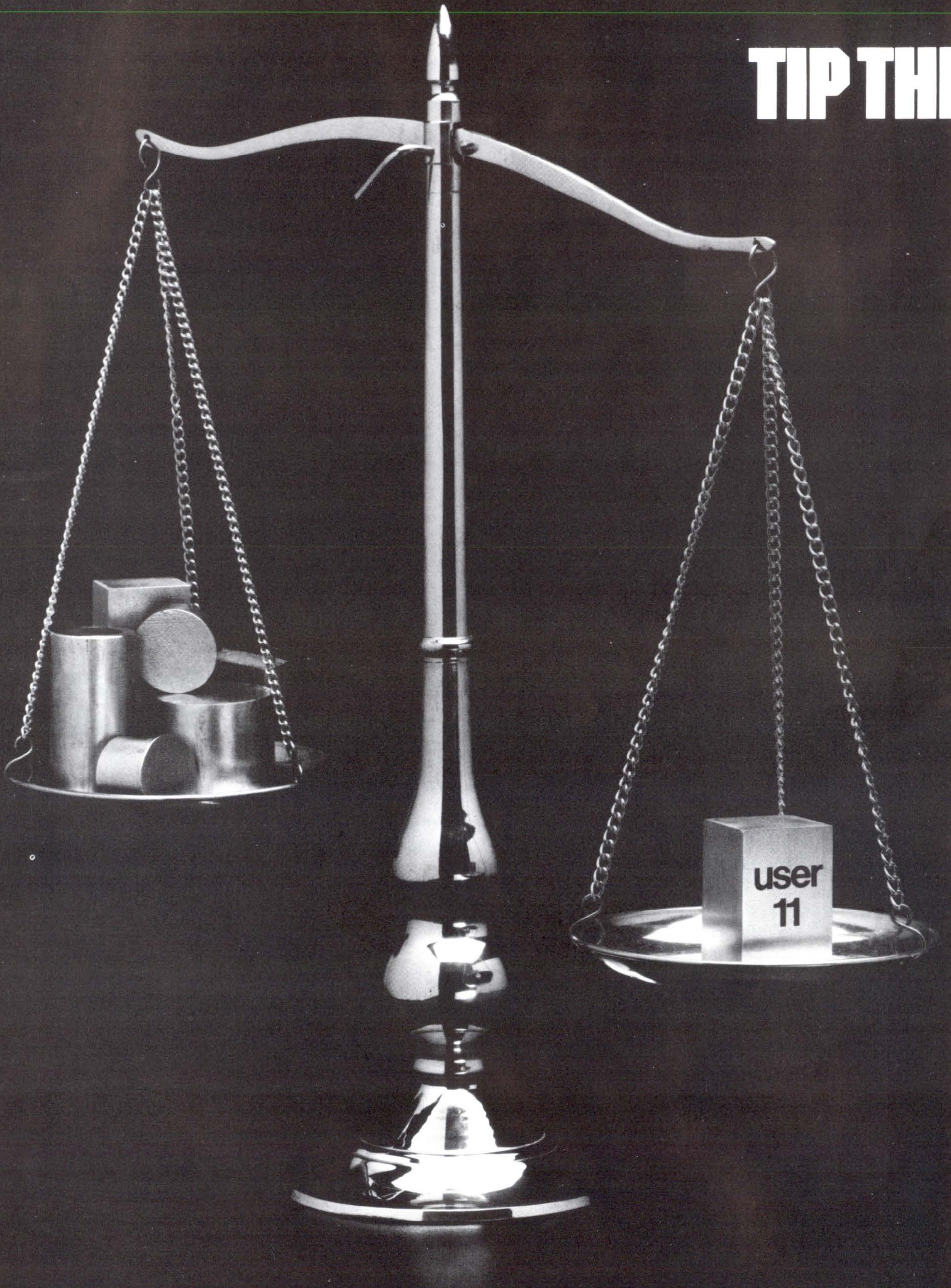
4.1.2 DDCNT — Device Flags

The device flag bits in DDCNT have the following meaning:

Bit	Symbol	Description
8-12		Unused
13	DDCONS	This device is its owner's console terminal.
14	DDUTIL	This device is temporarily assigned to FIP.
15	DDASN	This device is explicitly assigned by the job shown in DDJBNO.

. . . continued on page 86

TIP THE



APPLICATIONS SCALE IN YOUR FAVOR...WITH USER-11.

USER-11 is a comprehensive applications development facility for the DEC RSTS operating environment. Dozens of integrated programs harness RSTS's power for unparalleled productivity and performance in constructing on-line and batch application systems.

**PRODUCTIVITY...
A MATTER OF TIME.**

More than a data management system, USER-11 features common-function programs that permit numerous applications to be installed without writing a single line of code. Complete building blocks and interfaces are provided for those remaining applications requiring custom work.

**PERFORMANCE...
SIMPLY INCREDIBLE.**

USER-11 combines advanced BASIC and MACRO coding techniques with ultra-efficient file accessing mechanisms to optimize application system performance.

**RELIABILITY...
A PROVEN FACT.**

USER-11 is currently installed on hundreds of time-sharing systems world-wide with a reliability record that users repeatedly praise. All software is exhaustively tested and benchmarked prior to any distribution release.

**SECURITY...
MORE THAN RSTS.**

USER-11 incorporates a unique MENU system which flexibly and securely controls all processes. Secondary, encoded security databases are provided for each project. A special Run Time System is invoked to prevent accessing the RSTS ready state, unless the software developer desires this for the user.

**STANDARDIZATION...
A BYPRODUCT.**

All USER-11 generated packages employ programming and documentation conventions which enhance compatibility, readability, and maintainability.

**ADAPTABILITY...
NO PROBLEM.**

USER-11 programs are dictionary and parameter driven throughout. Files can be restructured without program modifications.

CIRCLE 30 ON READER CARD

**DOCUMENTATION...
GOOD AND PLENTIFUL.**

USER-11 features a wealth of easy-to-follow documentation. An extensive on-line "/HELP" facility is at software developer and user fingertips. All documentation is maintained and distributed on your system's compatible media.

TRAINING...ALL KINDS.

USER-11 training courses are held frequently with instructional programs to suit your need—beginner to expert.

FEATURES...ON AND ON.

USER-11 includes virtually every facility needed to quickly construct high performance management applications—nothing else is required. If you find this hard to believe or would like more information, contact us; we will furnish you with solid user proof!

NCCS
USER
11

**North County
Computer Services, Inc.**
2235 Meyers Ave.
Escondido, California 92025
(714) 745-6006, Telex: 182773

DEC and RSTS are registered trademarks of Digital Equipment Corporation.

© Copyright NCCS

The Supermarket of Computer Products

PRESENTS

Disk Sub Systems Ala Carte

DEC, DATA GENERAL and Others

For LSI11,* PDP11,* PDP11/70,* VAX,* UNIBUS,* & MASSBUS.*

*Tradenames of Digital Equipment Corp.

Featuring:

- CONTROL DATA's
  SMD, CMD, MMD, FMD,
  HAWK, LARK, PRINTERS,
  TAPE DRIVES & MEMORIES
- KENNEDY's
  WINCHESTERS
  & TAPE DRIVES
- DATA PRODUCT's
  PRINTERS
- CENTURY DATA's
  TRIDENTS

Plus a Menu of CONTROLLERS:

- EMULEX • SYSTEM INDUSTRIES
- WESTERN PERIPHERALS
- DILOG • BYTRONIX • RIANDA
- MCT • DIGIDYNE

ONLY GF OFFERS THIS UNIQUE COMBINATION:

- CUSTOM-TAILORED TO YOUR EXACT REQUIREMENTS
- FULL EMULATION
- COST SAVINGS OF 50%
- FAST DELIVERY
- EXPERT INSTALLATION & SERVICE
- OEM DISCOUNTS AVAILABLE

And these other GF products: Printers & Subsystems; Tape Drives & Subsystems; Computers, Memories and Controllers; CRTs, Terminals & Media; Turnkey Business Systems; Software (Operating & Application)

G F

GEORGE FÖLDVARY & ASSOCIATES, INC.
400 S. Beverly Drive, Suite 310, Beverly Hills, CA. 90212
(213)551-2840 • TWX 910-490-2511

GF: I am interested in:

□ Printers
□ Controllers
□ Memories

□ Tapes
□ CRTs
□ Media

I am an:

□ End user
□ Institutional
□ OEM

My requirements are:

□ Immediate □ 3-6 months □ Information only

Name ____________________

Title ____________________

Company ____________________

Address ____________________

City ____________ State ______ Zip ______

Telephone (______) ____________

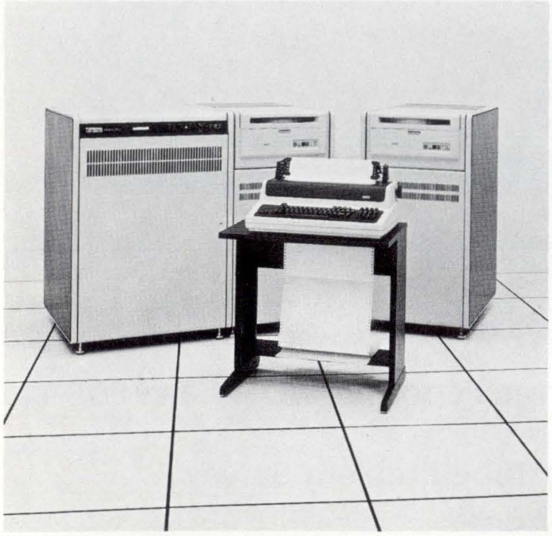
My computer is a ____________________

CIRCLE 31 ON READER CARD

The VAX-SCENE

Number 4 (RSTS PROFESSIONAL, Vol. 3, No. 3) September 1981

INSIDE:



VIDIO/11 and SCRNIO/11: CRT-Independent Screen Forms for VAX-11 and PDP-11

By Stephen F. Heffner, President, PSI (Pennington Systems Inc.)

Why Screen Formats?

People like forms. I know, we curse about the IRS 1040, the W-2, the report to Purchasing in quintuplicate; but when we have to supply data to an interactive program, we're most comfortable filling in a form. It probably has to do with the comforting prompting messages and explanatory information surrounding the field we're filling in, or perhaps a feeling of satisfaction from the completion of a form. Whatever the reason, forms seem to be much more satisfactory than line-by-line prompted entry, especially for computer-naive people.

With the widespread and ever-growing use of video terminals (CRTs to us hard-core types), interactive forms are a very practical method of communicating with a computer program, at least from a human-engineering point of view. The trouble is, they're a pain to program! First you have to worry about the intricacies of cursor control on your brand of CRT. Then how do you protect the screen from erroneous input? About the time you have all that figured out, your boss goes out and buys another brand of CRT and you start over. Meanwhile, your programs are a crazy quilt of escape sequences, reads, writes, special functions; the actual logic of the application is in there somewhere, but where?

On top of the technical difficulties of creating and servicing screen formats, there's the problem of accommodating changes in the application. The end user likes the form, but he'd like it even better if the title was over there, and this field got moved down a little bit, and, and, and... While you're at it, that "yes or no" field just became a "maybe" field as well, so allow an "M" to be entered, in addition to "Y" and "N".

Wouldn't it be nice to just drop all those capabilities into your application program in the form of straightforward, readable subroutine calls? We thought so too: enter SCRNIO/11 (screen eye oh) and VIDIO/11 (vid eye oh).

A Little History

As Ted Baxter might say, it all started in a little data entry application on a PDP-11/03, under RT-11. There were about 1,500,000 documents waiting to be entered onto floppy disks, and they had to go in fast and with minimum error. We started with a set of "primitives' for video I/O and used them to hard-code some formats for data entry, on a VT-52 as it happened. That worked pretty well, and we studied the data entry operators to see what could be done to speed up the input while (we hoped) cutting down on the error rate; this resulted in some"tuning' of the field input function. At this point, we had the first cut at VIDIO/11, the video "primitives", pretty well shaped up.

Meanwhile, we started analyzing the error checking process in terms of field attributes, although the screen formats were still "hard coded" in the application. It became pretty clear we wanted the formats out of the application program, in as generalized a fashion as possible. That requirement became even stronger when we provided VIDIO/11 to an OEM for use in a banking application package, under RSX-11M; the application would involve many screen formats and a wide variety of input error checking.

In the process, we found we also needed to use our "primitives" on a totally different brand of CRT, so we reimplemented them for the new model. This rubbed our noses in the need for CRT independence, so we added that to our list of things to do. Furthermore, it turned out that the **same programs** had to run from different kinds of CRTs on the same system; the independence ought to be at **run-time** so the same task image would be usable with any supported CRT. We added support for RSX-11D and IAS at about the same time.

The result of all this was SCRNIO/11, our screen format subroutine library and editor. Recently we have converted both VIDIO/11 and SCRNIO/11 to VAX-11 native mode (using CONPAX, our assembly language conversion utility I wrote about in the last issue); we have a RSTS version underway, and will thus have both packages usable under almost every major DEC operating system.

Anatomy of a Screen Format

SCRNIO/11 stores screen formats in files (one per file); alternatively, they can be built into the user task (memory-resident). A screen format consists of **background text** and **fields** (for input or output). The background text can be anything the user wants; any connection between it and the fields imbedded in it is up to the user. A screen format occupies an area on the physical screen determined by its first and last lines, and can be as small as a single line or as large as the whole screen (except the last line, reserved for messages and operator communication). This allows multiple formats on the screen at the same time, without any limitations on their simultaneous use.

If a screen format is disk-resident (the usual case), the application program retrieves it into a user-supplied buffer with a SCRNIO/11 subroutine call; it is then available for use. This lets the programmer choose whether to read all his formats in once, to save on disk I/O during use, or to have several formats share a common buffer, to save memory.

Field Attributes: a Baker's Dozen Plus a Few

Each field in a screen format can have a wide variety of attributes assigned to it. These attributes come in two flavors: **processing** attributes and **error-checking** attributes. They include:

1. Field number, 1 through 127. Arbitrary as far as SCRNIO/11 is concerned. Shows on the screen (unless suppressed by "no number" attribute), so CRT operator can refer to a field the same way the application program does.
2. Line and column. This is relative to the screen format, so fields move with the format if it is relocated on the CRT screen.
3. Field length. In the case of a variable-length field, maximum length. For a fixed-length field, this is the length to which the field will be filled with its fill character when it is left or right justified.
4. Fill character (any printable character or <NULL>), and whether to justify left or right. (Only for fixed-length fields.)
5. A default value (optional). This is returned to the application program if the CRT operator just hits <RTN>. This can be used to save keystrokes when a field is usually one particular value, but may be other values from time to time.
6. A "missing" value (optional). This is returned to the program if the CRT operator just hits <TAB>, or just hits <RTN> and no default value is defined. This can serve as a "place marker" (for instance, in a record) meaning we don't have a value for this field yet.
7. "Mandatory" or "optional". If a field is mandatory, SCRNIO/11 will require an explicit override by the operator if he doesn't enter (or default) a value for this field.
8. A mask for the field's values (optional). This mask is an image of the value; each character may specify numeric, alpha, a special character, or anything. A "DEC standard" date mask would be "99-AAA-99" (for instance, 12-AUG-81).
9. A low and/or high value for the field, for range checking (optional).
10. A set of "legal values" for the field (optional). Up to 127 such values are allowed per field. They must match the field's mask (if any). For instance, a field might have to be 0, 2, 7, or 9; or the Y/N/M (yes, no, maybe) field mentioned above.
11. Display only. This means the field can't be use for input. Also suppresses the field's number on the screen.
12. No number. The field may be used for input, but don't show its number on the screen.
13. No terminator required. If the operator types to the end of the field, complete the input. Otherwise, SCRNIO/11 requires an explicit terminator (such as <RTN>).
14. No echo. Good for password fields and such.
15. No override permitted. If an error is detected, SCRNIO/11 won't allow the operator to override it if this attribute is turned on.
16. No reentry on error. This attribute tells SCRNIO/11 to perform all error checks and indicate the results in the status returned to the program, but don't complain about any errors found. The application can then choose what to do about errors.

When the application program gets a field value via SCRNIO/11, the status returned with the value indicates any errors found (and overridden, unless the field is "no reentry"). The program can ignore such errors or take action, as appropriate. If a field is "no override", the program need not worry about data errors; they will never get through SCRNIO/11. (But the operator can always abort the input with <will return an error status to the program.)

An Editor for Nonprogrammers

Since one of our goals was to remove the programmer from his "middle man" position between the end user and his application, we felt that it was important to provide a way to create or edit screen formats that wouldn't be intimidating to a nonprogrammer. The result was SCRFMT (SCReen ForMaT editor), a menu-driven, self-prompting format editor. SCRFMT is about as "bullet-proof" as we can make it; it error-checks all inputs and behaves gracefully when it gets something unexpected, and the user can "back up" from **any** point where he's being asked for input, just by hitting <ESC>, with no harm done. It also has default values established for all inputs where they make sense, so the user can "carriage return" his way through many of the required inputs.

SCRFMT allows the user to create a new screen format, change one he already has, or simply review a format in various ways. (See the SCRFMT menus and forms.) It runs on any supported CRT with a screen at least 24 lines by 80 columns, and uses reverse video (if available) to highlight the field it's currently requesting input on.

Video Primitives: Cursor Control, Scrolling and More

We have retained VIDIO/11, the video I/O "primitives" we started with, for use when screen formats won't do the whole job. These include read and write (fixed and variable length) with cursor control, with or without echo or required terminator; clear line or screen; video attributes (such as reverse video); and partial screen scrolling. For instance, an application might require a dynamic display on part of the screen while reserving another part for a screen format; or the user might request that some text data be scrolled in the middle of the screen, between two formats.

VIDIO/11 tracks the cursor position, so the application program can find out where the cursor is at any time, and can specify "current" line or column. VIDIO/11 also provides a save/restore function for the current cursor position and video attributes, on **all** CRTs.

VIDIO/11 comes with SCRNIO/11 (which uses it for all video I/O), and is also available separately. For example, it might be used in a real-time application where formats aren't appropriate, or to provide CRT independence for a screen-oriented text editor.

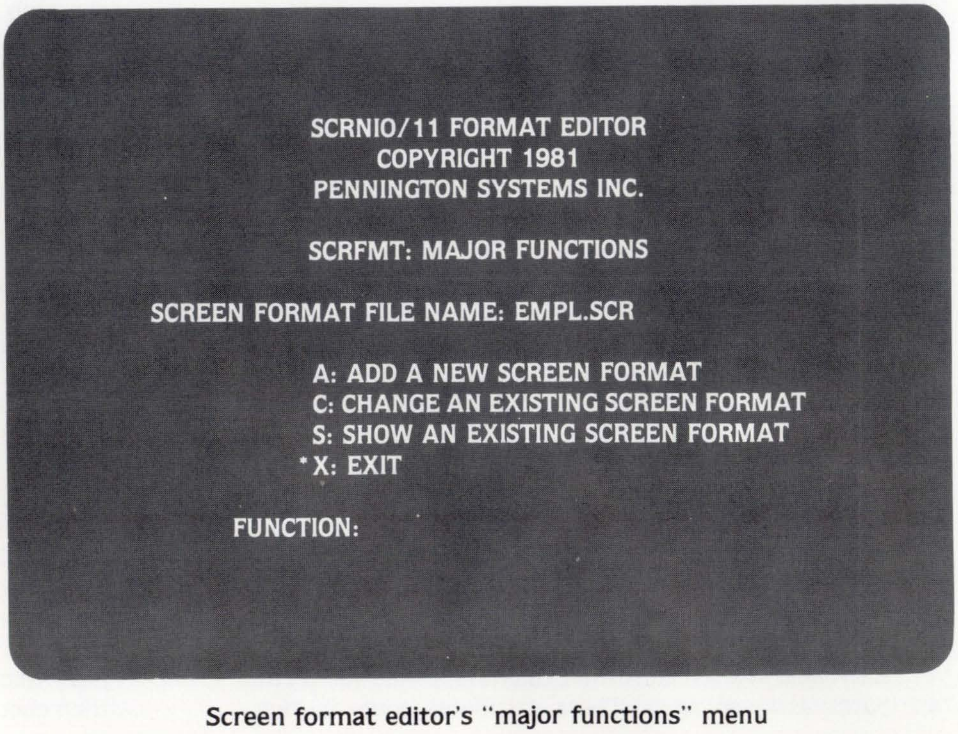
More About CRT Independence

VIDIO/11 (and therefore SCRNIO/11) configures itself at run-time for the CRT on which it's being run. It finds the CRT model through a configuration file created with VCRCNF (Vidio CReate CoNFiguration). The system manager creates a text file in which each port's CRT may be described; this text file is then run through VCRCNF to create the configuration file VIDIO/11 wants. When an application program containing VIDIO/11 is run, VIDIO/11 first checks the user's directory for such a configuration file, then the system account; so a user can have a "local" version of the configuration file. (On the VAX, the file is found through a logical name, allowing complete flexibility for the user.) The configuration file can include a "wildcard" entry that matches all ports (normally the last entry if used, for obvious reasons).

If no configuration file is found, or if the user's port isn't described, VIDIO/11 prompts the user on the CRT to enter the terminal model name.

Once VIDIO/11 knows the CRT name, it looks for a CRT support file. As with the configuration file, it first checks the current directory, then the system account. (And, again, on the VAX it uses a logical name.) Assuming such a support file is found, VIDIO/11 reads it to configure for the CRT being used. The whole process is carried out without attention from the application program or the CRT operator (unless he must enter the CRT name).

After VIDIO/11 has configured for the current CRT, it handles requests for video functions (either directly from the application program or from SCRNIO/11) on a CRT-independent basis. If the program tries to do something



```
          SCRNIO/11 FORMAT EDITOR
              COPYRIGHT 1981
          PENNINGTON SYSTEMS INC.

          SCRFMT: MAJOR FUNCTIONS

  SCREEN FORMAT FILE NAME: EMPL.SCR

          A: ADD A NEW SCREEN FORMAT
          C: CHANGE AN EXISTING SCREEN FORMAT
          S: SHOW AN EXISTING SCREEN FORMAT
         *X: EXIT

      FUNCTION:
```

Screen format editor's "major functions" menu

DBL

for

RSTS

DBL is a DEC DIBOL-11 source code compatible language and compiler, with structured extensions, available from DISC for RT-11, RSTS and RSX-11M operating systems. DBL runs under the RT-11 runtime under RSTS, and uses the normal RT-11 linker to link DBL programs. Because DBL object modules are compatible between RT-11 and RSTS, it is possible to do program development under RSTS and run the same programs under RT-11.

Some of the features offered with DBL under RSTS:

- **Structured Programming Extentions**
- **A more powerful and sophisticated debug facility which requires no special linking of a "debug" module**
- **Better ISAM support — file reorganizations are seldom necessary**
- **Intra-job and Inter-job message communication**
- **Easy interfacing with assembly-level subroutines and chaining to assembly-language programs**

Discounts are provided to OEM's along the following discount structure:

Quantity 1	**$4,200.00**	**11 - 25**	**$2,520.00**
2 - 10	**$2,940.00**	**26 +**	**$2,100.00**

Prices effective February 1, 1981

For more information about DBL under RT-11, RSTS or RSX-11M, or any of DISC's program development tools, call or write:

DISC

Digital Information Systems Corporation
6247 Fair Oaks Blvd.
Carmichael, CA 95608
(916) 485-4849

DEC, DIBOL-11, RT-11, RSTS & RSX-11M are registered trademarks of Digital Equipment Corporation.

DBL and DISC are registered trademarks of Digital Information Systems Corporation.

CIRCLE 32 ON READER CARD

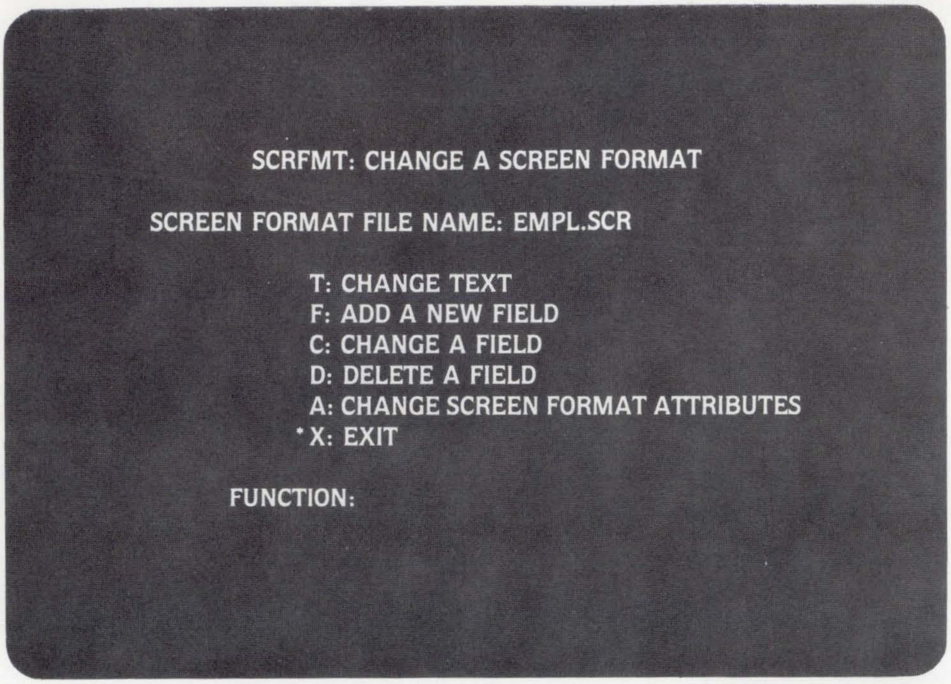
that's impossible on the current CRT, VIDIO/11 returns a specific error code that says so. This makes it possible to write application programs that are completely CRT-independent but take advantage of whatever functions the current CRT provides; SCRFMT is a good example of such a program.

What About Overhead?

Since we expected VIDIO/11 and SCRNIO/11 to be used in a wide variety of situations, on all shapes and sizes of VAXes and PDP-11s, we had to worry about the burden they place on the system. We were concerned about three areas of overhead: run-time processing, physical memory, and (on the PDP-11) address space in the user task. VIDIO/11 (and therefore SCRNIO/11) is implemented to require only **one** I/O request on a typical field input or output. We looked at the advantages of doing character-by-character I/O (instant character checking, character insert and delete, etc.) and decided they didn't justify the considerable overhead required to service all those exec requests (or worse yet, swapping on each character!). On the other hand, we wanted to trap illegal input or output characters, in order to protect the screen contents; fortunately we were able to do so without character-by-character I/O (except under RSX-11D and IAS, where it is unavoidable). The resulting CPU overhead is no greater than required for normal terminal I/O.

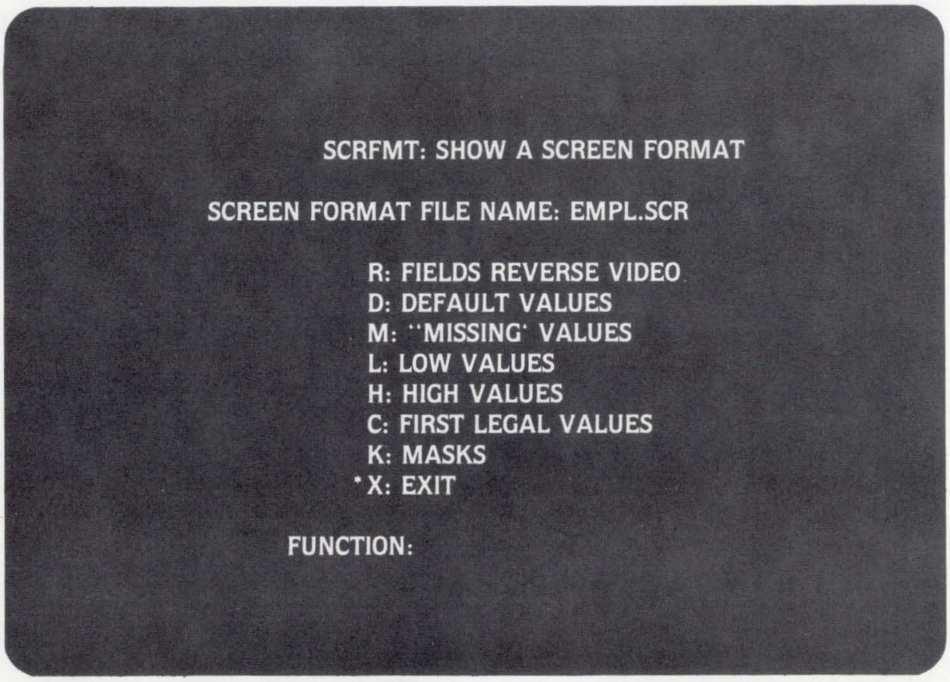
We also didn't want to take up any more physical memory than absolutely necessary, so we made all the code pure, reentrant, and position independent. This means it can go into shared libraries, shared sections, multi-user tasks, or Run Time Systems. (For multi-user tasks under RSTS, see the June issue of the RSTS Professional.)

To use a minimum of address space (a subject dear to the heart of any B+2 or COBOL user), we structured VIDIO/11 and SCRNIO/11 to be overlaid effectively, using either disk-resident or memory-resident overlays. Also, only the specific subroutines needed by the application are built into the image; unused routines don't take up space. We also fully data-compress the text of a screen format, to minimize the size of the user buffer required for it.



```
          SCRFMT: CHANGE A SCREEN FORMAT

SCREEN FORMAT FILE NAME: EMPL.SCR

          T: CHANGE TEXT
          F: ADD A NEW FIELD
          C: CHANGE A FIELD
          D: DELETE A FIELD
          A: CHANGE SCREEN FORMAT ATTRIBUTES
         *X: EXIT

     FUNCTION:
```

Screen format editor's "change a format" menu



```
          SCRFMT: SHOW A SCREEN FORMAT

SCREEN FORMAT FILE NAME: EMPL.SCR

          R: FIELDS REVERSE VIDEO
          D: DEFAULT VALUES
          M: "MISSING' VALUES
          L: LOW VALUES
          H: HIGH VALUES
          C: FIRST LEGAL VALUES
          K: MASKS
         *X: EXIT

     FUNCTION:
```

Screen format editor's "show a format" menu

The Bottom Line

VIDIO/11 and SCRNIO/11 are proving themselves in a wide variety of application areas, from Berkeley to Boston and France to the Persian Gulf. They find themselves employed in pharmaceutical laboratories, financial institutions, government offices, and manufacturing plants, running on CRTs made by Hazeltine, GTC, H-P, DEC (of course), and others.

SCRNIO/11 and VIDIO/11 are

RSTS/E ON VAX
ROSS/V
(RSTS/E Operating System Simulator for VAX)

ROSS/V is a software package, written in VAX-11 MACRO, which provides a RSTS/E monitor environment for programs running in PDP-11 compatibility mode on DEC's VAX-11.

ROSS/V supports:

- The BASIC-PLUS interactive environment.
- Concurrent use of multiple run-time systems.
- Update mode (multi-user read/write access to shared files.)
- CCL (Concise Command Language) commands.
- An extensive subset of RSTS/E monitor calls.

ROSS/V runs under VMS and interfaces to programs and run-time systems at the RSTS/E monitor call level. ROSS/V makes it possible for DEC PDP-11 RSTS/E users to move many of their applications directly to the VAX with little or no modification and to continue program development on the VAX in the uniquely hospitable RSTS/E environment. Most BASIC-PLUS programs will run under an unmodified BASIC-PLUS run-time system.

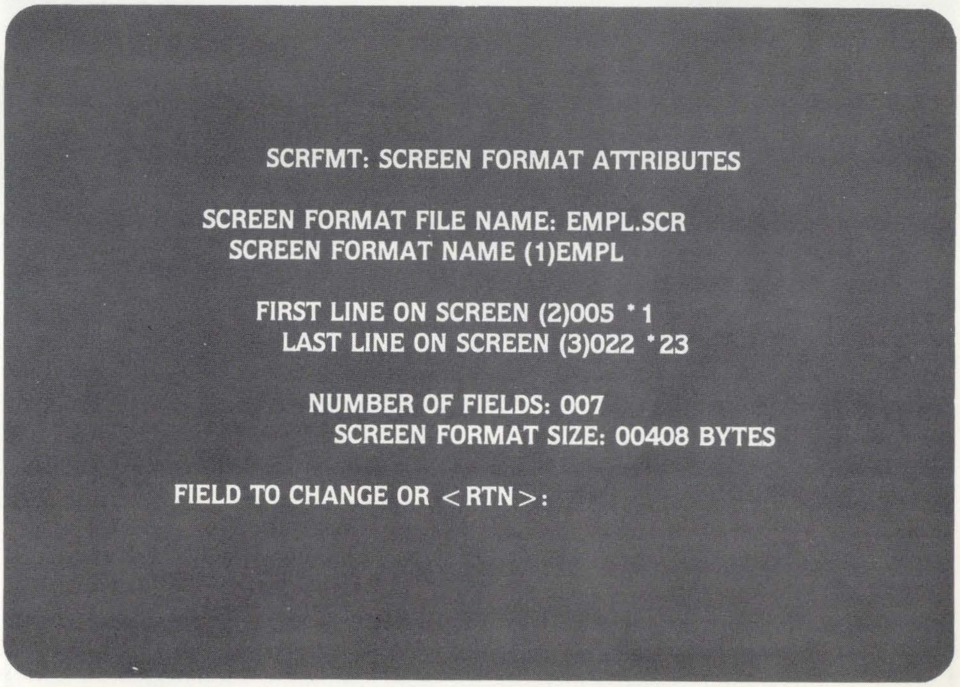
RSTS, PDP-11, VAX-11, and DEC are trademarks of Digital Equipment Corporation.

ROSS/V is available from:

(Eastern U.S.)
Evans Griffiths & Hart, Inc.
55 Waltham Street
Lexington, Massachusetts 02173
(617) 861-0670

(Central U.S.)
Interactive Information Systems, Inc.
10 Knollcrest Drive
Cincinnati, Ohio 45237
(513) 761-0132

(Western U.S.)
Online Data Processing, Inc.
N. 637 Hamilton
Spokane, Washington 99202
(509) 484-3400



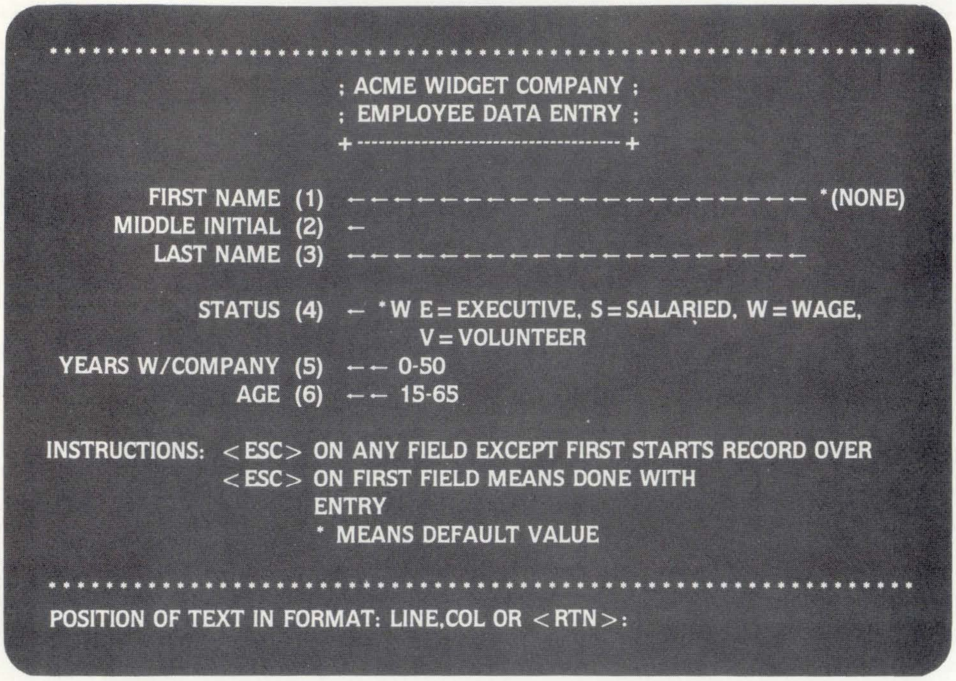
```
                SCRFMT: SCREEN FORMAT ATTRIBUTES

          SCREEN FORMAT FILE NAME: EMPL.SCR
            SCREEN FORMAT NAME (1)EMPL

               FIRST LINE ON SCREEN (2)005 *1
                 LAST LINE ON SCREEN (3)022 *23

                      NUMBER OF FIELDS: 007
                        SCREEN FORMAT SIZE: 00408 BYTES

        FIELD TO CHANGE OR <RTN>:
```

Screen format editor's "format attributes" form

available under permanent license from PSI. Both products cost $3,500 for the first CPU; VIDIO/11 by itself costs $1,500. Additional CPUs can be licensed at substantial discounts. OEM licenses are available as well. Consulting and application assistance are also available from PSI on a per diem basis.

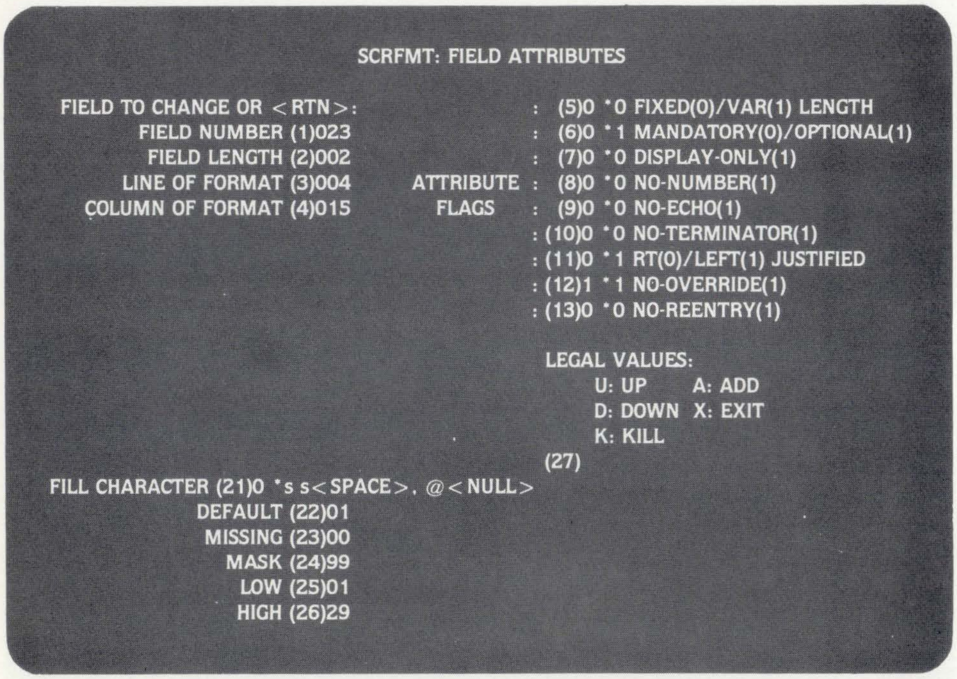
Further information can be obtained from PSI (Pennington Systems Incorporated), 65 South Main Street, Pennington, NJ 08534; phone: (609) 737-2727 cable address: PSIUSA.

(SCRFMT screen formats COPYRIGHT 1981 by Pennington Systems Incorporated reprinted by permission.)



```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

                         ; ACME WIDGET COMPANY ;
                         ; EMPLOYEE DATA ENTRY ;
                         +-----------------------+

         FIRST NAME  (1)  ——————————————————————  *(NONE)
     MIDDLE INITIAL  (2)  —
          LAST NAME  (3)  ——————————————————————

             STATUS  (4)  —  *W E=EXECUTIVE, S=SALARIED, W=WAGE,
                                V=VOLUNTEER
  YEARS W/COMPANY  (5)  —— 0-50
                AGE  (6)  —— 15-65

INSTRUCTIONS:  <ESC> ON ANY FIELD EXCEPT FIRST STARTS RECORD OVER
               <ESC> ON FIRST FIELD MEANS DONE WITH
                     ENTRY
                     * MEANS DEFAULT VALUE

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
POSITION OF TEXT IN FORMAT: LINE,COL OR <RTN>:
```

Entering or changing a screen format's text



```
                           SCRFMT: FIELD ATTRIBUTES

FIELD TO CHANGE OR <RTN>:                       :  (5)0 *0 FIXED(0)/VAR(1) LENGTH
           FIELD NUMBER (1)023                  :  (6)0 *1 MANDATORY(0)/OPTIONAL(1)
           FIELD LENGTH (2)002                  :  (7)0 *0 DISPLAY-ONLY(1)
         LINE OF FORMAT (3)004      ATTRIBUTE   :  (8)0 *0 NO-NUMBER(1)
       COLUMN OF FORMAT (4)015        FLAGS     :  (9)0 *0 NO-ECHO(1)
                                                : (10)0 *0 NO-TERMINATOR(1)
                                                : (11)0 *1 RT(0)/LEFT(1) JUSTIFIED
                                                : (12)1 *1 NO-OVERRIDE(1)
                                                : (13)0 *0 NO-REENTRY(1)

                                                  LEGAL VALUES:
                                                      U: UP      A: ADD
                                                      D: DOWN   X: EXIT
                                                      K: KILL
                                                (27)
FILL CHARACTER (21)0 *s s<SPACE>, @<NULL>
                DEFAULT (22)01
                MISSING (23)00
                   MASK (24)99
                    LOW (25)01
                   HIGH (26)29
```

Entering or changing a field's attributes

About the Author... Stephen F. Heffner has been an independent consultant since 1972 in the areas of distributed processing, timesharing systems, data communications networks, database management, minicomputer-based dedicated applications, and software conversion, among others. He established Pennington Systems Incorporated (PSI) in 1977 to provide software products and consulting services. He is the designer and implementer of both VIDIO/11 and SCRNIO/11.

Report For Commercial Working Group On Mag Tape Problem Survey

By Ross Miller, Online Data Processing, Inc.

The following is a compilation of the type of mag tape failures that were reported in a recent tape survey we conducted, together with user comments regarding fixes or ways of preventing problems. It was apparent from the respondants comments that if DEC would produce a manual on proper handling of tapes, selection of tapes and handling and cleaning procedure of tape transport, geared specifically for operators, it would be a significant step toward solving mag tape problems.

Tape Transport Problems

A significant cause of problems resulted from a dirty capstan or improperly cleaned capston which causes the tape to slip as it tries to start tape motion. This will show up as possibly hundreds of errors on the tape when the tape drive encounters an error. What actually happens is that when an error is encountered, the tape transport will stop the tape, back the tape up and try to write it again. If there isn't good traction on the capstan as it reverses and goes forward, you will get creep in your positioning on the tape as well as a condition where the tape doesn't come up to speed fast enough.

It should be pointed out that the capstan on the DEC tape drives should be cleaned only with a clean, lint-free cloth and water. No other cleaning materials or solutions should be used on the capstan.

Tape Quality

There are four main areas of concern regarding tape quality.

1. The type of mylar used, which is the backing for the magnetic tape. There are several different types available and the quality varies considerably.
2. The adhesive used to attach the oxide coating to that mylar.
3. Testing and quality control in manufacturing.
4. Clarification of whether the tape you are buying is surplus tape or seconds.

The quality of the tape will affect you in the following ways:

The type of mylar, as well as the adhesive used to attach the oxide to the mylar, may cause problems on high speed tape drives such as the new 125 inch per second tape drives. The problem is referred to as sticktion where fast movement of the tape across the heads will generate heat on the head and when you stop the tape, the mylar will actually melt and stick to the head or other components.

Another problem you can experience with tapes is the shedding of the oxide coating. This is due to poor mylar or poor adhesive used in attaching the oxide to the mylar. As the oxide sheds from the tape, it will contaminate the head and as you put other tapes on the tape drive, it will deposit that oxide on other tapes which may not be shedding and will contaminate an entire tape library. This becomes critical with the higher density and higher speed tape transports.

On the issue of testing and quality control, the simple statement that not all tapes are created equal sums it up. Be very careful to read the specification for the tape and determine what the tape manufacture means when he says it is guaranteed. Guaranteed against what and for how long.

The last issue we have encountered is surplus tapes or secondary tapes. These tapes are generally not high quality and will be sold as new tapes. Frequently you have problems with these tapes since they are an older type of tape and are subject to shedding oxide, being brittle, getting creases in them easily and readily and are frequently made using lower quality mylar or coatings which will create sticktion problems as well as the shedding problem.

Operator Damage

IBM has developed a fairly complete operators guide on how to properly handle tapes. We have found, however, that frequently DEC users have not been exposed to proper handling techniques and will unwittingly damage tapes by squeezing the tapes on the flanges. A tape that sticks out instead of stacking neatly on rewinds is called a "flyer." If the tape is squoze on the flanges, these "flyers" will be crimped producing edge damage. As it goes through the tape drive this crimped edge will lift the tape off the head temporarily as the tape passes the crimped position. This will create all sorts of problems for your machine.

Furthermore, you should be very careful to have a dust and dirt free environment for the tape drive. Operators should insure that their hands are clean and that physical touching of the tape is kept to a minimum and should be restricted to touching the leader section of the tape. We are all familiar with diagrams that show head clearance on a disk drive between the surface of the disk and the head and what contaminants will do. At 1600 BPI or 6250 BPI densities, you have a similar problem with contaminants, if the material lifts the tape off the head even a tiny fraction, it will cause total signal loss from the tape. Many respondents to the survey indicated they were **cleaning** the **tape drive after each tape** was used. If there is any question about shedding from a tape you should definitely clean your tape drive, thoroughly cleaning the head and columns, after each tape is used.

Software

The last and yet fairly significant point is a definite need for improvement in tape handling by the software. Many people reported a high frustration level when processing several tapes in a back-up set whether DEC or the RMS back-up utility in which case, after processing two or three tapes, a tape error would cause an abort of the operation and therefore, necessitate starting from the beginning again. Most respondants were business application sites and were working with large block transfers of 2,048 bytes per block or larger. Many were working with multi-volume tape sets.

DEC Maintenance

The remainder of the problems could be the result of a lack of information to DEC field service people on what a tape drive is, how to maintain it and how to repair it properly. Several respondents indicated that DEC had spent days trying to repair their tape drives with little or no success.

Summary

In summary, we feel that the following could help correct most of the problems. Improved documentation by DEC for their personnel in maintaining the drives, documentation for tape operators to familiarize themselves with the peculiar problems of tape processing, proper handling of the tape and proper selection of tape to be used on the tape transport. This is not an endorsement of any particular tape manufacturer, but we have found that people have had the least problems with the Graham Epic 480 tapes and the highest number of problems with tapes manufactured by such companies as Wabash, Memorex, and other less noted tape manufacturers. Reasonable acceptance was found by people using 3M tapes and BASF tapes.

One final comment, there is a definite need for improving the software which is used to access the tape drives.

This report has been prepared from the survey conducted through the VAX/VMS SIG Newsletter by:

Ross Miller
ONLINE DATA PROCESSING, INC.
N. 637 Hamilton
Spokane, Washington 99202
(509) 484-3400

TIPS & TECHNIQUES

A Column For The Advanced RSTS/E User

By Steven L. Edwards, Software Techniques

Welcome to "Tips & Techniques." This column is dedicated to the premise that the free exchange of knowledge between users will benefit the entire user community.

Further, this column is dedicated to the premise that RSTS is not an end point, but it is an excellent starting point to build upon. We all know that one system can not be all things to all people. The RSTS developers do a very good job, but their efforts can not solve everyone's problems, ie., the developers are currently required to maintain Basic-Plus (read 16KW) compatibility. In future columns we will see how this limits the performance of the CUSPs, and what to do about it.

Correction!

In the last issue of the RSTS Professional, there was an article about a mini Basic-Plus-2 debugger routine (DEB). The routine located line numbers by searching through the program section (PSECT) named $CODE. The routine located the starting address and the length of $CODE by defining two additional PSECTs named $CODD, and $CODF. These new program sections had the default access code of read-write (RW). Since that article was written, a patch to the compiler has been released that changes $CODE and $PDATA to have the read-only attribute. Since TKB allocates memory for program sections alphabetically within access code (RO or RW), $CODE is no longer allocated between $CODD and $CODF. If you change the access code of $CODD and $CODF to be read-only, the routine will work as described.

EDT Version 2

In this issue we will examine EDT version 2. EDT V2 is an entirely new full screen editor with journaling and user definable keys. EDT V2's relation to EDT V1 is purely casual. EDT V1 was (yes WAS) difficult to learn and worse to use. EDT V2 is the most incredible editor I've seen, and it is available on all of DEC's major operating systems. If you are using anything else, please stop, you are wasting your valuable time. Using EDT V2 as a full screen editor (change/keypad mode) will increase your productivity. I never thought anything could replace TECO as my primary editor, but EDT V2 won me over in a single afternoon. EDT V2 lacks the conditional features of TECO, but it performs better (speed-wise) on an 11/34 than TECO/VTEDIT on an 11/70! I recommend building the memory resident library shared version if you have more than one user editing at a time.

I strongly urge any serious RSTS/E user to read the EDT manual that came with patch kits "E" and "F."

After I used EDT V2 for a while, I began to notice a few minor irritations:

- EDT V2 leaves your VT100 in ANSI mode regardless of it's initial setting.
- EDT V2 requires your terminal to be set to 'NO ESC SEQ' for the alternate keypad to be of any value.
- EDT V2 requires each user to have their own initializer file in their account.

Soon these minor irritations exceeded my tolerance, so I developed the patches below.

Each of these patches can 'stand alone,' ie. don't install the first patch if you don't want your VT100 to always be reset to VT52 mode.

If you have developed minor irritations of a software nature, developed a useful tip or technique, or have a question requiring an in-depth response, please drop me a line.

```
;       EDT V2 PROGRAM PATCHES

;       EDT FEATURE PATCH

;       JAM VT100 BACK TO VT52 MODE UPON EXIT.

;       COPYRIGHT (C) 1981 SOFTWARE TECHNIQUES, INC., LOS ALAMITOS, CA

;       MAC EDTV52.PAT=EDTV52.COR
;       LBR IOMOD.OLD=ED2LIB/EX:IOMOD
;       PAT IOMOD.NEW=IOMOD.OLD/CS:176443,EDTV52.PAT/CS:21276
;       LBR ED2LIB/RP=IOMOD.NEW
;       TKB @ED2???

        .TITLE  IOMOD

        .IDENT  /SOFTEC/

        .PSECT  IOCOD
$$$LOC  =       .                       ; START OF PSECT.
IO.CXR  =       $$$LOC+742              ; CLEAR XRB.
IO.CFR  =       $$$LOC+766              ; CLEAR FIRQB.
.       =       $$$LOC+3360             ; ADD SOME NEW CODE.

        CALL    IO.CFR                  ; CLEAR FIRQB.
        MOV     #405,R0                 ; POINT TO FIRQB+FQFUN.
        MOVB    #20,(R0)+               ; UU.TRM.
        MOV     #-1,(R0)                ; FOR OUR TERMINAL.
        EMT     66                      ; .UUO HOOK.

        CMPB    @#414,#377              ; FIRQB+FQNAM1+2.
        BNE     10$                     ; XON?, NOPE.
                                        ; WE ASSUME THAT ANY TERMINAL
                                        ;   WITH XON SET IS A VT100.

        CALL    IO.CXR                  ; CLEAR XRB.
        MOV     #442,R0                 ; POINT TO XRB.
        MOV     #LEN,(R0)               ; LENGTH OF TEXT.
        MOV     (R0)+,(R0)+             ;    "    "    "
        MOV     #TXT,(R0)               ; ADDRESS OF TEXT.
        EMT     4                       ; .WRITE

10$:    CALL    IO.CFR                  ; CLEAR FIRQB.
        EMT     50                      ; .RTS

ESC     =       155.                    ; ESCAPE.

        .ENABL  LC
TXT:    .BYTE   ESC, '[, '?, '2, '1     ; SET VT52 MODE.
LEN     =       .-TXT

        .END
```

ENABLE/34 adds a new dimension to your PDP—11/34 RSTS/E system, extending your main memory up to 4 million bytes!

ENABLE/34:

□ Substantially increases system throughput

□ Allows all currently excuting programs to remain memory resident

□ Installs with a simple AUTOPATCH procedure

□ Allows present memory to be used in concert with added memory.

Best of all, **ENABLE/34** will not affect system operation. Your system can be tested, maintained, and operated as a standard PDP—11/34 system.

ENABLE/34 is the joint product of two of the most innovative companies in today's computer enhancement industry. SOFTWARE TECHNIQUES, INC. developed the software. And ABLE, a leading manufacturer of PDP—11 enhancements for five years, produced the hardware.

If delivery problems and budget restraints have inhibited your system development, add a new dimension to your existing system with **ENABLE/34.**

For more information, call:

ABLE COMPUTER
1732 Reynolds Avenue
Irvine, California 92714
(714) 979-7030
TWX: 910-595-1729

Software Techniques, Inc.
5242 Katella Avenue
Los Alamitos, California 90720
(714) 995-0533

ENABLE/34 is not intended to replace newer or larger machines. It is designed to upgrade existing systems.

PDP, DEC, and RSTS/E are all registered trademarks of Digital Equipment Corporation.
ENABLE and **ENABLE/34** are registered trademarks of ABLE Computer.

CIRCLE 34 ON READER CARD

```
;       EDT V2 PROGRAM PATCHES

;       EDT FEATURE PATCH

;       SET 'NO ESC SEQ' UPON ENTRY.

;       COPYRIGHT (C) 1981 SOFTWARE TECHNIQUES, INC., LOS ALAMITOS, CA

;       MAC EDTNES.PAT=EDTNES.COR
;       LBR IOINIT.OLD=ED2LIB/EX:IOINIT
;       PAT IOINIT.NEW=IOINIT.OLD/CS:46475,EDTNES.PAT/CS:15150
;       LBR ED2LIB/RP=IOINIT.NEW
;       TKB @ED2???

        .TITLE  IOINIT

        .IDENT  /SOFTEC/

        .PSECT  IOINIT
.=.+76
        CALL    EDTNES                  ; CLEAR ESC SEQ.
        NOP
        NOP

        .PSECT  EDTNES
EDTNES: MOVB    #377,@#407              ; OUR TERMINAL.
        MOVB    #200,@#431              ; NO ESC SEQ.
        EMT     66                      ; .UUO HOOK.
        RETURN

        .END
```

```
 !      EDT V2 PROGRAM PATCHES
 !
 !      CHANGE DEFAULT INITIALIZER FILENAME TO 'LB:INI.EDT'
 !
 !      EDT FEATURE PATCH
 !
 !      COPYRIGHT (C) 1981 SOFTWARE TECHNIQUES, INC., LOS ALAMITOS, CA
 !
 !      LBR EDTDAT=ED2LIB/EX:EDTDAT
 !      RUN $ONLPAT
 !              (WITH APPOLOGIES FOR USING ONLPAT TO PATCH A '.OBJ')
 !      ...
 !      LBR ED2LIB/RP=EDTDAT
 !      TKB @ED2???
 !
File to patch? EDTDAT.OBJ
Base address? 1462
Offset address? 0
 Base   Offset  Old     New?
001462  000000  042105  ? "LB
001462  000002  044524  ? ":I
001462  000004  044516  ? "NI
001462  000006  020006  ? ^C                    (UPARROW/C TO EXIT)
```

a Routine from Rick

Dear Carl & Dave — Maybe you can use this routine in your next issue. It's a great magazine — keep up the good work.

— Rick Powell
Software Performance Research
8090 Steeple Chase Ct., Springfield, VA 22153

```
CRE TEST.CMD        ]
                    ]
*I                  }  use an editor to create TEST.CMD
ATPK TEST.CMD       ]  with the command ATPK TEST.CMD
*EX                 ]
Ready               ]
ATPK TEST.CMD       }  then type ATPK TEST.CMD
Ready               ]
ATPK TEST.CMD       ]
Ready               ]
ATPK TEST.CMD       ]
Ready               ]
ATPK TEST.CMD       ]
Ready               ]
ATPK TEST.CMD       ]
Ready               ]  count the number of times it nests
ATPK TEST.CMD       }  to see how many psuedo-keyboards
Ready               ]  are available.
ATPK TEST.CMD       ]
Ready               ]
ATPK TEST.CMD       ]
Ready               ]
ATPK TEST.CMD       ]
Ready               ]
ATPK TEST.CMD       ]
?No PK's available - please try again later
?Command file aborted
?No PK's available - please try again later
?Command file aborted
Ready
Ready
Ready
Ready
Ready
Ready
Ready
Ready
Ready
Ready
```

[Thanks, Rick.]

The Analyst

European Hand-Crafted Carving

Here's the ideal gift for the "programmer who has everything". Into each figure goes an assortment of exotic tropical woods, such as Jacaranda, Pau-Violeta, Palisander, Teak, Sandlewood, Ironwood, Zebrawood, Rosewood, and others, each used to create irresistible charm and humor. Stands approximately 12 inches tall.

Other professions available send for catalog today!

N.S.D. SALES
70 Jamesway
Southampton,
Pa 18966

$39.95
ADD $4.00
SHIPPING&HANDLING
Pa. Residents add 6% sales tax

CIRCLE 35 ON READER CARD

PDP II USERS

WANT AN ALTERNATIVE? YOU CAN HAVE ONE

with computer maintenance

from TYMSHARE®

- Maintenance Provided on PDPII Systems
- Mixed Vendor Systems Is Our Specialty
- Servicing Most Major U.S. Cities

Don't wait until your system is down again.
For more information write or call now!

Return to: Tymshare Inc. • 1513 E. Del Amo Blvd. • Carson, CA 90746 or call 213-638-0574

YOUR NAME ______________________

YOUR TITLE ______________________

COMPANY NAME ______________________

ADDRESS ______________________

____

____

PHONE NO. ______________________

□ I WOULD LIKE TO BE CONTACTED BY TYMSHARE TO DISCUSS MY MAINTENANCE NEEDS IN MORE DETAIL.

□ I WOULD LIKE A PRICE QUOTE ON THE COST TO MAINTAIN MY COMPUTER SYSTEM

TYPE OF MAINTENANCE COVERAGE YOU DESIRE (CHECK ONE)

□ 8 am to 5 pm Monday through Friday
□ 8 am to 8 pm Monday through Friday
□ 24 hour Monday through Friday
□ 24 hour, 7 days per week
□ Per Call Only
□ Other — Please indicate below:

MY PRESENT CONTRACT EXPIRES ______________

EQUIPMENT TYPE	QUANITY	EQUIPMENT TYPE	QUANTITY

Getting The Most Out Of CSPCOM

By Kevin Paul Herbert

Copyright (c) 1981 by Software Techniques, Inc., 5242 Katella Ave., Suite #101, Los Alamitos, CA 90720

The release of Version 7.0 of RSTS added many new features. Among these features are CSPCOM, the Cusp Compiler, and resident libraries. CSPCOM is used to compile the CUSPs (Commonly Used System Programs) to run under the RSX run-time system. DECs intended purpose was to allow users of DIBOL and COBOL to use RSX as their default run-time system and never need Basic-Plus, the language most CUSPs are written in.

CSPCOM is a scaled-down version of Basic-Plus 2. It contains most of the functionality of B+2, but is only supported for use in the compilation of CUSPs. Programs compiled under CSPCOM tend to be larger than their Basic-Plus counterparts, since under Basic-Plus, programs have a very large run-time system for support (a typical Basic-Plus run-time system is from 13K to 16K, depending on whether it is built with various optional features and whether the system has a floating point processor) but under RSX all the support code needs to be linked in with the program itself.

The resident library feature of RSTS/E allows regions of code (libraries) to be shared among several users at once. Any number of users may use a library at one time, and only one copy of the library will occupy physical memory.

Using resident libraries, it is possible to overcome the problem of CSPCOM CUSPs being larger than their Basic-Plus counterparts. I have included a command file to build a CSPCOM resident library, which occupies 8K of physical memory. Using CSPCOM and the CSPCOM resident library yields CUSPs the same size or smaller (typical savings of 1K per CUSP) as their Basic-Plus counterparts, which run considerably faster.

Users of DIBOL and COBOL can make use of the CSPCOM library to save memory. The addition of an 8K resident library is easily balanced by the reduction in CUSP size (typically 3K to 4K per CUSP). Since DIBOL and COBOL users already have compiled their CUSPs under RSX, there will be no speed increase, but there will be memory savings.

Users of Basic-Plus can reduce their overall memory usage by using the CSPCOM resident library. Since CUSPs are an average of 1K smaller under RSX with the CSPCOM library, and most systems have 8 CUSPs running at one time (take a look at your detached jobs!) the addition of the CSPCOM library will certainly not hurt. Combined with the speed increase as a result of running true compiled programs, most systems will realize an overall speed increase (a speed increase is especially noticable in the spooling package).

In addition to the creation of the CSPCOM resident library, it is possible to declare some heavily-used CUSPs as sharable tasks. These sharable tasks use both the CSPCOM resident library and a special resident library created just for one specific CUSP. Systems using two copies of any of the DEC spoolers (BATRUN, SPLRUN, etc.) can reduce their memory usage tremendously (if you use more than one spooler at once, you can save an average of 10K per additional spooler), which easily makes up for the addition of any features needed to use CSPCOM.

I have written patches to the BUILD program to support building CUSPs against the CSPCOM resident library. In addition, BUILD also will handle the creation of multi-user CUSPs. The patch to the CSPCOM program causes the compiler to indicate that the code generated is read-only (CSPCOM, like Basic-Plus 2, generates read-only code, but a bug in the compiler flags the code as read/write). The program FIXBP2.B2S (which may be compiled with CSPCOM) is used to correct the attributes of the CSPCOM library to indicate that the modules in the library are also read-only. In addition, FIXBP2 has a switch (/RW) which will change the .PSECT attributes back to the original read-write (this should be used if it becomes necessary to install patches to CSPCOM.OLB since the patches released by DEC assume that the library is read-write.)

The following procedure should be used to install the patches. This procedure assumes the following things:

1. Resident library support was included in your monitor.
2. RSX emulation was included in your monitor.
3. You have built RMS-11 (used by FIXBP2).
4. Seq 11.16.001 for MAKSIL (creating multi-user tasks) has been installed. This is only necessary if you wish to use the new multi-user feature of BUILD.
5. You have installed the RSX run-time system.

If you do not have any of these features, please include them before attempting to use these patches.

```
Step 1. Patch CSPCOM to create read-only code:

RUN $ONLPAT
Command file name? FIXCSP
File to patch? $CSPCOM.TSK          ;The rest is automatic
... patch will be installed ...
Command file name? ^Z               ;Control/Z to exit

Step 2. Compile the FIXBP2 program

RUN $CSPCOM
CSP>FIXBP2/OBJ=FIXBP2.B2S
CSP>^Z                              ;Control/Z to exit

Step 3. Task build the FIXBP2 program

RUN $TKB
TKB>@FIXBP2

Step 4. Extract the CSPCOM object library

RUN $LBR
LBR>CSPCOM=LB:CSPCOM/EX:??????
LBR>^Z                              ;Control/Z to exit

Step 5. Fix the CSPCOM object library
```

```
RUN FIXBP2
What is the name of the .OBJ file holding the library?
 CSPCOM

Step 6. Save the old CSPCOM library in case something
 happens

RUN $PIP
*LB:CSPCOM.OLD=LB:CSPCOM.OLB/RE
*^Z                                     ;Control/Z to exit

Step 7. Create a new CSPCOM library.

RUN $LBR
LBR>LB:CSPCOM/CR:165.:1152.:256.
LBR>LB:CSPCOM=CSPCOM
LBR>^Z                                  ;Control/Z to exit

Step 8. Patch the BUILD program

RUN $CPATCH

CPATCH  V7.0-07 RSTS V7.0-07 SOFTEC Dev 11/70
File to patch - BUILD.BAS=BUILD.BAS
#CSPBLD/CS:28071
#^Z                                     ;Control/Z to exit
File to patch - ^Z                      ;Control/Z to exit

Step 9. Compile the BUILD program

RUN $CSPCOM
CSP>BUILD/OBJ=BUILD
CSP>^Z                                  ;Control/Z to exit

Step 10. Task-build the BUILD program

RUN $TKB
TKB>BUILD=BUILD,LB:CSPCOM/LB
TKB>/
ENTER OPTIONS:
TKB>UNITS=12
TKB>ASG=SY:5:6:7:8:9:10:11:12
TKB>EXTTSK=512
TKB>//

Step 11. Replace the BUILD program on your library account

RUN $PIP
*$BUILD.TSK=BUILD.TSK
*$BUILD.BAC/DE:NOWARNING
*^Z                                     ;Control/Z to exit
```

```
Step 12. Task-build the CSPCOM library

RUN $TKB
TKB>@CSPLIB

Step 13. Convert the task builder's output to a resident
 library

RUN $MAKSIL
MAKSIL  V7.0-07+/MU PATCH          RSTS V7.0-07 SOFTEC Dev 11/70
Resident Library name? CSPCOM
Task-built Resident Library input file <CSPCOM.TSK>?
Include symbol table (Yes/No) <Yes>?
Symbol table input file <CSPCOM.STB>?
Resident Library output file <CSPCOM.LIB>?
CSPCOM built in 8 K-words, 516 symbols in the directory
CSPCOM.TSK renamed to CSPCOM.TSK<40>
```

At this point, you have completed the patch and installation procedure. The files CSPCOM.TSK and CSPCOM.STB should be placed in an important account other than [1,2]. I recommend the use of account LB: for this purpose. The file CSPCOM.LIB should be placed in account [0,1]. After the files are placed in the appropriate accounts, use UTILTY to add the resident library and modify your start-up procedure to add the library automatically.

If you want to create any multi-user tasks, it will be necessary to use your preferred editor (Steve would rather have me say EDT) to modify the BUILD command files. The procedure involves locating the command lines to build a program, and appending the /MU switch to the COMPILE command. For example:

```
$BREAK
$DOPAT @~PATLOC:PA1023.CMD
OLD ~INPUT:SYSTAT.BAS
COMPILE ~SYSTEM:SYSTAT<232>/MU
```

What Can One Coupon Get For You?
A Free Resume Kit...

From the Only Employment Service with Twin Inhouse PDP 11/70 RSTS systems.

An Invaluable Aid In Your Job Changing Process

The Kit presents in an uncomplicated format what we've learned in over 15 years of employment service to the computer industry. Fill out the coupon and return it to us. We'll send you our free Resume Kit.

Name	Applications
Home Address	Degree Yr. Grad. No. Yrs. Experience
City State Zip	Current Employer
Home Phone Business Phone	Job Title
Hardware	Languages

We have many exclusive Data Processing positions available — locally and nationwide.

Scientific Placement, Inc. Employment Service

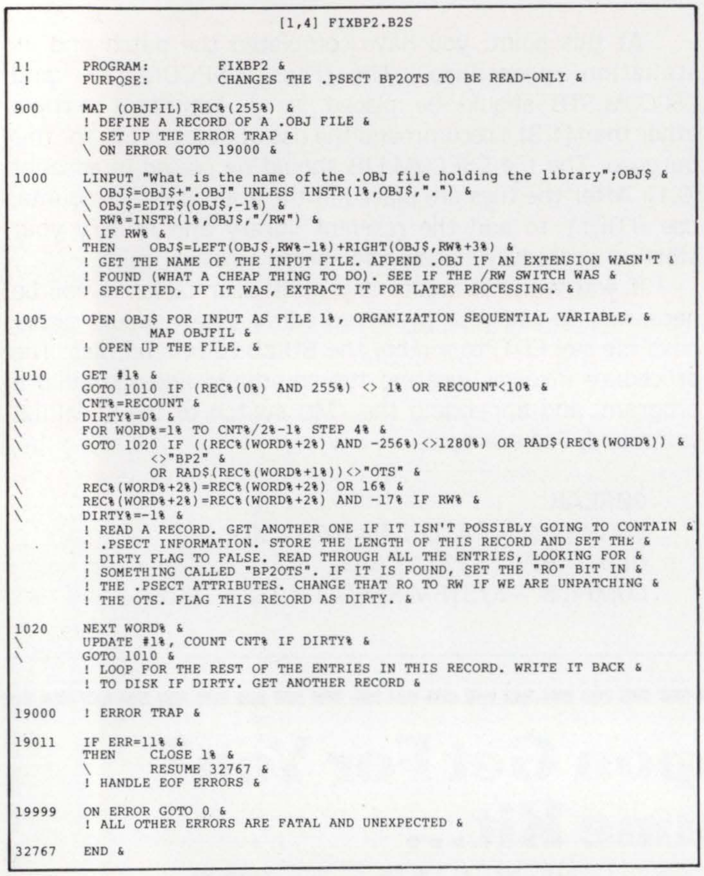
P.O. Box 19949 Houston, Texas 77224 (713) 496-6100

RSO 9/81

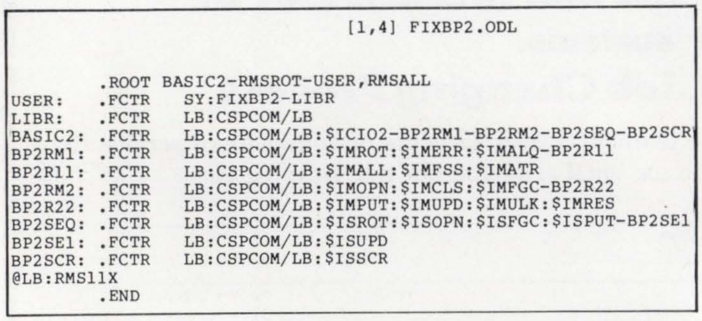
CIRCLE 37 ON READER CARD

After you have made all the desired changes to the command files, run the BUILD program to build the new library. When BUILD requests the Run-Time System to use, respond with 'RSX.' BUILD will then ask about using CSPCOM. Answer 'Yes.' BUILD will then ask if you want to use the cusp RESLIB, and where it is located. After answering these questions, BUILD will act as described in The System Generation Manual.

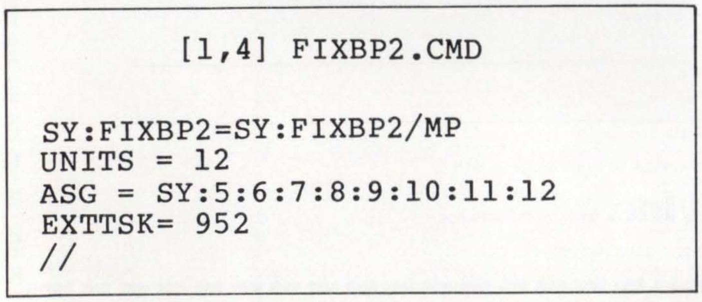
After the BUILD is complete, users of Basic-Plus should delete all the old .BAC files on the library. It is not necessary for current CSPCOM users (Dibol and Cobol) to delete anything.



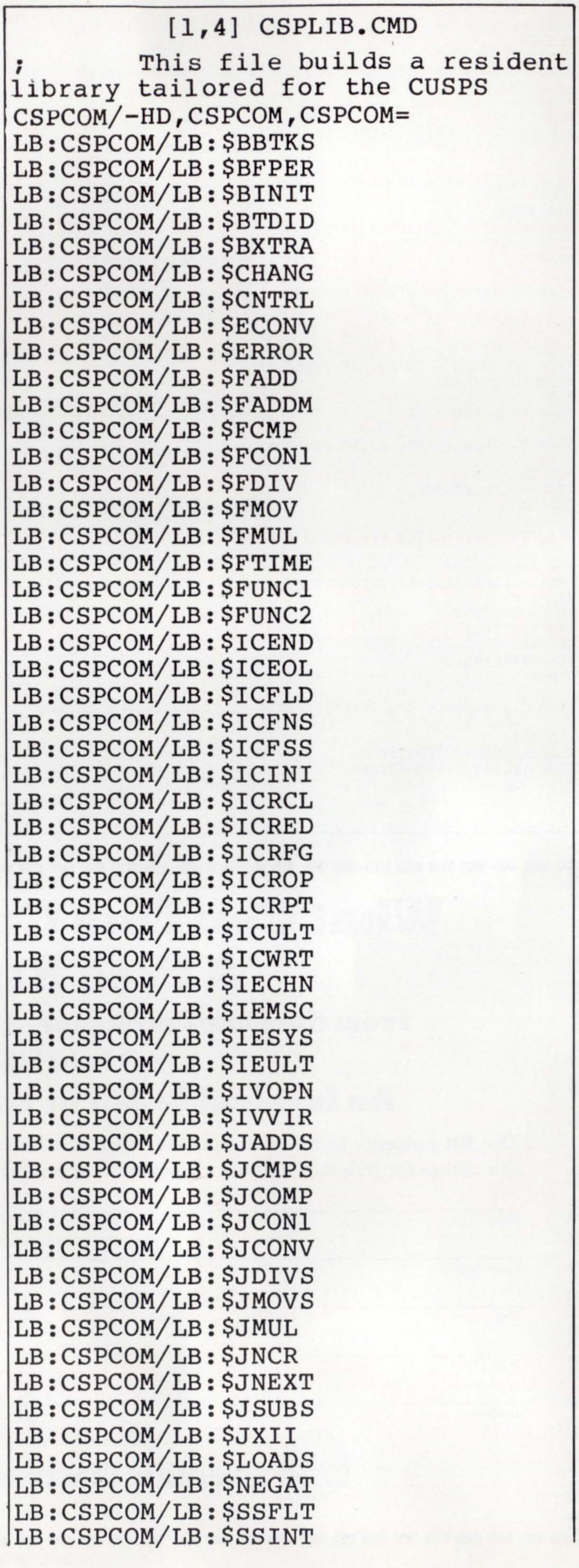
```
                              [1,4] FIXBP2.B2S

1!      PROGRAM:        FIXBP2 &
 !      PURPOSE:        CHANGES THE .PSECT BP2OTS TO BE READ-ONLY &

900     MAP (OBJFIL) REC%(255%) &
        ! DEFINE A RECORD OF A .OBJ FILE &
        ! SET UP THE ERROR TRAP &
        \ ON ERROR GOTO 19000 &

1000    LINPUT "What is the name of the .OBJ file holding the library";OBJ$ &
        \ OBJ$=OBJ$+".OBJ" UNLESS INSTR(1%,OBJ$,".") &
        \ OBJ$=EDIT$(OBJ$,-1%) &
        \ RW%=INSTR(1%,OBJ$,"/RW") &
        \ IF RW% &
        THEN    OBJ$=LEFT(OBJ$,RW%-1%)+RIGHT(OBJ$,RW%+3%) &
        ! GET THE NAME OF THE INPUT FILE. APPEND .OBJ IF AN EXTENSION WASN'T &
        ! FOUND (WHAT A CHEAP THING TO DO). SEE IF THE /RW SWITCH WAS &
        ! SPECIFIED. IF IT WAS, EXTRACT IT FOR LATER PROCESSING. &

1005    OPEN OBJ$ FOR INPUT AS FILE 1%, ORGANIZATION SEQUENTIAL VARIABLE, &
                MAP OBJFIL &
        ! OPEN UP THE FILE. &

1u10    GET #1% &
\       GOTO 1010 IF (REC%(0%) AND 255%) <> 1% OR RECOUNT<10% &
\       CNT%=RECOUNT &
\       DIRTY%=0% &
\       FOR WORD%=1% TO CNT%/2%-1% STEP 4% &
\       GOTO 1020 IF ((REC%(WORD%+2%) AND -256%)<>1280%) OR RAD$(REC%(WORD%)) &
                <>"BP2" &
                OR RAD$(REC%(WORD%+1%))<>"OTS" &
\       REC%(WORD%+2%)=REC%(WORD%+2%) OR 16% &
\       REC%(WORD%+2%)=REC%(WORD%+2%) AND -17% IF RW% &
\       DIRTY%=-1% &
        ! READ A RECORD. GET ANOTHER ONE IF IT ISN'T POSSIBLY GOING TO CONTAIN &
        ! .PSECT INFORMATION. STORE THE LENGTH OF THIS RECORD AND SET THE &
        ! DIRTY FLAG TO FALSE. READ THROUGH ALL THE ENTRIES, LOOKING FOR &
        ! SOMETHING CALLED "BP2OTS". IF IT IS FOUND, SET THE "RO" BIT IN &
        ! THE .PSECT ATTRIBUTES. CHANGE THAT RO TO RW IF WE ARE UNPATCHING &
        ! THE OTS. FLAG THIS RECORD AS DIRTY. &

1020    NEXT WORD% &
\       UPDATE #1%, COUNT CNT% IF DIRTY% &
\       GOTO 1010 &
        ! LOOP FOR THE REST OF THE ENTRIES IN THIS RECORD. WRITE IT BACK &
        ! TO DISK IF DIRTY. GET ANOTHER RECORD &

19000   ! ERROR TRAP &

19011   IF ERR=11% &
        THEN    CLOSE 1% &
        \       RESUME 32767 &
        ! HANDLE EOF ERRORS &

19999   ON ERROR GOTO 0 &
        ! ALL OTHER ERRORS ARE FATAL AND UNEXPECTED &

32767   END &
```



```
                    [1,4] FIXBP2.ODL


        .ROOT BASIC2-RMSROT-USER,RMSALL
USER:   .FCTR   SY:FIXBP2-LIBR
LIBR:   .FCTR   LB:CSPCOM/LB
BASIC2: .FCTR   LB:CSPCOM/LB:$ICIO2-BP2RM1-BP2RM2-BP2SEQ-BP2SCR
BP2RM1: .FCTR   LB:CSPCOM/LB:$IMROT:$IMERR:$IMALQ-BP2R11
BP2R11: .FCTR   LB:CSPCOM/LB:$IMALL:$IMFSS:$IMATR
BP2RM2: .FCTR   LB:CSPCOM/LB:$IMOPN:$IMCLS:$IMFGC-BP2R22
BP2R22: .FCTR   LB:CSPCOM/LB:$IMPUT:$IMUPD:$IMULK:$IMRES
BP2SEQ: .FCTR   LB:CSPCOM/LB:$ISROT:$ISOPN:$ISFGC:$ISPUT-BP2SE1
BP2SE1: .FCTR   LB:CSPCOM/LB:$ISUPD
BP2SCR: .FCTR   LB:CSPCOM/LB:$ISSCR
@LB:RMS11X
        .END
```



```
        [1,4] FIXBP2.CMD


SY:FIXBP2=SY:FIXBP2/MP
UNITS = 12
ASG = SY:5:6:7:8:9:10:11:12
EXTTSK= 952
//
```



```
          [1,4] CSPLIB.CMD
;       This file builds a resident
library tailored for the CUSPS
CSPCOM/-HD,CSPCOM,CSPCOM=
LB:CSPCOM/LB:$BBTKS
LB:CSPCOM/LB:$BFPER
LB:CSPCOM/LB:$BINIT
LB:CSPCOM/LB:$BTDID
LB:CSPCOM/LB:$BXTRA
LB:CSPCOM/LB:$CHANG
LB:CSPCOM/LB:$CNTRL
LB:CSPCOM/LB:$ECONV
LB:CSPCOM/LB:$ERROR
LB:CSPCOM/LB:$FADD
LB:CSPCOM/LB:$FADDM
LB:CSPCOM/LB:$FCMP
LB:CSPCOM/LB:$FCON1
LB:CSPCOM/LB:$FDIV
LB:CSPCOM/LB:$FMOV
LB:CSPCOM/LB:$FMUL
LB:CSPCOM/LB:$FTIME
LB:CSPCOM/LB:$FUNC1
LB:CSPCOM/LB:$FUNC2
LB:CSPCOM/LB:$ICEND
LB:CSPCOM/LB:$ICEOL
LB:CSPCOM/LB:$ICFLD
LB:CSPCOM/LB:$ICFNS
LB:CSPCOM/LB:$ICFSS
LB:CSPCOM/LB:$ICINI
LB:CSPCOM/LB:$ICRCL
LB:CSPCOM/LB:$ICRED
LB:CSPCOM/LB:$ICRFG
LB:CSPCOM/LB:$ICROP
LB:CSPCOM/LB:$ICRPT
LB:CSPCOM/LB:$ICULT
LB:CSPCOM/LB:$ICWRT
LB:CSPCOM/LB:$IECHN
LB:CSPCOM/LB:$IEMSC
LB:CSPCOM/LB:$IESYS
LB:CSPCOM/LB:$IEULT
LB:CSPCOM/LB:$IVOPN
LB:CSPCOM/LB:$IVVIR
LB:CSPCOM/LB:$JADDS
LB:CSPCOM/LB:$JCMPS
LB:CSPCOM/LB:$JCOMP
LB:CSPCOM/LB:$JCON1
LB:CSPCOM/LB:$JCONV
LB:CSPCOM/LB:$JDIVS
LB:CSPCOM/LB:$JMOVS
LB:CSPCOM/LB:$JMUL
LB:CSPCOM/LB:$JNCR
LB:CSPCOM/LB:$JNEXT
LB:CSPCOM/LB:$JSUBS
LB:CSPCOM/LB:$JXII
LB:CSPCOM/LB:$LOADS
LB:CSPCOM/LB:$NEGAT
LB:CSPCOM/LB:$SSFLT
LB:CSPCOM/LB:$SSINT
```

MENU
THE MISSING LINK.

MENU is an applications development aid that provides the DEC RSTS software developer with a powerful process control device.

MENU easily generates menus that guide the application user into selected programs based on his security level.

MENU supplements RSTS security by allowing multi-level access capabilities within an individual account. Separate project-level control minimizes System Manager interaction for system level security.

MENU is driven by simple text files which determine extent of program control, type and level of security, screen displays, and presentation of on-line '/HELP' information.

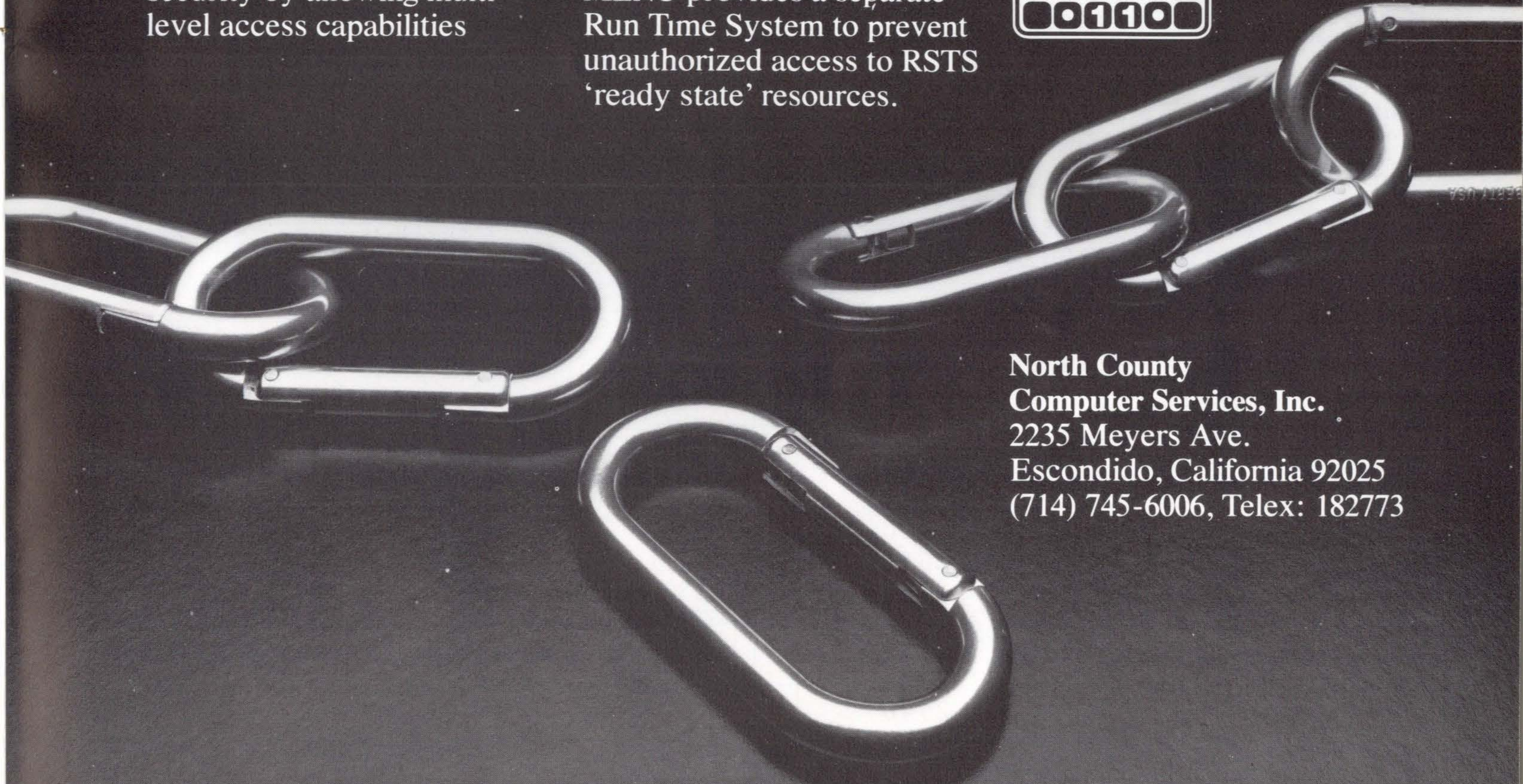
MENU provides a separate Run Tıme System to prevent unauthorized access to RSTS 'ready state' resources.

MENU installs in minutes and requires no software modifications.

MENU provides a common interface for all your users and application needs.

For information on **MENU,** please give us a call. We would be delighted to show you the missing link.

NCCS
USER
11



North County
Computer Services, Inc.
2235 Meyers Ave.
Escondido, California 92025
(714) 745-6006, Telex: 182773

DEC and RSTS are registered trademarks of Digital Equipment Corporation.

© Copyright NCCS

CIRCLE 38 ON READER CARD

```
LB:CSPCOM/LB:$SSVIR
LB:CSPCOM/LB:$STCFS
LB:CSPCOM/LB:$STCMP
LB:CSPCOM/LB:$STCOS
LB:CSPCOM/LB:$STCVT
LB:CSPCOM/LB:$STFN1
LB:CSPCOM/LB:$STFNS
LB:CSPCOM/LB:$STGTA
LB:CSPCOM/LB:$STLSS
LB:CSPCOM/LB:$STMOS
LB:CSPCOM/LB:$STMSC
LB:CSPCOM/LB:$STNMF
LB:CSPCOM/LB:$STPDB
LB:CSPCOM/LB:$STVLF
LB:CSPCOM/LB:$STXCM
LB:CSPCOM/LB:$TESTS
LB:CSPCOM/LB:$TIME
LB:CSPCOM/LB:RQLCB
LB:CSPCOM/LB:SAVRG
LB:CSPCOM/LB
/
STACK=0
PAR=CSPCOM:140000:40000
//
```

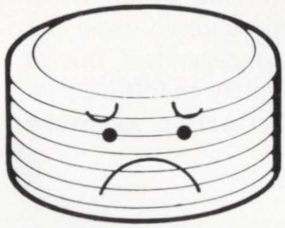
[1,4] FIXCSP.CMD

```
! Correct $CODE and $PDATA
attributes in CSPCOM.TSK
File to patch?
Base address? 551:440
Offset address? 0
 Base   Offset  Old     New?
000440  000000  002750  ? 2740
000440  000002  000000  ? ^Z
Offset address? 10
 Base   Offset  Old     New?
000440  000010  002750  ? 2740
000440  000012  000002  ? ^Z
Offset address? 20
 Base   Offset  Old     New?
000440  000020  002750  ? 2740
000440  000022  000000  ? ^Z
Offset address? ^Z
Base address? 551:654
Offset address? 0
 Base   Offset  Old     New?
000654  000000  002650  ? 2660
000654  000002  002650  ? 2640
000654  000004  002650  ? 2640
000654  000006  002450  ? 2460
000654  000010  002650  ? 2640
000654  000012  002650  ? 2640
000654  000014  002650  ? 2640
000654  000016  000001  ? ^Z
Offset address? ^Z
Base address? 551:572
Offset address? 0
 Base   Offset  Old     New?
000572  000000  002650  ? 2640
0005/2  000002  000002  ? ^Z
Offset address? 10
 Base   Offset  Old     New?
000572  000010  002754  ? 2744
000572  000012  000200  ? ^Z
Offset address? ^Z
Base address? ^Z
File to patch? ^Z
```

```
*H/2!/V<cr>
2!<tab><tab>PROGRAM<tab><tab>: BUILD.BAS<cr>
*H/2250<tab>/V<cr>
2250<tab>! &<cr>
*7AV<cr>
<tab><tab>GOTO 2250 UNLESS KBM% &<cr>
*I<cr>
<tab><tab>PRINT NOLIB$ IF RESLIB% AND RESLIB$=NL$ &<cr>
<tab>\<tab>GOTO 2250 IF (KBM%=0%) OR (RESLIB% AND RESLIB$=NL$) &<cr>
<esc>*KV<cr>
<tab>\<tab>PRINT RTSD$ UNLESS RTS.TYP% &<cr>
*G/KBM% AND RTS.TYP%/V<cr>
<tab><tab><tab>IF KBM% AND RTS.TYP% THEN &<cr>
*I/ AND (RESLIB%=(RESLIB$=NL$))/V<cr>
<tab><tab><tab>IF KBM% AND RTS.TYP% AND (RESLIB%=(RESLIB$=NL$)) THEN &<cr>
*H/2280<tab>/V<cr>
2280<tab>PRINT RTSD$ IF KBM% &<cr>
*I<cr>
<tab><tab>PRINT NOLIB$ IF RESLIB% AND RESLIB$=NL$ &<cr>
<tab>\<tab><tab><esc>*V<cr>
<tab>\<tab><tab>PRINT RTSD$ IF KBM% &<cr>
*H/10000<tab>/V<cr>
10000<tab>! &<cr>
*33AV<cr>
  \ RTSD$="Run-Time System must have a default extension of '.BAC' or '.TSK'" &<cr>
*I<cr>
<tab>\ URES$="Use the CUSP resident library" &<cr>
<tab>\ LNAM$="Resident library name" &<cr>
<tab>\ NOLIB$="Resident library can not be used" &<cr>
<esc>*V<cr>
  \ RTSD$="Run-Time System must have a default extension of '.BAC' or '.TSK'" &<cr>
*H/15860<tab>/V<cr>
15860<tab>! &<cr>
*G/TKB%<>0%/AV<cr>
<tab>\ S$=CSP$ UNLESS FNCSPCOM%(S$) <tab>IF CSP% &<cr>
*I<cr>
<tab>\ MU%=0% &<cr>
<tab>\ MU%=INSTR(1%,S$,"/MU") IF FNR%("COMPILE",-3%) &<cr>
<tab>\ P%=0% &<cr>
<tab>\ S$=LEFT(S$,MU%-1%)+RIGHT(S$,MU%+3%) IF MU% &<cr>
<tab>\ MU%=MU%<>0% &<cr>
<tab>\ TKB%=TKB% OR MU% &<cr>
<esc>*V<cr>
<tab>\ S$=CSP$ UNLESS FNCSPCOM%(S$) <tab>IF CSP% &<cr>
*H/15900<tab>/V<cr>
15900<tab>! &<cr>
*39AV<cr>
<tab>\<tab>PRINT #12%,CVT$$(FNFLMRG$(T0$,".TSK",7%)+"/FP="+T0$+","+ &<cr>
*3J17C/T1$=/G/FP/11C/"/V<cr>
<tab>\<tab>T1$=FNFLMRG$(T0$,".TSK",7%)+"/FP" &<cr>
*AV<cr>
<tab><tab><tab>LB$+LIB$+".OLB/LB",2%)+CRLF$+"/" &<cr>
*I<cr>
<tab>\<tab>T1$=T1$+"/MU" IF MU% &<cr>
<tab>\<tab>T1$=T1$+"="+T0$+","+<esc>*3DV<cr>
<tab>\<tab>T1$=T1$+"="+T0$+","+LB$+LIB$+".OLB/LB",2%)+CRLF$+"/" &<cr>
*2G/"/I<cr>
 &<cr>
<tab>\<tab>PRINT #12%,CVT$$(T1$<esc>*V<cr>
<tab>\<tab>PRINT #12%,CVT$$(T1$,2%)+CRLF$+"/" &<cr>
*3AV<cr>
<tab>\<tab>PRINT #12%,"UNITS=12";CRLF$; &<cr>
*I<cr>
<tab>\<tab>PRINT #12%,"RESLIB=";RESLIB$;"/RO" IF RESLIB% AND (TKB%=MU%) &<cr>
<esc>*V<cr>
<tab>\<tab>PRINT #12%,"UNITS=12";CRLF$; &<cr>
*6AV<cr>
<tab>\<tab>PRINT #12%,RPIP$;CRLF$; &<cr>
*I<cr>
<tab>\<tab>IF MU% &<cr>
<tab>THEN<tab>PRINT #12%,"RUN ";RUNLIB$;"MAKSIL";CRLF$; &<cr>
<tab><tab><tab>FNFLMRG$(T0$,NL$,4%);CRLF$; &<cr>
<tab><tab><tab>CRLF$; &<cr>
<tab><tab><tab>"No";CRLF$; &<cr>
<tab><tab><tab>CRLF$; &<cr>
<tab><tab><tab>RPIP$;CRLF$; &<cr>
<tab><tab><tab>T2$;"/DE";CRLF$; &<cr>
<tab><tab><tab>FNFLMRG$("<124>",T2$,16%); &<cr>
<tab><tab><tab>"=";FNFLMRG$(T0$,".SIL",7%);"/RE";CRLF$; &<cr>
<tab><tab><tab>"^Z" &<cr>
<tab>! IF THE /MU SWITCH WAS PRESENT, RUN MAKSIL FOR THE USER AND &<cr>
<tab>! CONVERT THE TASK TO A .LIB AND .SIL. &<cr>
<tab>ELSE<tab><tab><esc>*3DV<cr>
<tab>ELSE<tab><tab>PRINT #12%,RPIP$;CRLF$; &<cr>
*H/16940<tab>/V<cr>
16940<tab>RTS.EXT$=RAD$(PEEK(RTS.LNK%+6%)) &<cr>
*20AV<cr>
<tab>&<cr>
*I<cr>
<tab>\<tab><tab>IF CSP% &<cr>
<tab><tab><tab>THEN<tab>RESLIB%=ASCII(CVT$$(FNF$(URES$,"Yes",8192%,NL$) &<cr>
<tab><tab><tab><tab><tab>,-1%))=89% &<cr>
<tab><tab><tab>\<tab>IF RESLIB% &<cr>
<tab><tab><tab><tab>THEN<tab>RESLIB$=FNF$(LNAM$,"LB:CSPCOM",28%,NL$) &<cr>
<tab><tab><tab><tab>\<tab>IF FNFLNMCK$(RESLIB$+".TSK",16397%,NL$)=NL$ &<cr>
<tab><tab><tab><tab><tab>OR FNFLNMCK$(RESLIB$+".STB",16397%,NL$)=NL$ &<cr>
<tab><tab><tab><tab><tab>THEN<tab>RESLIB$=NL$ &<cr>
<tab><tab><tab><tab><tab>\<tab>RTS.TYP%=0% <esc>*DV<cr>
<tab><tab><tab><tab><tab>\<tab>RTS.TYP%=0% &<cr>
*8AV<cr>
<tab><tab>! CSP%  -1% IF WANT TO USE CSPCOM ELSE 0 &<cr>
*AV<cr>
<cr>
*I<cr>
<tab><tab>! IF WE WANT TO USE CSPCOM, ASK ABOUT THE CUSP RESLIB. &<cr>
<tab><tab>! IF WE WANT THE CUSP RESLIB, GET THE NAME AND LOOK UP THE &<cr>
<tab><tab>! .TSK AND .STB FILES FOR THE RESLIB. IF EITHER AREN'T &<cr>
<tab><tab>! FOUND, CLEAR THE RESLIB NAME AND SET THE RTS TO BE NOT FOUND. &<cr>
<tab><tab>! THIS WILL BE CHECKED LATER. &<cr>
<esc>*V<cr>
<cr>
*EX<cr>
```

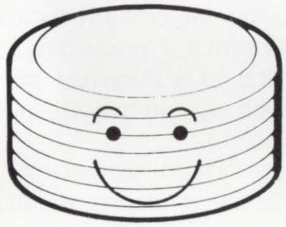
♥

OPTIMIZE your RSTS/E* disks with DSKBLD.



FILING PROBLEMS YOU MAY BE PUTTING UP WITH.

- Fragmented UFD's
- Poor file clustersizes.
- Scattered files.
- Excessive arm movement.
- Excessive FIP overhead.
- Disk bound response problems.



DSKBLD HAS THE SOLUTIONS.

- Contiguous UFD's.
- Organized files.
- Optimum clustersizes.
- Minimized FIP overhead.
- Reduced arm movement.
- Lower disk access overheads.

Overtime for disk management is greatly reduced or eliminated. Disks get rebuilt more often because it's painless.

DSKBLD is an extremely fast RSTS/E disk-to-disk utility which:

□ Supports mixed disk types.

□ Untangles directory links.

□ Creates contiguous UFD's to proper size.

□ Creates accounts in sorted order.

□ Locates files adjacent to owner's UFD.

□ Optimizes file clustersizes.

□ Saves accounting data.

□ Is easy to use and fully documented.

□ Provides logging and statistics.

□ Is inexpensive.

INTRODUCTORY PRICE $500.

*RSTS/E is a registered trademark of Digital Equipment Corporation.

MANUS
SERVICES CORPORATION

AUTHORIZED
digital
COMPUTER DISTRIBUTOR

HOME OFFICE CONTACT:
Gary Vowels
1700 Westlake Avenue N.
Lake Union Building
Seattle, Washington 98109
206-285-3260

PORTLAND OFFICE:
Don Shafer
9498 S.W. Barbur Blvd.
Portland, Oregon 97219
503-245-1261

□ Send me more information on the DSKBLD RSTS/E Disk.

Name ____________________

Title ____________________

Company ____________________

Address ____________________

City ____________________

State ____________ Zip ____________

Telephone ____________________
(Area) (Number) (Ext.)

CIRCLE 39 ON READER CARD

OPERATIONAL SYSTEM STATUS

Gary W. Miller, System Software Manager, Garden Way Manufacturing

This paper describes a system status program that is used by our operational staff. This program seems to give all the information needed about the system without the problems encountered using the VT50PY or some other similar program. This program is 'not' system or terminal independent. Due to the massive number of PEEK's and other system table references, any changes by DEC to RSTS/E's internal structure could make this program useless. Even with these restrictions, I feel that a program like this could be helpful to anyone managing a RSTS/E system.

Before I get down to the bells and whistles of the program I think a little history would be appropriate. The idea for this program came from my exposure to an 'IBM' devotee who said RSTS/E could not be a very good system if it didn't allow you to monitor the system and use the terminal at the same time. Well, whenever I hear someone say that some hog of an 'IBM' system is better than my RSTS/E system, I see red! So, to prove that anything he could do I could do better, I sat down one evening and started to code. As the work progressed I was very surprised at the information I could pull out of the system and then display. Truthfully, I was afraid that I had bitten off more than I could chew, but to fail in a test against 'IBM' was unacceptable. The first test version was ready in about two days, and after some minor adjustments the program was working. After that, the changes that have been made have only added functionality and have not changed the original program design. I hope that this paper will inspire some thought amongst the RSTS/E community as to what kind of management utilties could be available and maybe someone would be willing to approach DEC to pass along those ideas.

The functional description of the program is somewhat vague since the display is very full, but I will make an attempt. The program's name is GSYS and is written in BASIC-PLUS for use on RSTS/E version 7 with a monitor configured for large files. When running on the system the program is 13K words and ties up two job slots. The terminal connected to the program must be a VT100 with an advanced video option installed. The set-up should have the AUTO XON bit turned on to allow the program to turn off the keyboard during a display. When the program starts out it assumes that the VT100 is set up as a VT52, part of the initialization changes the set-up to be ANSI and sets the width to 132. At termination the terminal's set-up is restored to the original values. At this writing a further enhancement has been added to allow an abbreviation library using the key-pad mode of the terminal.

For each job on the system, the job's PPN, job status, job name and keyboard number is displayed on the screen. For a detached job 'DET' is displayed in place of the keyboard number and for a job running on a PK the controling job's job number is displayed with a 'J' preceeding it in place of the keyboard number. Should a detached job go into hibernation, then the entire job area on the screen will flash to draw attention to the problem.

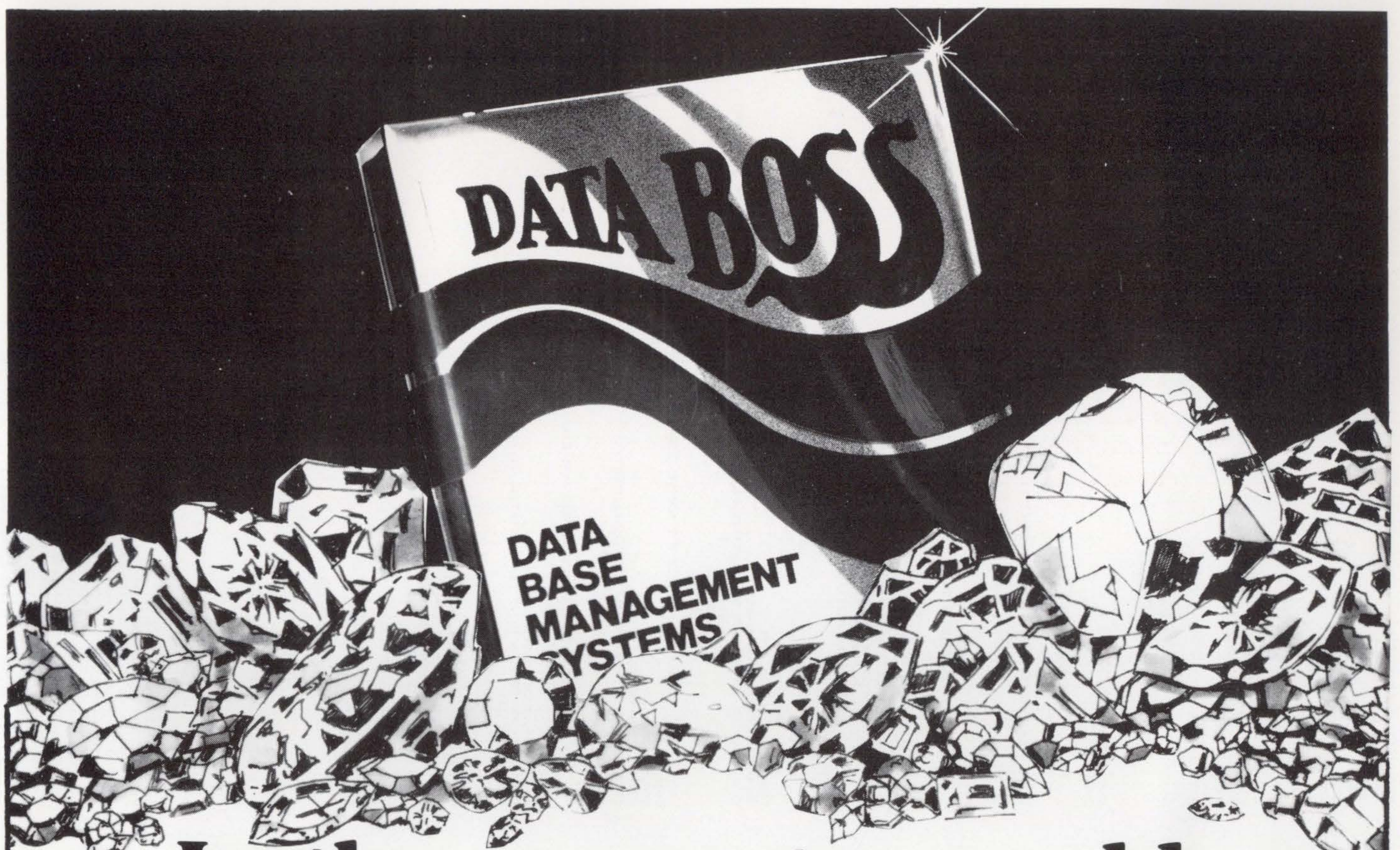
For each disk drive on the system, the pack ID, number of opened files, and free space is displayed. Should the number of errors on a drive exceed ten, then the open files count will flash to point out the situation. If the number of free blocks on a disk drop below 11,000 then the free space number will flash.

To point out a problem that most large RSTS/E users have, the small buffer count is displayed on the screen, and if the count drops below 100 then the number will flash. The number of jobs and the maximum number of jobs are also displayed on the screen and when the system is full these numbers will flash. The rest of the display is made up of the first 10 message receivers. On each receiver the job name, job number, and receiver ID is displayed. On the line printer and batch spoolers a special text is also displayed if a file is being printed or a batch stream is being run. For line printers this text contains the file name being printed, the file size, and what block is currently being printed.

The rest of the screen is set up as scroll region for output from a PK. The operator can type any command on the screen and see it execute in the scroll region. The output of the PK can also be directed to a file if an audit trail of the dialog is required.

To minimize typing the program function keys along with the special key-pad mode are used to create an abbreviation library for use by the operators, this library is set up by data statements in the program. These commands are both full commands as well as partials where the operator must fill in the rest of the command.

This program has been in use now for about five months and the response to it has been enthusiastic from all concerned. I am also pleased to add that my 'IBM' contact is now eating a lot of crow since his system can't do what my program does. Should anyone be interested in my program I could be convinced to let it loose as long as I don't have to maintain it.

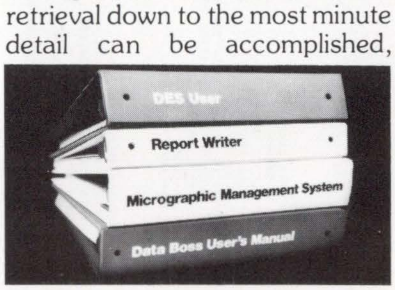


In the computer world, one gem stands a cut above the rest.

DATA BOSS is rapidly becoming the world's most valuable data base management system. DATA BOSS captured Datapro's highest rating in a user evaluation survey, scoring a perfect 4.0 mark in overall satisfaction. The reason is simplicity and reliability.

The simplicity of DATA BOSS/2 and DATA BOSS/32 systems saves you time and money. DATA BOSS operations can be learned within several hours by personnel without computer programming backgrounds.

DATA BOSS can be implemented in a fraction of the time normally required for an application. Up to 90% of your coding time is eliminated, and fast retrieval down to the most minute detail can be accomplished, without time-consuming sorting or re-programming.



DATA BOSS/2 and **DATA BOSS/32** are available under RSTS/E and the VMS operating systems on Digital Equipment Corporation PDP-11 and VAX-11. Its valuable options include a Data Entry Subsystem (DES) and a Report Writer and Micrographics Management System (MMS).

Explore the possibilities of joining over 100 worldwide users of DATA BOSS; Fortune 500 corporations, hospitals, retail conglomerates, utilities, and dozens of other firms.

Please write for our full catalog. If information is essential to your firm, this information will be essential to your success.

Florida Computer, Inc.
99 N.W. 183rd Street, No. Miami, Florida 33169 (305) 652-1710

Turnkey

Turnkey Software Limited, 12 High Street, Chalfont St. Giles, Bucks HP8 4QA, Telephone: 02407 5995/3410, Telex: 24224 ref:Turnkey 3003

CIRCLE 40 ON READER CARD

```
1       !>>PROGRAM     : GSYS.BAS
        !  EDIT DATE   : 00-XXX-00
        !  VERSION     : 1.0
2       !  RSTS V07
        !  CREATED ON : 19-Aug-80
        !  AUTHOR      : GARY/MILLER
        !  DESCRIPTION: THIS PROGRAM WILL ALLOW THE OPERATOR TO KEEP TRACK OF WHAT IS GOING
        !               ON THE SYTEM WITH ONE SCREEN. THIS WILL WORK ONLY ON A VT100 WITH
        !               THE ADVANCED VIDEO OPTION INSTALLED.
4       EXTEND
5       !>>STANDARD FEATURES:
                SCREEN FUNCTIONS

6       !>>CH-FILENAME-----ACCESS-------MNEMONIC
            1-U                         <UNUSED>
            2-U                         <UNUSED>
            3-U                         <UNUSED>
            4-U                         <UNUSED>
            5-U                         <UNUSED>
            6-U                         <UNUSED>
            7-U                         <UNUSED>
            8-U                         <UNUSED>
            9-U                         <UNUSED>
           10-U                         <UNUSED>
           11-U                         <UNUSED>
           12-COMMAND KB:
8       !****************************************************************
        !                 PROGRAM MODIFICATIONS
        !
        !
        !****************************************************************

12      DIM PARTI%(30%),PARTII%(30%),RC$(10%,3%)        !>>INCORE DIM'S

13        DIM M%(30%),SCR$(20%),JOB%(63%)

14        DIM FTBL$(4%,13%),FTBL%(4%,13%)

                         !>>SPECIAL FUNCTION TABLE


25      P9$="GSYS"
        \ ON ERROR GOTO 32000
        \ Z$=SYS(CHR$(6%)+CHR$(-7%))
        \ N$=""
        \ L$=CHR$(13%)+CHR$(10%)
        \ KB%=12%
        \ EOS%=-1%
        \ EOL%=-2%                              !>>PROGRAM NAME
                                                !  TRAP ERRORS
                                                !  ENABLE ^C TRAP
                                                !  N$=NULL STRING
                                                !  L$=<CR><LF>
                                                !  E0$=FILE-OPENING ERROR STRING
                                                !  E1$=STANDARD PROMPT ERROR STRING
                                                !  KB%=KEYBOARD CHANNEL
                                                !  EOS%=CLEAR TO EOS
                                                !  EOL%=CLEAR TO EOL
30      OPEN "KB:" AS FILE KB%
        \ X3%=1%
        \ V%=95%
                                                !>>OPEN KEYBOARD FOR INPUT
                                                !  CURSOR POSITIONING
                                                !  PAINT CHARACTER=UNDERSCORE
90      PRINT #KB%, L$; FNV$(1%,1%,N$,EOS%,V%);
                FNV$(1%,1%,P9$+"  "+DATE$(D9%)+"  "+TIME$(0%),0%,V%)
                                                !>>PRINT THE BANNER
                                                !  PRELIMINARY INPUT ROUTINES FOLLOW
500       RC$(K%,0%)=SPACE$(6%)+'' FOR K%=1% TO 10%
        \ RC$(K%,2%)=SPACE$(6%)+'' FOR K%=1% TO 10%
        \ RC$(K%,1%)=SPACE$(2%) FOR K%=1% TO 10%
        \ RC$(K%,3%)=SPACE$(37%) FOR K%=1% TO 10%


                                !>>CLEAR RECEIVER ARRAY


999                                             !
                                                !>>BEGIN MAINLINE PROCESSING
                                                !
1000      RC%(K%)=-1% FOR K%=1% TO 10%
        \ JOB%(K%)=-1% FOR K%=1% TO 63%

                         !>>SET ACTIVE FLAG


1020            CHANGE SYS(CHR$(6%)+CHR$(-3%)) TO PARTI%
        \       PARTI%(J%)=PARTI%(J%)+SWAP%(PARTI%(J%+1%)) FOR J%=5% TO 29% STEP 2%
        \       CHANGE SYS(CHR$(6%)+CHR$(-12%)) TO PARTII%
        \       PARTII%(J%)=PARTII%(J%)+SWAP%(PARTII%(J%+1%)) FOR J%=3% TO 29% STEP 2%



        !                GET MONITOR TABLES ... FUN ISN'T THIS ....



1030            CNT.KB%=PARTI%(3%)
        \       MAXCNT%=PARTI%(4%)
        \       DEVCNT%=PARTI%(5%)
        \       DEVPTR%=PARTI%(7%)
        \       MEMLST%=PARTI%(9%)
        \       JOBTBL%=PARTI%(11%)
        \       JBSTAT%=PARTI%(13%)
        \       JBWAIT%=PARTI%(15%)
        \       UNTCLU%=PARTI%(17%)
        \       UNTCNT%=PARTI%(19%)
        \       SATCTL%=PARTI%(21%)
        \       JSBTBL%=PARTI%(23%)
```

```
        \ WAIT 0%
        \ GOTO 2000% IF E%=-8192%
        \ PRINT #KB%,CHR$(155%)+'=';
        \ CMD$=IO$+''
        \ GOSUB 15000%
        \ PRINT #KB%,CHR$(155%)+'>';
        \ GOTO 2010 IF E%


                                !>>WAIT FOR COMMAND


2020      GOTO 4950 IF CMD$='ABORT'
        \ GOTO 3100 IF CMD$='ECHO' OR CMD$='GSYS'
        \ GOSUB 14120%
        \ T$=CMD$
        \ GOSUB 14030%
        \ GOTO 2010


                         !>>WAIT FOR COMMAND


3000      ECHO%=NOT(ECHO%)
        \ GOTO 2010

                         !>>RESET ECHO


3100      PRINT #KB%,FNV$(1%,24%,N$,0%,V%);CHR$(155%)+'[K';
        \ IF CMD$='ECHO' THEN 3000% ELSE
          IF CMD$='GSYS' THEN 2000% ELSE
          GOTO 2010


                         !>>COMMAND'S THAT "GSYS" UNDERSTANDS


4900    E%=FNI1%("TYPE <CR> TO EXIT .. ",4%)
        \STOP IF IO$="STOP"             !>>ABORT TO HERE ON ABNORMAL EXIT
4950      PRINT #KB%,FNSCROLL$(1%,24%);CHR$(155%)+'[?31';
        \ PRINT #KB%,FNV$(1%,1%,N$,EOS%,V%);L$ ;
                FNV$(20%,2%,"END OF "+P9$,0%,V%);
        \ PRINT #KB%,CHR$(155%)+'[?21';
        \ GOSUB 10020%
        \ GOTO 32760                    !>>EXIT POINT
                                        !  SUBROUTINES FOLLOW


5000      FOR JOB%=1% TO MAX.JOB%
        \ JDB%=PEEK(JOBTBL%+JOB%+JOB%)
        \ IF JDB%=0% OR JDB%=-1% THEN  PRINT#KB%,FNJOB$(JOB%);FNATTR$('7');SPACE$(25%); IF JOB%(JOB%)
        \                        JOB%(JOB%)=0%
        \                        GOTO 5100


                         !>>LOOK FOR JOB IF NOT ACTIVE CLEAR ENTRY ON SCREEN
5010      UNIBUS%=0%
        \ GOSUB 13000%
        \ GOSUB 12000% UNLESS UNIBUS%
        \ IF UNIBUS% THEN PRINT #KB%,FNJOB$(JOB%);FNATTR$('7');SPACE$(25%); IF JOB%(JOB%)
        \                        JOB%(JOB%)=0%
        \                        GOTO 5100

                         !>>GET DATA IF JOB DRIFTED AWAY CLEAR ENTRY


5020      PRINT #KB%,FNJOB$(JOB%);FNATTR$('7');
        \ PRINT #KB%,FNATTR$('5'); IF LEFT(JOSTATUS$,2%)='HB'
        \ JOS$=SPACE$(2%)
        \ LSET JOS$=JOSTATUS$
        \ JOSTATUS$=JOS$+''
        \ TTY$=NUM1$(JOB.TTY%)
        \ TTY$='J'+NUM1$(KB.OWNER%) IF TTY.TYPE$='PK'
        \ TTY$=JOB.DET$ IF JOB.DET$='DET'
        \ JOB.PPN$=NUM1$(SWAP%(JOB.PPN%) AND 255%)+','+NUM1$(JOB.PPN% AND 255%)
        \ PRINT #KB%,USING '## \     \ \ \ \      \ \\',
                JOB%,JOB.PPN$,TTY$,
                JOB.NAME$,JOSTATUS$;

        \ PRINT #KB%,FNATTR$('0');FNATTR$('7');
        \ JOB%(JOB%)=-1%

                         !>>PRINT JOB INFO


5100      UNIBUS%=0%
        \ NEXT JOB%
        \ PRINT #KB%,FNATTR$('0');
        \ RETURN


                         !>>THATS ALL OF SYS'TAT'


6000      FOR DISK%=1% TO DC%
        \ FIP.UNIT%=(DISK%-1%)*2%
        \ PACKID.ENTRY%=(FIP.UNIT%*5%)+LOGNAM%
        \       PACKID$=RAD$(PEEK(PACKID.ENTRY%))+RAD$(PEEK(PACKID.ENTRY%+2%))
        \       PRINT #KB%,FNDISK$(DISK%);PACKID$;' ';
        \       PACK.STATUS%=PEEK(UNTCNT%+FIP.UNIT%)
        \       IF PACK.STATUS%<0% THEN PRINT #KB%,SPACE$(11%); ELSE
                OPEN.FILES%=PACK.STATUS% AND 1023%
        \       FREE.BLOCKS=(32768.+(PEEK(SATCTL%+FIP.UNIT%) EQV 32767%)) +
                        (PEEK(SATCTM%+FIP.UNIT%)*65536.)
        \       DISK.ERROR%=SWAP%(PEEK(UNTCLU%+FIP.UNIT%)) AND 255%
        \       PRINT #KB%,FNATTR$('1;5'); IF FREE.BLOCKS<11000.
        \       PRINT #KB%,USING '###### ',FREE.BLOCKS;
        \       PRINT #KB%,FNATTR$('0');
        \       PRINT #KB%,FNATTR$('1;5'); IF DISK.ERROR%>10%
        \       SATCTM%=PARTI%(25%)
        \       SYSTEM.DATE%=PARTI%(27%)
        !               MOVE DATA INTO FAMILIAR NAMES (RSTS/E PROGRAMMERS MANUAL)
                        SECTION 7.2.17.1


1040            FREES%=PARTII%(3%)
        \       DEVNAM%=PARTII%(5%)
        \       CSRTBL%=PARTII%(7%)
        \       DEVOKB%=PARTII%(9%)
        \       TTYHCT%=PARTII%(11%)
        \       JOBCNT%=PARTII%(13%)
        \       RTSLST%=PARTII%(15%)
        \       ERLCTL%=PARTII%(17%)
        \       SNDLST%=PARTII%(19%)
        \       LOGNAM%=PARTII%(21%)
        \       DEVSYN%=PARTII%(23%)
        \       MEMSIZ%=PARTII%(25%)
        \       CCLLST%=PARTII%(27%)
        \       FCBLST%=PARTII%(29%)


        !               MOVE REST OF DATA TO FAMILIAR NAMES SECTION 7.2.17.2


1050            FOR K%=DEVNAM%+DEVOKB% TO DEVSYN%-2% STEP 2%
        \       IF CVT%$(SWAP%(PEEK(K%)))='PK' THEN
                PKDEV%=PEEK(DEVPTR%+K%-DEVNAM%)
        \       PK0%=PEEK(PEEK(PKDEV%)+8%)/2%
        \       GOTO 1070


        !       FIND 'PK0:' AND SAVE REAL TERMINAL NUMBER TO
        !       ADJUST FOR PK NUMBERS.


1060            NEXT K%


        !       LOOP THRU TABLES .....


1070      DC%=0%
        \ FOR DISK.ENTRY%=0% TO DEVOKB%-2% STEP 2%
        \       DISK.TYPE$=CVT%$(SWAP%(PEEK(DEVNAM%+DISK.ENTRY%)))
        \               FOR DISK.UNIT%=0% TO PEEK(DEVCNT%+DISK.ENTRY%)
        \               DC%=DC%+1%
        \               DISK.NAME$(DC%)=DISK.TYPE$+NUM1$(DISK.UNIT%)+':'
        \               NEXT DISK.UNIT%
        \ NEXT DISK.ENTRY%
                                        !>>BUILD DISK NAME TABLE FOR
                                        !  EACH UNIT.


1080      MAX.JOB%=SWAP%(PEEK(JOBCNT%)) AND 255%
        \ CHANGE SYS(CHR$(6%)+CHR$(14%)) TO M%
        \ PASWRD$=RAD$(M%(9%)+SWAP%(M%(10%)))+RAD$(M%(11%)+SWAP%(M%(12%)))
        \ PPN$=NUM1$(M%(8%))+'/'+NUM1$(M%(7%))
        \ T$='HELLO '+PPN$
        \ GOSUB 14000%
        \ GOSUB 10000%
        \ GOSUB 14030%
        \ T$=PASWRD$
        \ GOSUB 14030%
        \ T$=' '
        \ GOSUB 14030%
        \ ECHO%=-1%


                                !>>LOG PK INTO CORRECT ACCOUNT


1090      COL.MAX%=(MAX.JOB%+4%)/5%
        \  SYS.START%=1%
        \ SYS.OFF%=26%
        \ DISK.START%=1%
        \ DISK.OFF%=26%
        \ REC.OFF%=55%
        \ REC.START%=23%
        \ SCROLL.START%=COL.MAX%+DC%+3%
        \ PRINT #KB%,CHR$(155%)+'<'+CHR$(155%)+'[?3h'+FNSCROLL$(SCROLL.START%,24%);
        \ PRINT #KB%,FNATTR$('0')+FNV$(1%,1%,N$,EOS%,V%);
        \ GOSUB 16000%
        \ GOSUB 10010%


                        !>>SET UP FOR SYS....


2000      PRINT #KB%,RECORD 4096%,CHR$(19%);
        \ GOSUB 5000%
        \ GOSUB 6000%
        \ GOSUB 7000%
        \ GOSUB 9000%
        \ GOSUB 14120%
        \ PRINT #KB%,FNV$(1%,24%,N$,EOS%,V%);
        \ PRINT #KB%,RECORD 4096%,CHR$(17%);
                        !>>DI SY/S-
                        !  DO SY/D-
                        !  DO SY/M


2010      WAIT 60%
        \ E%=FNI%(N$)
```

```
        \       PRINT #KB%,USING '###',OPEN.FILES%;
        \       PRINT #KB%,FNATTR$('0');


                                !>>PRINT DISK STATUS ...


6010      NEXT DISK%
        \ JOB.LOG%=PEEK(JOBCNT%)
        \ JOB.NOW%=JOB.LOG% AND 255%
        \ JOB.LOG%=SWAP%(JOB.LOG%) AND 255%
        \ PRINT #KB%,FNV$(1%,COL.MAX%+DISK%+1%,N$,0%,V%);
        \ PRINT #KB%,FNATTR$('1;5');  IF JOB.NOW%>=JOB.LOG%
        \ PRINT #KB%,USING '##/##',JOB.NOW%,JOB.LOG%;
        \ PRINT #KB%,FNATTR$('0');
        \ FREE.BUF%=PEEK(FREES%+2%)
        \ PRINT #KB%,FNATTR$('1;5'); IF FREE.BUF%<100%
        \ PRINT #KB%;FNV$(3%,COL.MAX%+DISK%+2%,N$,0%,V%);
        \ PRINT #KB%,USING '###',FREE.BUF%;
        \ PRINT #KB%,FNATTR$('0');
        \ RETURN


                        !>>AFTER DISK STATUS PRINT NUMBER OF JOBS AND
                        !  SMALL BUFFER COUNT


7000      S.LINK%=PEEK(SNDLST%)
        \ RC%=0%

                        !>>START TRACE OF SEND LIST


7010      RC%=RC%+1%
        \ RCVR%=S.LINK%+2%
        \ RCVR.ID$=CVT%$(SWAP%(PEEK(RCVR%)))+CVT%$(SWAP%(PEEK(RCVR%+2%)))+
                  CVT%$(SWAP%(PEEK(RCVR%+4%)))
        \ RCVR.JOB%=(PEEK(RCVR%+6%) AND 255%)/2%
        \ LSET RC$(RC%,0%)=RCVR.ID$+''
        \ JOB.NAME$=SPACE$(6%)
        \ JOB%=RCVR.JOB%
        \ LSET RC$(RC%,1%)=NUM1$(JOB%)
        \ GOSUB 13000% IF JOB%>0% AND JOB%<63%
        \ LSET RC$(RC%,2%)=JOB.NAME$
        \ GOSUB 8000%
        \ RC%(RC%)=-1%
        \ S.LINK%=PEEK(S.LINK%)
        \ GOTO 7010 IF NOT(S.LINK%=0% OR S.LINK%=-1%)
        \ LSET RC$(RK%,0%),RC$(RK%,1%),RC$(RK%,2%),RC$(RK%,3%)=''
                                  FOR RK%=RC%+1% UNTIL RK%>=10%


                        !>>GET AND SAVE ALL RECEIVERS


7020      FOR K%=1% TO 10%
        \ GOTO 7030 IF RC%(K%)=0%
        \ PRINT #KB%,FNREC$(K%);
        \ PRINT #KB%,RC$(K%,0%);' ';RC$(K%,1%);' ';RC$(K%,2%);' ';RC$(K%,3%);
        \ RC%(K%)=0% IF RC$(K%,0%)=''

                        !>>PRINT ..


7030      NEXT K%
        \ RETURN

                        !>>THATS ALL FOLKS


8000      LSET RC$(RC%,3%)=''
        \ RETURN IF NOT( JOB.NAME$='BATDEC' OR
                         JOB.NAME$='BATRUN' OR
                         JOB.NAME$='RJESPL' OR
                         JOB.NAME$='SPLRUN'
        \ CHANNEL%=3%
        \ CHANNEL%=2% IF JOB.NAME$='BATRUN'
        \ CHANNEL%=7% IF JOB.NAME$='RJESPL'
        \ GOSUB 11000%
        \ LSET RC$(RC%,3%)=FILOPN$
        \ RETURN


                        !>>GET INPORTANT FILE THAT IS OPENED


9000      G$=DATE$(0%)+' '+TIME$(0%)
        \ PRINT #KB%,FNV$(30%,COL.MAX%+5%+1%,N$,EOL%,V%);G$;
        \ RETURN


10000     D$=SYS(CHR$(6%)+CHR$(16%)+CHR$(0%)+CHR$(PK0%+PK%)+CHR$(0%)+
                CHR$(0%)+CHR$(0%)+CHR$(0%)+CHR$(0%)+CHR$(255%))
        \ RETURN

                        !>>SET UP PK ... AS WIDTH 54 AND HALF DUPLEX

10010     D$=SYS(CHR$(6%)+CHR$(16%)+CHR$(0%)+CHR$(255%)+CHR$(0%)+
                CHR$(0%)+CHR$(0%)+CHR$(0%)+CHR$(0%)+CHR$(0%)+
                CHR$(0%)+CHR$(0%)+CHR$(0%)+CHR$(0%)+CHR$(0%)+
                CHR$(0%)+CHR$(0%)+CHR$(0%)+CHR$(0%)+CHR$(0%)+
                CHR$(0%)+CHR$(255%)+CHR$(0%)+CHR$(0%)+CHR$(0%)+
                CHR$(0%))
      . \ RETURN

                        !>>SET UP KB AS ESC SEQ ON ENTRY

10020     D$=SYS(CHR$(6%)+CHR$(16%)+CHR$(0%)+CHR$(255%)+CHR$(0%)+
                CHR$(0%)+CHR$(0%)+CHR$(0%)+CHR$(0%)+CHR$(0%)+
                CHR$(0%)+CHR$(0%)+CHR$(0%)+CHR$(0%)+CHR$(0%)+
               CHR$(0%)+CHR$(0%)+CHR$(0%)+CHR$(0%)+CHR$(0%)+
               CHR$(0%)+CHR$(128%)+CHR$(0%)+CHR$(0%)+CHR$(0%)+
               CHR$(0%))
          \ RETURN

                         !>>SET UP KB AS NO ESC SEQ ON EXIT

11000       !------------------------------------------------------------

                         SUBROUTINE TO FIND INFO ABOUT A DISK FILE
                         OPENED BY A JOB.


               GLOBALS:

                                   JOB%     -       JOB YOU ARE LOOKING AT

                                   JOBTBL% -        JOB TABLE LOCATION

                                   CHANNEL%         CHANNEL YOU WANT INFO ON

                                   FCBLST% -        LIST OF FILE CONTROL BLOCKS


               RETURNS:


                                   FILOPN$ -        FILE INFORMATION

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


11010      FILOPN$=''
          \ JDB%=PEEK(JOBTBL%+JOB%+JOB%)
          \ IOB%=PEEK(JDB%)
          \ DDB%=PEEK(IOB%+CHANNEL%+CHANNEL%)
          \ RETURN UNLESS DDB%
          \ DRIVER.INDEX%=PEEK(DDB%) AND 255%
          \ RETURN IF DRIVER.INDEX%<>0% OR FCBLST%=0%


                         !>>LOOK IN IOB FOR CHANNEL SPECIFIED IF NOT A
                         !  DISK FILE FORGET LOOKING. THIS ROUTINE ONLY
                         !  WORKS ON LAGE FILE SYSTEMS.


11020       DIM FCB%(16%),WCB%(16%)
          \  WCB%=DDB%
          \  WCB%(K%)=PEEK(WCB%+K%+K%) FOR K%=0% TO 15%
          \  FCB%=WCB%(4%)-28%
          \  FCB%(K%)=PEEK(FCB%+K%+K%) FOR K%=0% TO 15%
          \  OFF%=FCB%(12%) AND 255%

          !      GET FCB , WCB  FOR CHANNEL


11030      F.CLUS%=FCB%(14%)
          \ GOTO 11040 IF F.CLUS%=2%^I% FOR I%=0% TO 8%
          \ RETURN

          !      LOOK FOR MEANINGFUL DATA TO PRINT

11040      F.NAME$=RAD$(FCB%(3%))+RAD$(FCB%(4%))+'.'+RAD$(FCB%(5%))
          \ SI=(SWAP%(FCB%(12%)) AND 255%)*65536.+32768.+(FCB%(13%) EQV 32767%)


11050      FILOPN$=DISK.NAME$(OFF%+1%)+'['+NUM1$(SWAP%(FCB%(2%)) AND 255%)+','+
                   NUM1$(FCB%(2%) AND 255%)+']'+F.NAME$
          \ FILOPN$=FILOPN$+'<'+NUM1$((SWAP%(FCB%(6%)) AND 255%))+'>'
          \ FILOPN$=FILOPN$+' '+NUM1$(SI)+'/'


          !      PRINT FILE NAME AND OTHER SUCH INFO....


11060      REC=(WCB%(3%) AND 255%) + 256.*(SWAP%(WCB%(3%)) AND 255%)
          \ FILOPN$=FILOPN$+NUM1$(REC)


          !      PRINT RECORD IN FILE

11070      RETURN

12000                !---------------------------------------------------------------!


                         SUBROUTINE TO DETERMINE JOB STATUS FROM THE MONITOR
                         FLAGS AND OTHER DATA.


                         GLOBALS:


                                  JOB%     -       JOB NUMBER.

                                  JBWAIT% -        ADDRESS OF THE JOB WAIT TABLE.

                                  JBSTAT% -        ADDRESS OF THE JOB STATUS TABLE.

                                  LOGNAM% -        ADDRESS OF LOGICAL NAME TABLE.

                                  DEVNAM% -        ADDRESS OF DEVICE NAME TABLE.

                                  DEVOKB% -        NUMBER OF DISK DEVICES IN DEVNAM.
```

```
        !      PRINT SLOT # AND SWAPPING FILE IF SWAPPED, WHERE
        !      A               SWAP0.SYS
        !      B               SWAP1.SYS
        !      C               SWAP2.SYS
        !      D               SWAP3.SYS


        !      OR PRINT SOME OTHER APPROPRIATE MESSAGE


12070          RETURN
        !      SURE SEEM LIKE A WHOLE LOT OF CODE TO FIND OUT
               WHAT IS GOING ON . BUT WHO AM I TO QUESTION.



13000   !      ------------------------------------------------------------------      !



               THIS ROUTINE GETS A JOBS PPN , KEYBOARD NUMBER , JOB NAME ,
               CONNECT TIME , CPU TIME , JOB SIZE , AND DETERMINES IF THE
               JOB IS DETACHED OR NOT.



               GLOBALS:

                      JOBTBL% -       ADDRESS OF JOB TABLE.


               RETURNS:

                      JOB.PPN%        JOBS PROJECT PROGRAMMER NUMBER.

                      JOB.TTY%        JOBS KEYBOARD.

                      JOB.NAME$       JOB NAME.

                      CONTME  CONNECT TIME IN MINUTES.

                      CPUTME          CPU TIME IN SEC.

                      JOB.SIZE%       JOBSIZE IN K WORDS.

                      JOB.DET$        DET IF JOB DETACHED ATT IF NOT.

                      JOB.RTS$        JOBS RUN TIME SYSTEM.

                      TTY.TYPE$       TYPE OF KEYBOARD.



13010          JDB%=PEEK(JOBTBL%+JOB%+JOB%)
        \      JDB2%=PEEK(JDB%+8%)
        \      JOB.PPN%=PEEK(JDB2%+24%)
        \      JOB.NAME$=RAD$(PEEK(JDB2%+12%))+RAD$(PEEK(JDB2%+14%))
        \      JOB.SIZE%=PEEK(JDB%+22%) AND 255%
        \      JDRTS%=PEEK(JDB%+12%)
        \      JOB.RTS$=RAD$(PEEK(JDRTS%+2%))+RAD$(PEEK(JDRTS%+4%))
        \      DDB%=PEEK(PEEK(JDB%))
        \      INTERFACE.TYPE%=PEEK(DDB%+30%) AND 255%
        \      IF (PEEK(DDB%+2%) AND 255%)=JOB%+JOB% AND
               (PEEK(DDB%+6%) AND 8192%)<>0% THEN JOB.DET$='ATT'
               ELSE    JOB.DET$='DET'
        !      GET AS MUCH AS YOU CAN . IF THE OWNER JOB INDEX FROM THE DDB IS
        !      THE SAME AS THE JOB NUMBER AND THIS IS THE JOB'S CONSOLE TERMINAL
        !      THEN JOB IS ATTACHED ELSE IT IS DETACHED.


13020          JOB.TTY%=SWAP%(PEEK(DDB%+2%)) AND 255%
        \      TTY.TYPE$='KB'
        \      IF INTERFACE.TYPE%=8% THEN TTY.TYPE$='PK'
        \      JOB.TTY%=JOB.TTY%-PK0%
        \      KB.OWNER%=(PEEK(PEEK(PKDEV%+JOB.TTY%+JOB.TTY%)+2%) AND 255%)/2%


        !      GET KEYBORAD NUMBER AND SET UP TYPE AND NUMBER , DO SPECIAL
        !      ADJUST FOR PSEUDO KEYBOARD NUMBERS.


13030          CPUTME=PEEK(JDB2%+2%)
        \      CPUTME=CPUTME+65536. IF CPUTME<0.
        \      CPUTME=CPUTME+65536.*(SWAP%(PEEK(JDB2%+10%)) AND 255%)
        \      CONTME=PEEK(JDB2%+4%)
        \      CONTME=CONTME+65536. IF CONTME<0.
        \      RETURN



        !      EXTRACT CPU TIME AND CONNECT TIME AND CONVERT TO REAL
        !      NUMBERS.



14000!

        **** PSEUDO-KEYBOARD RUN


14010   OPEN 'NL:' AS FILE 2% \ PK%=0%

14020   K$=NUM1$(PK%)
        \      K$="PK"+K$+":"
```

```
                RETURNS:

                        JOSTATUS$ - JOB STATUS (SYSTAT COMPATABLE).


12010           JOB.JBWAIT%=PEEK(JBWAIT%+JOB%+JOB%)
        \       JOB.JBSTAT%=PEEK(JBSTAT%+JOB%+JOB%)
        \       JOSTATUS$='RN'
        \       IF (JOB.JBSTAT% AND JOB.JBWAIT%)<> 0% THEN 12050 ELSE
                JOSTATUS$='RS'
        \       JDB%=PEEK(JOBTBL%+JOB%+JOB%)
        \       IF PEEK(JDB%+24%)=0% AND
                   (PEEK(JDB%+22%) AND -256%)=(NOT 15359%) THEN 12050
                ELSE JOB.JBWAIT%=JOB.JBWAIT% AND (NOT 16384%) IF (JOB.JBWAIT% AND (NOT 16384%))<>0%
        \       JOSTATUS$='BF'
        \       IF (JOB.JBWAIT% AND 16384%)<>0% THEN 12050
                        ELSE JOSTATUS$='SL'
        \       IF (JOB.JBWAIT% AND 8192%)=0% THEN 12020
                ELSE JOSTATUS$='SR' IF (PEEK(JDB%+4%) AND 255%)=5%
        \       GOTO 12050

        ! POSSIBLE STATES                       CONDITIONS
        !       RN                              (JBWAIT AND JBSTAT)<>0
        !       RS (RESIDENCY WAIT)             IF M.PHYA=0 AND M.CTRL=LCK,SWP AND IN

                TURN OFF BUFFER WAIT BIT IF ANY OTHERS ARE ON. CHECK FOR THE FOLLOWING

        !       BF (BUFFER WAIT)                IF BIT 14 IS ON.
        !       SL (SLEEP WAIT)                 IF BIT 13 IS ON.
        !       SR (RECEIVER WAIT)              IF BIT 13 IS ON AND ERROR CODE IS 5


12020           JOSTATUS$='FP'
        \       IF (JOB.JBWAIT% AND 4096%)<>0% THEN 12050
                ELSE JOSTATUS$='TT'
        \       IF (JOB.JBWAIT% AND 2048%) THEN 12050
                ELSE JOSTATUS$='HB'
        \       IF JOB.JBWAIT%=0% THEN 12050
                ELSE H2%=2%
        \       GOTO 12040 IF T%=2%

        !       FP                              IF BIT 12 IS ON
        !       TT                              IF BIT 11 IS ON.
        !       HB                              NO BITS ARE ON.

        !       IF NONE OF THE PREVIOUS STATES THEN WE HAVE A 'DEVICE WAIT'.
        !       BYPASS THE CHANNEL # CHECK IF WE KNOW WE HAVE A KB INPUT WAIT.


12030           CHANNEL%=PEEK(PEEK(JDB%+6%)+6%) AND 255%
        \       H2%=PEEK(PEEK(PEEK(JDB%)+CHANNEL%)) AND 255%
        \       IF H2%=0% THEN
                        UNIT%=SWAP%(PEEK(PEEK(PEEK(JDB%)+CHANNEL%)+2%)) AND 255%
        \               UNIT%=PEEK(PEEK(PEEK(PEEK(JDB%)+CHANNEL%)+8%)-4%) AND 255% IF FCBLST%
        \       JOSTATUS$=CVT%$(SWAP%(PEEK(LOGNAM%+(UNIT%*10%)+4%)))
        \       GOTO 12050

        ! INDEX INTO THE IOB AND LOOK AT THE HANDLER INDEX (H2%).
        ! IF WE HAVE A DISK (H2%=0%) THEN
        !       GET THE INTERNAL DISK UNIT NUMBER AND EXTRACT THE
        !       PROPER NAME FROM LOGNAME.


12040           JOSTATUS$=CVT%$(SWAP%(PEEK(DEVNAM%+DEVOKB%+H2%-2%)))
        \       JOSTATUS$='^C' IF PEEK(PEEK(JDB%+6%)+10%)<0% IF H2%=2%

        !       WE HAVE A NON DISK WAIT. INDEX INTO THE DEVNAM TABLE TO GET THE
        !       PROPER NAME . CHECK FOR ^C ON KB INPUT WAITS.


12050           JDSWAP%=SWAP%(PEEK(JDB%+30%)) AND 255%
        \       M.CTRL%=PEEK(JDB%+22%)
        \       IF M.CTRL%<0% THEN
                        JDSWAP%=64%
        \       IF (M.CTRL% AND 1536%)<>0% THEN
                JDSWAP%=JDSWAP%+64%
        \       IF (M.CTRL% AND 512%)<>0% THEN
                        JDSWAP%=JDSWAP%+64%

        ! GET CONTENTS OF JDSWAP AND M.CTRL.

        !       IF M.CTRL<0 THEN
        !               LOCKED                          IF BIT 15 IS ON.
        !               SWAPPING IN (OR WANTS TO)               IF BIT 10 IS ON.
        !               SWAPPING OUT(OR WANTS TO)               IF BIT 9 IS ON


12060           IF (JDSWAP% AND 63%)<>0% THEN
                JOSTATUS$=JOSTATUS$+' '+CHR$(65%+(JDSWAP% AND 192%)/64%)+
                        RIGHT(NUM$(100%+(JDSWAP% AND 63%)),3%)
                ELSE    IF (JDSWAP% AND 192%)=0% AND (PEEK(JDB%+2%) AND 16384%)<>0% THEN
                        JOSTATUS$=JOSTATUS$+' '+'NSW '
                ELSE    JOSTATUS$=JOSTATUS$+' '+
                        MID('    LCK SWI SWO ',(JDSWAP% AND 192%)/16%+1%,4%)
```

```
        \       OPEN K$ AS FILE 3%
        \       FIELD #3%, BUFSIZ(3%) AS P$
        \       RETURN
14030!

        ***** SUBROUTINE TO SEND T$ TO PK WHEN PK IS IN KB WAIT *****


14040     PUT #3%,RECORD 4%,COUNT 1%
        \ GOTO 14060

                                    !>>KEY BOARD WAIT

14050     GOSUB 14130
        \ SLEEP 1%
        \ GOTO 14040

                                    !>>GET OUTPUT THEN GO BACK TO CHECK
                                    !  FOR KEY BOARD WAIT


14060     GOSUB 14130%
        \ LF%=-1%  IF ASCII(T$)>=32% OR LEN(T$)=0%
        \ T$=T$+CHR$(13%) IF LF%
        \ LSET P$=T$

                              !>>GET ANY WAITING OUTPUT PUT ON <CR>
                              !  SET INTO BUFFER


14070     PUT #3%,RECORD 9%,COUNT LEN(T$)
        \ IF ECHO% THEN PRINT #2%,T$;

                                    !>>ECHO ENABLED


14080     PUT #3%,RECORD 4%,COUNT 1%
        \ GOTO 14100%

                                    !>>KEY BOARD WAIT ???


14090     GOSUB 14130%
        \ SLEEP 1%
        \ GOTO 14080%

                                    !>>GRAB OUTPUT AND GO BACK TO
                                    !  TRY FOR KEY BOARD WAIT


14100     GOSUB 14130%
        \ RETURN

                                    !>>LOOK FOR MORE OUTPUT AND THE
                                    !  GO BACK FOR MORE.


14110   RETURN                                !ALL DONE
14120!

        **** GET OUTPUT FROM PK ****


14130     GET #3%
        \ M$=LEFT(P$,RECOUNT)
        \ M1$=LEFT(M$,2%)
       .\ M$=RIGHT(M$,5%) IF M1$='^C' OR M1$='^Z'
        \ M$=RIGHT(M$,2%) IF LF%
        \ PRINT #2%,M$; IF ECHO%
        \ M$=RIGHT(M$,2%) IF LF%
        \ LF%=0%
        \ PRINT #KB%,M$;
        \ GOTO 14130


14140     RETURN

                              !>>NO OUTPUT


14200     LSET P$=CHR$(3%)
        \ PUT #3%,RECORD 9%,COUNT 1%
        \ GOSUB 14130%
        \ IF LINE=14130% THEN RESUME 14130% ELSE
         IF LINE<14070% THEN RESUME 14040% ELSE
         IF LINE>14070% THEN RESUME 14080% ELSE
          STOP

                        !>>FORCE ^C THEN RETURN TO WHAT YOU WHERE DOING


15000     RETURN IF ASCII(CMD$)<>128%
        \ E%=-1%
        \ CMD$=LEFT(CVT$$(CMD$,37%),2%)
        \ RETURN IF ASCII(CMD$)<>79%
        \ E%=0%
        \ CMD$=RIGHT(CMD$,2%)
        \ PF%=ASCII(CMD$)-79%
        \ DF$=FTBL$(PF%,0%)

                  !>>CHECK FOR ESC COMMAND THEN CHECK FOR 'O' PREAMBLE


15010    E%=FNI%(N$)
        \ RETURN IF E%
        \ GOTO 15010 IF ASCII(I0$)<>128%
        \ I0$=RIGHT(I0$,3%)
        \ FC%=INSTR(1%,'qrstuvwxypnlm',IO$)
        \ PRINT #KB%,FNV$(1%,24%,N$,EOS%,V%);FTBL$(PF%,FC%);
        \ CMD$=FTBL$(PF%,FC%)
        \ IF FTBL%(PF%,FC%) THEN PRINT #KB%
        \                                   RETURN

                        !>>GET FUNCTION WITHIN PROGRAM FUNCTION


15020     E%=FNI%(FNV$(1%,24%,CMD$,EOS%,V%))
        \ CMD$=CMD$+IO$
        \ RETURN

                        !>>GET LAST OF DATA FOR COMMAND


16000     READ F%,P%,C$,FLG%
        \ RETURN IF C$='<<END>>'
        \ FTBL$(F%,P%)=C$
        \ FTBL%(F%,P%)=FLG%
        \ GOTO 16000%

                        !>>READ IN COMMAND TABLE


20000     DEF FNJOB$(JOB%)
        \ Y%=JOB%-(COL.MAX%*((JOB%-1%)/COL.MAX%))
        \ JOB.COL%=(JOB%-1%)/COL.MAX%
        \ X%=SYS.OFF%*JOB.COL%
        \ X%=X%+SYS.START%
        \ FNJOB$=FNV$(X%,Y%,N$,0%,V%)
        \ FNEND


20010     DEF FNATTR$(Z$)
        \ FNATTR$=CHR$(155%)+'['+Z$+'m'
        \ FNEND


20020     DEF FNDISK$(DISK%)
        \       Y%=DISK%+COL.MAX%
        \       X%=DISK.START%
        \ FNDISK$=FNV$(X%,Y%,N$,0%,V%)
        \ FNEND


                        !>>DISK OFFSET ...


20030     DEF FNREC$(RC%)
        \ Y%=RC%-(5%*((RC%-1%)/5%))
        \ X%=(RC%-1%)/5%
        \ X%=X%*REC.OFF%+REC.START%
        \ Y%=Y%+COL.MAX%
        \ FNREC$=FNV$(X%,Y%,N$,0%,V%)
        \ FNEND


20040     DEF FNSCROLL$(T%,B%)
        \ FNSCROLL$=CHR$(155%)+'['+NUM1$(T%)+';'+NUM1$(B%)+'r'
        \ FNEND


24000   !>>STANDARD FUNCTIONS
24005 DEF FNI%(Z$)=FNI1%(Z$,0%)          !>>KEYBOARD INPUT FUNCTION
                                         !  (NO ECHO CONTROL)
24010 DEF FNI1%(Z$,Z1%)
        \FNI1%,E%=0%
        \E1%=-1%
        \IO$=N$
        \ Z2%=Z1%
        \ Z2%=2% IF Z2%=1%
                                         !>>ECHO-CONTROL INPUT FUNCTION
24020   PRINT #KB%, Z$;
        \ PRINT #KB%, RECORD 256%, CHR$(Z2%)+CHR$(V%); IF Z2%
                                         !  PRINT PROMPT STRING
                                         !  SET MODE,MAX LENGTH,PAINT CHAR
24030     GET #KB%
        \ E1%=0%
        \ FIELD #KB%, RECOUNT AS Z1$
        \ IO$=CVT$$(Z1$,37%)+''
        \ IO$=LEFT(Z1$,3%)    IF ASCII(Z1$)=128%
                                         !>>GET RESPONCE
                                         !  AND SET IO$


24040   E%,FNI1%=E% OR (IO$="/E")
        \ IO$=LEFT(IO$,1%) IF Z1%=1%
        \ PRINT #KB% UNLESS X3%
        \ IF     X3%
        THEN    Z%=FNV4%(N$,0%) IF X3% AND 512%
        \       Z%=FNV4%(N$,1%) IF X3% AND 1024%
        \       X3%=X3% AND 255%         !  SET ERROR FLAG, FN VALUE
                                         !  CLEAR BOTTOM OF SCREEN
24050   FNEND
24060 DEF FNV%(Z$)
        \E%=0%
        \FNV%=VAL(Z$)                    !>>FNV%=CONVERT STRING TO INTEGER
24070   FNEND
24080 DEF FNV(Z$)
        \E%=0%
        \FNV=VAL(Z$)                     !>>FNV=CONVERT STRING TO REAL
24090   FNEND
24800                                    !>>SCREEN
                                         !  CONTROL
                                         !  FUNCTIONS
24810 DEF FNV$(X%,Y%,Z$,Z1%,V%)
        \ IF     X%*Y%*X3%=0%
```

```
        THEN    FNV$=Z$
        ELSE    Z9$=CHR$(155%)+"["+NUM1$(Y%)+';'
        \       Z8$=Z9$+NUM1$(X%)+'H'
        \       Z8$=Z8$+CHR$(155%)+"[J" IF Z1%=-1%
        \       Z8$=Z8$+CHR$(155%)+"[K" IF Z1%=-2%
        \       Z8$=Z8$+Z$
        \       Z8$=Z8$+STRING$(Z1%,V%-32%*(V%=0%))+Z9$+NUM1$(X%+LEN(Z$))+'H' IF Z1%>0%
        \       FNV$=Z8$
24840   FNEND                            !>>FNV$=CONSTRUCT CURSOR POSITIONED STRING
                                         !  X%,Y%=SCREEN COORDINATES
                                         !  Z$=BASE STRING
                                         !  Z1%=LENGTH TO "CLEAR"
                                         !  V%="CLEAR" CHARACTER
24850 DEF FNV4%(Z$,Z1%)
        \Z%=80%-LEN(Z$)
        \Z%=1% IF Z%<1%
        \Z$=CHR$(7%)+Z$ IF LEN(Z$)
        \X3%=X3% OR (512%*2%**Z1%) IF X3%
        \Z1%=23%+Z1%
        \PRINT #KB%, FNV$(1%,Z1%,N$,EOL%,32%); FNV$(Z%/2%,Z1%,Z$,0%,32%);
        \PRINT #KB% UNLESS X3%
        \FNV4%=0%
24860   FNEND                            !>>FNV4%=CENTER Z$,PRINT AT BOTTOM OF SCREEN
                                         !  Z1%=0 FOR LINE 23
                                         !      1 FOR LINE 24
30000   !>> CCL ENTRY
31000     DATA 1,1,'QUE LP0:/FORM:OPR=',0,1,2,'QUE LP1:/FORM:OPR=',0
31001     DATA 1,3,'QUE LP0:=',0,1,4,'QUE LP1:=',0
31002     DATA 1,5,'QUE/L LP0:',-1,1,6,'QUE/L LP1:',-1
31003     DATA 1,7,'QUE/L LP:',-1,1,8,'QUE/S LP0:',-1
31004     DATA 1,9,'QUE/S LP1:',-1,1,10,'QUE/S LP:',-1
31020     DATA 2,1,'PLE /INT LP0SPL:FORM OPR/WIDTH:75',-1,2,2,'PLE /INT LP0SPL:FORM OPR',-1
31021     DATA 2,3,'PLE /INT LP1SPL:FORM OPR/WIDTH:75',-1,2,4,'PLE /INT LP1SPL:FORM OPR',-1
31022     DATA 2,5,'PLE /INT LP0SPL:FORM NORMAL/WIDTH:132',-1,2,6,'PLE /INT LP1SPL:FORM NORMAL/WIDTH:132',-1
31023     DATA 2,7,'PLE /INT LP0SPL:END',-1,2,8,'PLE /INT LP1SPL:END',-1
31024     DATA 2,9,'PLE /INT LP0SPL:RESTART',-1,2,10,'PLE /INT LP1SPL:RESTART',-1
31025     DATA 2,11,'PLE /INT LP0SPL:ABORT',-1,2,12,'PLE /INT LP1SPL:ABORT',-1
31040     DATA 3,1,'DIR RPT:/S',-1,3,2,'DIR DB0:(1,203)/S',-1
31041     DATA 3,3,'DIR GARDEN:(1,6)/SU',-1,3,4,'DIR GARDEN:(1,7)/SU',-1
31900     DATA 1,0,'Queue ',0,2,0,'Please ',0
31901     DATA 3,0,'Direct ',0
31999     DATA 0,0,'<<END>>',0
32000   !>> ERROR TRAPS
32010   IF ERR=28% THEN SLEEP 1%
        \ Z1$=SYS(CHR$(6%)+CHR$(-7%))
        \ IO$,CMD$=CHR$(3%)
        \ IF E1% THEN RESUME 24040% ELSE
          RESUME 14200%


32020   IF ERR<>33% THEN 32040 ELSE
         UNIBUS%=1%
        \ RESUME 32030

                !       UNIBUS TIMEOUT FATAL ERROR


32030     RETURN

                !       RETURN FOR MORE FUN

32040     IF ERR=15% AND ERL=24030% THEN
                        FNI1%,E%=-8192%
        \       IO$=N$
        \       RESUME 24050

                        !>>WAIT EXAUST ...


32050   IF ERR=5% THEN
                IF ERL=14040% THEN RESUME 14060% ELSE
                IF ERL=14080% THEN RESUME 14100         !NO JOB LOGGED IN
32060   IF ERR=3% THEN
                IF ERL=14040 THEN RESUME 14050 ELSE
                IF ERL=14080 THEN RESUME 14090
32070   IF ERR=11% AND ERL=14130% THEN RESUME 14140%

                                        !>>NO PK OUTPUT
32080   IF ERR=8% THEN PK%=PK%+1%
        \ RESUME 14020% IF PK%<8%
        \ PRINT 'All psuedo keyboards are in use.'
        \ STOP

                        !>>KEEP TRYING ...


32090     IF ERR=3% AND ERL=24020% THEN RESUME 24030%

32600   !>> STANDARD ERROR TRAPS
32610   IF ERL=24030 AND ERR=11%
         THEN IO$=CHR$(26%)
        \ RESUME 24040


32650   IF ERL = 24060 THEN FNV% = 0%
        \E% = -1%
        \RESUME 24070                   !  FNV% ERROR HANDLER
32660   IF ERL = 24080 THEN FNV = 0
        \E% = -1%
        \RESUME 24090                   !  FNV ERROR HANDLER
32700   PRINT L$;"?UNFORESEEN ERROR DETECTED IN "; P9$
        \ ON ERROR GO TO 0
                                        !  ERRORS END HERE
32760   CLOSE Z% FOR Z% = 1% TO 12%
                                        !  EXIT THRU HERE
                                        !  CLOSE EVERYTHING

32766   CHAIN U1$ U1% IF U1$ <> N$      !>>CHAIN AWAY
32767   END                             !>>T2APP
```

AARDVARK SOFTWARE introduces the complete UCSD PASCAL system available under RSTS/E for the PDP-11 family.

Not merely a compiler!

This system was designed to facilitate software development and create an ideal Pascal teaching environment. Designed and developed by John Hayward, Ph.D. of Wheaton College, Wheaton, Illinois.

ADVANTAGES

***Based on enhanced UCSD PASCAL, utilizing:**
- **long integers**
- **strings**
- **segmented procedures**

***Simplified user interface—automatic linking**
***Powerful screen editor**
***Compatible with UCSD microcomputer systems**
***Supports "random access" of files**
***Includes Assembler**

Price: $995

Write us, or call toll free for information: 1-800-558-1538

AARDVARK SOFTWARE INC.
783 North Water Street
Milwaukee, Wisconsin 53202

UCSD PASCAL is a trademark of the Regents of the University of California.
PDP-11 and RSTS/E are trademarks of the Digital Equipment Corporation.

CIRCLE 41 ON READER CARD

NOTE: The information contained in this document should not be construed as a commitment by the author for the accuracy or reliability of this information.

Installation of this software is not supported and will result in the operation of a non-standard system. Use this software at your own risk.

The Implementation of Hidden Files Under RSTS/E Version 7.0

By John F. Sandhoff, Data Processing, California State University at Long Beach

An Introduction to Hidden Files

There exists in certain RSTS/E applications a desire to make certain files 'invisible' to the general user public. DEC has partly recognized this need and incorporated, under version 7.0, a means of making it impossible for non-privileged users to get directory information of cross-account, private files. This article describes a method developed by and in use at Long Beach State University that permits specific public files to be transparent to non-privileged users.

Why a need for hidden files? In many environments, especially educational installations, it is highly desirable for certain programs to be able to access a data file without using privileges, but still be able to maintain a high degree of security. Defining a file as public allows a program to access it, but unfortunately it also allows anyone that knows the filename to access it. If the name could be kept secret, and a discreet name chosen to prevent a lucky guess, all requirements would be met.

The $DIRECT program can be disabled, but such an action is not very practical. Besides, users can still write their own directory program by using the appropriate SYS calls. If the directory SYS calls are also disabled (or privileged), many programs that rely on performing a directory lookup, such as $PIP.SAV and the task builder, would fail to operate. Even though it can be argued that disabling $TKB provides definite system throughput gains, such a move is generally impractical.

The process of hiding files as described in this document is a relatively simple change to the FIP directory lookup code. It is an extension of feature patch 3.5.9, which restricts directory lookups. What happens is this:

When a directory lookup request is made to FIP (SYS calls 6+15 or 6+17), a section of code is executed that scans a user's UFD. Normally, if the specified file exists, the directory information is returned. With patch 3.5.9 installed, this information is returned only if the requested file is defined as public to the calling user. (Please refer to the System Release Notes for more information). The hidden files patch presented here carries this mechanism one step further — if a file is public, another status bit is checked, and if this bit is on, the file is considered by FIP to be hidden, resulting in no directory information being returned.

After the installation of the hidden files option, a hidden file can be opened by a program (OPEN statement, CALFIP), and loaded or run with the 'OLD' or 'RUN' command, as applicable. However, such a file will not appear on a directory listing (except, of course, to the owner or a privileged user), and can not be accessed by programs that perform a directory lookup before opening the file. (Note that this includes PIP.SAV, TKB.TSK, and some RSX-based programs. RNO is not affected.)

Particular attention should be given to the fact that this patch functions in conjunction with DEC feature patch 3.5.9. Any installation that is interested in enhanced file security should already have this feature patch installed. This patch can be applied to either sub-version -07 or -08 of RSTS V7.0.

Please understand that the modifications described in this document do not in any way cause the RSTS file structure or FIP to lose track of any files. The modifications affect ONLY the return of information as requested by the directory lookup SYS calls by non-privileged tasks. A privileged user or job maintains full functionality of the SYS calls. It should also be pointed out that files can never be hidden from their owner. They will ALWAYS appear on the owner's directory listing, and this patch by design will NOT hide the files from their owner (or any privileged user).

Installing Hidden Files on your system involves the following changes:

- A patch to the RSTS FIP routines to modify the directory lookup calls

- A patch to $DIRECT to look for and respect the 'hidden file' status bit. Since $DIRECT does not use SYS calls, a separate procedure is necessary to implement hidden files in this program.

- A one-word patch to the CLEAN option of INIT. Without this patch, all hidden files will be reset to non-hidden status every time your system crashes.

- A one-word patch to $ONLCLN, for the same reason as given above.

- DEC feature patch 3.5.9 MUST be installed to enable hidden files.

- DEC feature patch 10.3.1 should be installed to the DIRECT program. While this is not mandatory, it is highly recommended for security reasons. Note that if it is not installed, any user can see all of another user's files, EXCEPT for files that have been hidden.

The patch to the RSTS FIP routines requires 9 (decimal) words of patch space. Currently unused space available in 'PATCH' is being used for the new code. Be aware that published patches to the RSTS operating system may in time overlap the space we have chosen. If such a time occurs, three alternatives must be explored:

1) Do not install the DEC patch. Depending upon what it does, this is probably an unwise choice.

2) Remove the hidden files patch. Depending upon how useful you have found them to be, you may or may not wish to do this.

3) Find someplace else for the hidden files patch. If you are not against performing a SYSGEN, and can justify getting rid of only one small buffer (I know, I know. . .), it is easy to 'gen in the patch as its own little segment. If reader interest is high enough, this topic will be addressed in a revised release of this document.

How the Hidden Files Feature Works

The Hidden Files Feature, in the words of one RSTS Internals person, will immediately crash your system and leave the disk unusable. Don't believe it! There are a set of status bits located in the link fields in the UFD entry. Some of these bits are used, some are not. One bit that is currently 'reserved' has been employed here to implement hidden files.

Because an unused bit is being used, problems may, I repeat, MAY, be encountered with future releases of RSTS/E. However, there is an extremely simple way of almost instantly clearing all hidden files if the need should arise. By removing the patch applied to either $ONLCLN or the CLEAN option of INIT, and then cleaning the disk, this unused bit is cleared and the disk is back to a totally standard format. (Obviously, all hidden files are 'unhidden' by this process.)

The implementation of hidden files has been in use on this campus since the original installation of RSTS V7.0, with no problems arising from its use. It is equally effective on both our system and private disks. No limitations to the system have been imposed by this change.

By now, I am certain that the bit-twiddlers among you are curious as to exactly what has been done to implement hidden files. Figure 1 diagrams a typical UFD Name Entry Blockette:

```
!------------------------------!---------!
!   Link to Next Name Entry    !  Flags  ! +0    ULNK
!------------------------------!---------!
!      File Name (Part 1 in RAD50)       ! +2    UNAM
!----------------------------------------!
!      File Name (Part 2 in RAD50)       ! +4
!----------------------------------------!
!      File Name (Part 3 in RAD50)       ! +6
!------------------!---------------------!
! Protection Code  ! Status Byte         ! +10   USTAT
!------------------!---------------------!
!           File Access Count            ! +12   UACNT
!------------------------------!---------!
!   Link to Accounting Entry   !  Flags  ! +14   UAA
!------------------------------!---------!
! Link to 1st Retrieval Entry  !  Flags  ! +16   UAR
!------------------------------!---------!
```

DEC
RSTS/E USERS

From one of the pioneers in commercial data processing using RSTS. Off the shelf software ready for immediate delivery. Completely interactive. Extensively documented. Fully supported. Ideal for OEM's, service bureaus or end users. Cost effective solutions including:

- ACCOUNTS PAYABLE
- GENERAL LEDGER
- FINANCIAL REPORTING
- ACCOUNTS RECEIVABLE
- PAYROLL
- FIXED ASSETS

For complete details, contact us ai:

Plycom services, inc.
P.O. Box 160
Plymouth, IN 46563
(219) 935-5121

CIRCLE 42 ON READER CARD

Note that each link field contains 4 flag bits. Hidden files makes use of bit 3 in the accounting entry link. If this bit is set, the file is considered to be hidden.

Extreme care was exercised when choosing which flag bit to utilize. Some of the various uses for many of these bits are: entry in use, file contains a bad block, disk caching, and flags reserved for CLEAN. Because of possible future extensions to the RSTS disk mechanism, DEC may in some future release decide to use the same bit that we are now using. For this reason, the availability of hidden files on future releases of RSTS cannot be guaranteed. (Of course, if DEC chooses to support the hidden file structure . . .)

Supplied with this document is a listing of a not perfect but nonetheless rather effective program used to hide and unhide files. As written, it is capable of stand alone operation (RUN command), or chain entry (CCL entry). When installed with the proper CCL's, both the 'HIDE' and 'UNHIDE' commands become available. Indirect command files can also be utilized to hide or unhide a list of files. This program must be defined as privileged, since it does access the MFD and UFD's.

Since a file's hidden status is not preserved by the standard backup programs provided, a rather crude method is presented in Appendix B to create an indirect command file that can be used to rehide all hidden files on the system (it must be performed from a privileged account). Sites choosing to implement hidden files are encouraged to watch for the release of a very fast and efficient system cataloging program that can create a recovery file to restore hidden files.

The Actual Coding Involved

This section provides the reader with the actual coding used, both the original sequence from FIP, and the modifications necessary to put hidden files into effect.

```
;
;        THE ROUTINE LISTED BELOW IS CONTAINED WITHIN PSECT 'DIRO' OF
;        THE FIP CODE. IT IS RESPONSIBLE FOR RETURNING DIRECTORY
;        INFORMATION AS REQUESTED BY SYS CALLS 6+15 AND 6+17 WHEN
;        THE TARGET DEVICE IS DISK.
;
;        THE FIRST FEW LINES DEAL WITH RETRIEVING THE NEXT FILENAME AND
;        ARE NOT NECESSARY TO THE UNDERSTANDING OF WHAT WE ARE INTERESTED
;        IN. STARTING AT THE CODE LABELED 'XXXX', WE HAVE A VALID
;        FILENAME ENTRY BLOCKETTE IN CORE, POINTED TO BY REGISTER R5.
;        SEE FIGURE 1 ELSEWHERE IN THIS DOCUMENT FOR THE LAYOUT OF
;        THIS INFORMATION.
;
;        THIS ROUTINE RETURNS WITH THE CARRY BIT SET IF THE FILENAME
;        WAS NOT FOUND. NOTE THAT UFD'S AND FILES MARKED FOR DELETION
;        ARE IGNORED.
;
;        THIS CODE ASSUMES THAT FEATURE PATCH 3.5.9 HAS BEEN INSTALLED.
;        THIS PATCH CHANGES THE INSTRUCTION AT '..DIRP' FROM AN
;        UNCONDITIONAL BRANCH TO A CONDITIONAL BRANCH.
;


TOP:     CALL    1$                  ; GET NEXT UNIT ELSEWHERE
START:   CALL    @#RDENXT            ; GET NEXT FILENAME
         BEQ     TOP                 ; NO MORE FILES, GET NEXT UNIT

XXXX:    BITB    #US.UFD!US.DEL,10(R5); CHECK THE USTAT BYTE
         BNE     START               ; IF UFD OR DELETED, ACT AS IF NOT THERE
         TSTB    @#FIPRIV            ; IS THIS USER PRIVILEGED?
..DIRP:  BEQ     20$                 ; SKIP ROUTINES IF PRIV USER
         CMP     (R0),@#FIUSER       ; ARE WE REQUESTING DATA ON OURSELVES?
         BEQ     20$                 ; YES, SKIP ROUTINES

         MOVB    11(R5),R2           ; RETRIEVE FILE'S PROTECTION CODE
         CMPB    1(R0),@#FIUSER+1;   SAME PROJECT NUMBERS?
         BEQ     10$                 ; YES, SHIFT ACCORDINGLY
         ASH     #-2,R2              ; SHIFT PROTECTION CODE
10$:     ASH     #-3,R2              ; SHIFT AGAIN
20$:     RETURN

;
;        THE FOLLOWING CODE CONTAINS THE MODIFICATIONS ADDED BY THE
;        HIDDEN FILES PATCH. THE LOCATION LABELED HERE AS 'PATCH' IS
;        WITHIN THE RSTS PATCH SPACE.
;

TOP:     CALL    1$                  ; GET NEXT UNIT ELSEWHERE
START:   CALL    @#RDENXT            ; GET NEXT FILENAME
         BEQ     TOP                 ; NO MORE FILES, GET NEXT UNIT

XXXX:    BITB    #US.UFD!US.DEL,10(R5); CHECK THE USTAT BYTE
         BNE     START               ; IF UFD OR DELETED, ACT AS IF NOT THERE
         TSTB    @#FIPRIV            ; IS THIS USER PRIVILEGED?
..DIRP:  BEQ     20$                 ; SKIP ROUTINES IF PRIV USER
         CMP     (R0),@#FIUSER       ; ARE WE REQUESTING DATA ON OURSELVES?
         BEQ     20$                 ; YES, SKIP ROUTINES

         CALL    PATCH               ; *** THIS IS OUR JUMP TO THE PATCH
         CMPB    1(R0),@#FIUSER+1;   SAME PROJECT NUMBERS?
         BEQ     10$                 ; YES, SHIFT ACCORDINGLY
         ASH     #-2,R2              ; SHIFT PROTECTION CODE
10$:     ASH     #-3,R2              ; SHIFT AGAIN
20$:     RETURN


PATCH:                               ; THIS IS THE ROUTINE PLACED IN PATCH
                                     ; SPACE
         MOVB    11(R5),R2           ; RETRIEVE FILE'S PROTECTION CODE
         BIT     #10,14(45)          ; LOOK FOR 'HIDDEN BIT' IN UAA FIELD
         BEQ     5$                  ; NOT HIDDEN, PROCEDE, ELSE
         BIS     #24,R2              ; FUDGE THE PROTECTION CODE TO
                                     : ASSURE FAILURE
5$:      RETURN
```

Patching Procedure

This section lists step-by-step instructions to apply the hidden files patch to RSTS/E V7.0 systems. This patch to FIP assumes that there is enough space in 'PATCH' to add the necessary code. (This is discussed more completely elsewhere.)

For those of you brave enough to venture into the fascinating world of bit-twiddling, here goes:

NOTE: If ANY of the old values appear incorrect, ESPECIALLY if any of the locations in the patch space are non-zero, ABORT THE PATCH PROCEDURE by typing a Control/C. Do NOT install the patches if things "don't seem right".

1) Add the following patch to FIP. If the directory code on your system is non-resident, this patch can NOT be installed to your currently running monitor. You must use the INIT PATCH option. Note that if DEC patch 3.5.9 has already been installed, the portions marked by asterisks must be deleted!

```
     RUN $ONLPAT
     Command file name?                        (Type a <LF> if entering it by hand)
                                               (Type 'filnam=' to save the patch)
     File to patch? RSTS.SIL                   (Type a <LF> for current SIL)
     Module name? OVR                          (FIP if code is resident)
     Base address? ..DIRP
**   Offset address? 0
**    Base   Offset  Old     New?
**   ??????  000000  000415  ? 1415            (This is DEC patch 3.5.9)
**   ??????  000002  ??????  ? ^Z
     Offset address? 10
      Base   Offset  Old     New?
     ??????  000010  116502  ? 4737
     ??????  000012  000011  ? PATCH+356
     ??????  000014  126037  ? ^Z
     Offset address? ^Z
     Base address? ^Z
     Module name? RSTS
     Base address? PATCH
     Offset address? 356
      Base   Offset  Old     New?
     ??????  000356  000000  ? 116502
     ??????  000360  000000  ? 11
     ??????  000362  000000  ? 32765
     ??????  000364  000000  ? 10
     ??????  000366  000000  ? 14
     ??????  000370  000000  ? BEQ+2
     ??????  000372  000000  ? 52702
     ??????  000374  000000  ? 24
     ??????  000376  000000  ? 207
     ??????  000400  ??????  ? ^C                (Uparrow/C to exit)
```

2) Add the following patch to (0,1)INIT.SYS

```
File to patch? (0,1)INIT.SYS
Base address? CLEAN
Offset address? 3402
 Base   Offset  Old     New?
113002  003402  042765  ? <LF>               (This step is for verification)
113002  003404  000013  ? 3
113002  003406  000014  ? ^C                 (Uparrow/C to exit)
```

3) Add the following patch to $ONLCLN.SAV

```
File to patch? SY:[1,2]ONLCLN.SAV
Base address? 55702
Offset address? 0
 Base   Offset  Old     New?
055702  000000  042765  ? <LF>               (This step is for verification)
055702  000002  000013  ? 3
055702  000004  000014  ? ^C                 (Uparrow/C to exit)
```

4) Marvel that your system is still running. The binary patches are finished, all that is left is a modification to the $DIRECT program, and the typing in of a program that makes it easy to hide a file.

5) Change the following three lines in the $DIRECT source. The first line contains only an addition, as indicated.

```
2250      P%(I%)=U%(U%,I%) FOR I%=1% TO 3% &
          \ P%(5%)=SWAP%(U%(U%,4%)) AND 255% &
          \ ULNK%=U%(U%,0%) &
          \ UAA%=U%(U%,6%) &
**===>    \ RETURN IF UAA% AND 8% IF P1%<>1% UNLESS FILE.PPN%=P0% &
Add       \ R%=FNL%(UAA%,F%) &
this      \ P%(4%)=U%(R%,2%) &
line      \ P%(6%)=U%(R%,3%) &
          \ P%(7%)=U%(R%,4%) &
          \ P%(8%)=U%(R%,1%) &
          \ P%(9%),P8%=U%(R%,7%) &
          \ P%(10%)=U%(R%,5%) &
          \ P%(11%)=U%(R%,6%) &
          \ P%(I%)=0% FOR I%=12% TO 23% &
          \ P%(24%)=(U%(U%,5%) AND 255%)+(SWAP%(U%(U%,5%)) AND 255%) &
          \ P%(25%)=U%(FNL%(U%(U%,7%),F%),1%) &
          \ MSB.SI%=P%(11%) AND 255% AND P%(10%)=0% &
          \ GOSUB 10100 IF FCBLST%<>0% AND (S1% AND 1032%) &
          \ P%(4%)=(P%(4%)+P8%-1%) AND -P8% IF S2% AND 2048% &
          \ I%=U%(R%,0%) AND (NOT 15%) &
          \ IF I% THEN R%=FNL%(I%,F%) &
          \       P%(I%+11%)=U%(R%,I%) FOR I%=1% TO 7% &
          \       I%=U%(R%,0%) AND (NOT 15%) &
          \       IF I% THEN R%=FNL%(I%,F%) &
          \               P%(I%+18%)=U%(R%,I%) FOR I%=1% TO 5% &

2355        IF (S1% AND 5%) THEN &
                  X% = (USTAT% AND 128%)/128% &
          \       X% = (UAA% AND 8%)/4% UNLESS X% &
          \       PRINT #0%, MID(" *-",X%+1%,1%); &
          &
                  ! THE FILE IS MARKED FOR DELETION IF BIT 7 OF &
                  ! THE USTAT BYTE IS SET &
                  ! THE FILE IS HIDDEN IF BIT 3 OF THE ACCOUNTING &
                  ! LINK IS SET &
                  ! NOTE THAT THE MDEL FLAG TAKES PRECEDENCE OVER &
                  ! THE HIDDEN FLAG &
          &

300       CROSS.PROTECT%=-1%  (This is DEC feature patch 10.3.1)
```

6) Recompile the $DIRECT source.

7) If you are lucky enough to have this document on a release tape, load the tape and copy the file 'HIDE.BAS' from it. Otherwise, limber up your typing finger(s), and start typing in the listing from Ap-

pendix A. Of course, files can be hidden by CAREFULLY using $ODT from a privileged account, but I am not one to recommend it.

8) Compile the program obtained from step 7 and make it public and privileged. If desired, place the following two CCL commands into your system startup procedure:

   CCL HIDE-=LB:HIDE.*;PRIV 30000
   CCL UNHIDE-=LB:HIDE.*;PRIV 30000

   Change the file spec to indicate where this program resides on your system.

9) You are finished.
   The FIP patch will not take effect until the system is shut down and rebooted.

Appendix A

This section lists the program used to hide and unhide files on the system. It is suggested that you try to beg, borrow or steal a distribution tape instead of trying to type this whole mess in.

```
10        EXTEND
100     &
        !      G E N E R A L   D E S C R I P T I O N &
        &
        ! THIS PROGRAM ALLOWS A USER TO HIDE OR UNHIDE A FILE. BIT 3 &
        ! OF THE ACCOUNTING LINK WORD IN A FILE'S UFD ENTRY INDICATES &
        ! WHETHER OR NOT IT IS HIDDEN. &
        &
110     ! COPYRIGHT (C) 1981 &
        ! CALIFORNIA STATE UNIVERSITY, LONG BEACH &
        &
        ! THIS SOFTWARE IS BEING PROVIDED FREE OF CHARGE TO THE DEC USER &
        ! COMMUNITY AND MAY BE COPIED ONLY WITH THE INCLUSION OF THIS &
        ! COPYRIGHT NOTICE. THIS SOFTWARE, OR ANY OTHER COPIES THEREOF, IN &
        ! WHOLE OR IN PART, MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE &
        ! TO ANY OTHER PERSON EXCEPT FOR NON-PROFIT USE AND TO ONE WHO &
        ! AGREES TO THESE LICENSE TERMS. TITLE TO AND OWNERSHIP OF THIS &
        ! SOFTWARE SHALL AT ALL TIMES REMAIN IN CALIFORNIA STATE UNIVERSITY, &
        ! LONG BEACH. &
        &
        ! THE INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE WITHOUT &
        ! NOTICE AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY CALIFORNIA &
        ! STATE UNIVERSITY, LONG BEACH, OR ANYONE ELSE. &
        &
        ! WHILE SUPPORT IS NOT GUARANTEED FOR THIS PRODUCT, USERS WISHING &
        ! TO CONTACT THE AUTHOR SHOULD USE THE FOLLOWING ADDRESS: &
        &
        !    JOHN F. SANDHOFF &
        !    DATA PROCESSING &
        !    CALIFORNIA STATE UNIVERSITY, LONG BEACH &
        !    1250 BELLFLOWER BLVD &
        !    LONG BEACH, CA  90840 &
        &

900     ! &
        !      D I M E N S I O N S &
        ! &

910       DIM MATCH%(9), FIL%(9), M%(30) &

930       DIM #2, UFD%(3583,7) &
                ! UFD AS A VIRTUAL ARRAY &

1000    &
        !      M A I N L I N E   C O D E &
        &

1010      I$ = "V7.0-07A" &
                ! VERSION/EDIT &

1020      PRINT &
        \ PRINT "HIDE - File Hide Utility  "; I$ &
        \ PRINT FNERR$(0%) &
        \ PRINT &
                ! TELL THEM WHO WE ARE &

1030      GOSUB 18000 &
                ! GO GET INITIALIZED &

1090      PRINT "Do you wish to Hide or Unhide files (H or U) "; &
        \ INPUT X$ &
        \ X$ = LEFT(CVT$$(X$,39%),1%) &
        \ GOTO 1090 UNLESS X$ = "H" OR X$ = "U" &
        \ HIDE% = (X$ = "H") &
                ! FIND OUT WHAT THEY WISH TO DO &

1100      ERR.FLAG% = 0% &
        \ UNLESS IN% THEN &
                PRINT &
        \       PRINT "File(s) to "; &
```

EEC SYSTEMS

INTRODUCING LEX-11

WORD PROCESSING FOR THE VAX, PDP-11, AND LSI-11

Immediate availability on all DEC operating systems including VMS, RSTS/E, RSX-11M, RT-11 also under UNIX and TSX-PLUS. LEX-11 is a proven system with installations world wide. This easy to use system is flexible and includes many features such as single keystroke functions, menu driven, user definable menus, calculator, forms, list processing, data base management facilities, electronic mail and spelling dictionary. Complete turnkey systems available.

Local distributors wanted: OEM discounts.

EEC Systems
286 Boston Post Road
Wayland, MA 01778
(617) 358-7781/2 (617) 443-6376

CIRCLE 43 ON READER CARD

```
        \       PRINT "un"; UNLESS HIDE% &
        \       PRINT "hide"; &
                ! ERR.FLAG IS USED WITH INDIRECT FILES &
                ! IF WE'RE TALKING TO THE KEYBOARD (AND NOT AN INDIRECT &
                ! FILE), PROMPT WITH THE RIGHT QUESTION &

1105      INPUT LINE #IN%, FILE.SPEC$ &
        \ UNLESS IN% THEN &
                GOSUB 23000 &
        \       GOTO 1100 IF ERR.FLAG% &
        \       GOTO 1105 IF IN% &
                ! GET AN INPUT FILE SPEC. THEN: &
                !   IF INPUT IS CURRENTLY FROM THE KEYBOARD, THEN &
                !     SEE IF AN INDIRECT COMMAND FILE WAS SPECIFIED &
                !     IF SO, WE NEED TO GET A FRESH INPUT LINE (FROM THE FILE) &
                !  NOTE THAT AN INDIRECT FILE CAN ONLY BE SPECIFIED FROM &
                !  TERMINAL INPUT. AN AT-SIGN APPEARING INSIDE AN &
                !  INDIRECT FILE IS TREATED AS PART OF THE FILENAME. &

1110      CHANGE SYS(CHR$(6%)+CHR$(-23%)+FILE.SPEC$) TO M% &
        \ FILE.SPEC$ = RIGHT(FILE.SPEC$,LEN(FILE.SPEC$)-RECOUNT+2%) &
        \ FLAG2% = M%(29%) OR SWAP%(M%(30%)) &
        \ GOTO 1170 UNLESS (FLAG2% AND 1%) &
        \ IF FLAG2% < 0% THEN &
                PRINT "?Not a valid device - "; &
                        RAD$(M%(23%) OR SWAP%(M%(24%))); &
                        RAD$(M%(25%) OR SWAP%(M%(26%))); ": - Continuing" &
        \       GOTO 1170 &
                ! IGNORE THIS FILESPEC IF NO FILENAME &
                ! IF WE GOT A BAD DEVICE NAME, WARN THEM, THEN CONTINUE &

1120      DEV$ = "" &
        \ IF FLAG2% AND 8192% THEN &
                DEV$ = CHR$(M%(23%)) + CHR$(M%(24%)) &
        \       DEV$ = DEV$ + CHR$(M%(25%)+48%) IF M%(26%) &
        \       DEV$ = DEV$ + ":" &

1130      UNLESS (FLAG2% AND 128%) THEN &
                PPN% = MY.PPN% &
        ELSE &
                IF PRIV% THEN &
                        PPN% = M%(5%) OR SWAP%(M%(6%)) &
        ELSE &
                PRINT "?Cross-account not allowed - continuing" &
        \       GOTO 1170 &
                ! USE OUR PPN IF NONE SPECIFIED, ELSE &
                ! USE SUPPLIED PPN IF WE ARE PRIVILEGED, ELSE &
                ! ISSUE A COMPLAINT AND CONTINUE &

1140      PPN$ = "[" + NUM1$(SWAP%(PPN%) AND 255%) + "," + &
                        NUM1$(PPN% AND 255%) + "]" &
        \ UFD$ = DEV$ + PPN$ &
                ! BUILD THE PPN STRING &
                ! BUILD THE UFD SPECIFICATION &

1150      X$ = RAD$(M%(7%) OR SWAP%(M%(8%))) + &
                RAD$(M%(9%) OR SWAP%(M%(10%))) &
        \ X$ = X$ + RAD$(M%(11%) OR SWAP%(M%(12%))) IF (FLAG2% AND 8%) &
        \ X$ = X$ + "???" UNLESS (FLAG2% AND 8%) &
        \ FSPEC$ = CVT$$(UFD$ + LEFT(X$,6%) + "." + RIGHT(X$,7%),2%) &
        \ CHANGE X$ TO FIL% &
        \ WILD% = 0% &
        \ FOR I% = 1% TO 9% &
        \       FIL%(I%)=0% IF FIL%(I%)=63% &
        \       WILD% = -1% UNLESS FIL%(I%) &
        \ NEXT I% &
                ! CHANGE THE SCANNED FILENAME INTO A 9-CHAR STRING &
                ! ADD A WILD EXTENSION IF NONE SPECIFIED &
                ! CONVERT THE STRING TO BYTES &
                ! SET UP FOR WILDCARD MATCH FUNCTION &
                ! FLAG WHETTHER WE ARE DEALING WITH WILDCARDS &

1160      GOSUB 2000 &
                ! DO THE ACTUAL WORK &

1170      GOTO 1110 IF LEN(FILE.SPEC$) &
        \ GOTO 32767 IF CCL% UNLESS IN% &
        \ GOTO 1100 &
                ! FIGURE OUT WHERE TO DISPATCH TO: &
                !       1110 IF MORE TO DIGEST &
                !       END IF CCL ENTRY AND NO MORE &
                !       ELSE BACK FOR ANOTHER COMMAND LINE &

2000    &
        !       G E T   S O M E   W O R K   D O N E &
        &

2010      OPEN UFD$ FOR INPUT AS FILE 2%, MODE 16384% &
        \ IF CVT$%(MID(SYS(CHR$(12%)),13%,2%)) = 0% THEN &
                PRINT "?No files in account "; UFD$ &
        \ CLOSE 2% &
        \ RETURN &
                ! OPEN THE UFD, WRITE ENABLED &
                ! COMPLAIN IF NOTHING THERE &

2020      ERR.FLAG% = 0% &
        \ FOUND% = 0% &
        \ NEXT.LINK% = UFD%(0%,0%) &
        \ UCS% = UFD%(31%,0%) &
                ! INITIALLY SAY NO ERRORS AND NO FILES FOUND &
                ! GET LINK TO FIRST FILE &
                ! ALSO GET UFD CLUSTER SIZE &

2030      UNTIL ERR.FLAG% &
        \ LINK% = FNLINK%(NEXT.LINK%,UCS%) &
        \ GOTO 2040 UNLESS LINK% &
        \ USTAT% = UFD%(LINK%,4%) &
        \ NEXT.LINK% = UFD%(LINK%,0%) &
        \ GOSUB 3000 UNLESS (USTAT% AND 192%) &
        \ ERR.FLAG% = 1% IF FOUND% UNLESS WILD% &
        \ NEXT &
                ! LOOP UNTIL WE GET AN ERROR &
                ! RETRIEVE LINK TO NEXT FILE &
                ! EXIT LOOP IF NO MORE FILES &
                ! ELSE GET STATUS WORD AND NEXT LINK &
                ! GO PROCESS THIS FILENAME &
                ! FAKE AN ERROR (TO TERMINATE LOOP) IF WE FOUND A FILE &
                !   AND WE ARE NOT WILDCARDING (FOR SPEED) &
                ! CONTINUE LOOP (FALLS THROUGH IF ANY ERRORS) &

2040      CLOSE 2% &
        \ UNLESS FOUND% THEN &
                PRINT "File not found - "; FSPEC$; " - continuing" &
                        UNLESS WILD% &
        \       PRINT "No matching files - "; FSPEC$; " - continuing" &
                        IF WILD% &

2050      RETURN &

3000    &
        !       C H A N G E   T H E   F I L E ' S   S T A T U S &
        &

3010      P%(I%) = UFD%(LINK%,I%) FOR I% = 1% TO 3% &
        \ RETURN UNLESS FNMATCH% &
        \ FOUND% = -1% &
        \ FILNAM$ = RAD$(P%(1%))+RAD$(P%(2%))+"."+RAD$(P%(3%)) &
        \ IF WILD% THEN &
                PRINT FILNAM$; &
        \       INPUT X$ &
        \       RETURN UNLESS LEFT(CVT$$(X$,39%),1%) = "Y" &
                ! RETURN IF THIS FILE DOESN'T MATCH OUR SPECS &
                ! IF WILDCARDING, VERIFY THAT THIS FILE IS RIGHT &

3020      GOTO 3100 IF HIDE% &
        \ UNLESS (UFD%(LINK%,6%) AND 8%) THEN &
                PRINT "%File is not hidden" &
        \       RETURN &
                ! GO ELSEWHERE IF WE ARE HIDING FILES &
                ! COMPLAIN IF WE WANT TO UNHIDE AN UNHIDDEN FILE &

3030      UFD%(LINK%,6%) = UFD%(LINK%,6%) AND -9% &
        \ PRINT UFD$; FILNAM$; " unhidden" UNLESS WILD% &
        \ RETURN &
                ! ZERO THE 'HIDDEN' BIT &

3100      IF (UFD%(LINK%,6%) AND 8%) THEN &
                PRINT "%File is already hidden" &
        \       RETURN &
                ! COMPLAIN IF WE WANT TO HIDE A HIDDEN FILE &

3110      UFD%(LINK%,6%) = UFD%(LINK%,6%) OR 8% &
        \ PRINT UFD$; FILNAM$; " hidden" UNLESS WILD% &
        \ RETURN &
                ! SET THE 'HIDDEN' BIT &

3120      ERR.FLAG% = 1% &
        \ RETURN &
                ! USED FOR ERROR RETURN &

18000   &
        !       I N I T I A L I Z A T I O N &
        &

18100   IN% = 0% &
                ! SPECIFY INITIAL INPUT FROM THE CONSOLE &

18110     ON ERROR GOTO 19000 &
        \ Y$ = SYS(CHR$(6%)+CHR$(-7%)) &
                ! SET ERROR HOOK AND ENABLE CTRL/C TRAP &

18120     MY.PPN% = PEEK(PEEK(528%)+24%) &
        \ PRIV% = (PEEK(PEEK(520%)+2%) AND 1024%) <> 0% &
                ! GET OUR PPN &
                ! SEE IF WE ARE PERMANENTLY PRIVILEGED &

18130     RETURN &

19000   &
        !       E R R O R   R E C O V E R Y &
        &

19100     IF ERR=28% THEN &
                CLOSE 2% &
        \       GOTO 32767 &
                ! CLOSE UP SHOP (NEATLY) ON A CTRL/C &

19200     IF ERR=11% THEN &
                RESUME 32767 IF ERL=1090% OR (ERL=1105% AND IN%=0%) &
        \       RESUME 3120     IF ERL=3010% &
        \       IF ERL=1105% THEN &
                        IN%=0% &
        \               RESUME 1170 &
                ! HANDLE CTRL/Z ON INPUT: &
                !       DURING 'HIDE/UNHIDE' QUERY &
                !       DURING 'FILE' QUERY TO THE CONSOLE &
                !       DURING WILDCARD VERIFICATION &
                !       IF INPUT FROM AN INDIRECT FILE, AND EOF FOUND, &
                !         RESET INPUT CHANNEL AND RESUME &

19300     IF ERR=5% AND ERL=2010% THEN &
                PRINT "?Non-existant account - "; UFD$ &
        \       RESUME 2050 &
                ! PROBLEMS FINDING THIS UFD &

19400     IF ERL=1110% THEN &
                PRINT "?Invalid file specification - terminating" &
        \       FILE.SPEC$ = "" &
        \       RESUME 1170 &
                ! PROBLEMS DURING THE FILE STRING SCAN &

19500     RESUME 23200 IF ERL=23100% &
                ! HANDLE PROBLEMS WITH INDIRECT COMMAND FILE &

19900     CLOSE 2% &
        \ PRINT "?Untrapped error - report to system manager" &
        \ PRINT CVT$$(RIGHT(SYS(CVT%$(1545%)+CHR$(ERR)),3%),4%); &
                " at line"; ERL &
        \ GOTO 32767 &
                ! OOPS... &
```

the IBM Solution
HASPBOX™

SOFTWARE RESULTS CORPORATION

the CDC Solution
UT200BOX™

RSTS•VAX•RSX!
DATA COMMUNICATIONS

- INSTALLATIONS THROUGHOUT NORTH AMERICA AND EUROPE
- FRONT END PROCESSOR-BASED FOR LOW OVERHEAD
- FULL LINE UTILIZATION FOR HI-THRUPUT (UP TO 19.2 KB)
- USER INTERFACE DESIGNED FOR RELIABILITY AND SIMPLICITY

614/421/2094   COLUMBUS, OH 43212 USA   TWX 810 482 1631

CIRCLE 44 ON READER CARD

```
20000   &
        !       D E F I N E D   F U N C T I O N S &
        &

21000     DEF* FNMATCH% &
                ! FUNCTION CHECKS FOR A MATCH IN FILENAMES &

21010     FNMATCH% = 0% &
        \ CHANGE RAD$(P%(1%))+RAD$(P%(2%))+RAD$(P%(3%)) TO MATCH% &
        \ GOTO 21020 IF MATCH%(I%) <> FIL%(I%) IF FIL%(I%) &
                        FOR I% = 1% TO 9% &
        \ FNMATCH% = 1% &
                ! ASSUME THAT WE WILL FAIL &
                ! SEE IF WE ARE RIGHT &
                ! IF NOT, CHANGE OUR MIND AFTER THE FACT &

21020     FNEND &

22000     DEF* FNLINK%(L%,C%) = &
                (((L% AND 3584%)/512%)*C% + &
                (SWAP%(L% AND -4096%)/16%))*32% + &
                ((L% AND 496%)/16%) &
                ! FUNCTION DECODES LINK INFORMATION INTO A &
                ! SUBSCRIPT FOR OUR ARRAY &

23000   &
        !       C H E C K   F O R   A N   I N D I R E C T   F I L E &
        &

23100     RETURN UNLESS ASCII(FILE.SPEC$) = 64% &
        \ FILE.SPEC$ = RIGHT(FILE.SPEC$,2%) &
        \ OPEN FILE.SPEC$ FOR INPUT AS FILE 4%, MODE 8192% &
        \ IN% = 4% &
        \ RETURN &
                ! SEQUENCE IS IMPORTANT IN CASE OF ERRORS: &
                !  MUST OPEN THE FILE BEFORE REASSIGNING IN% &

23200     PRINT "?Indirect file error - "; FNERR$(ERR) &
        \ ERR.FLAG% = -1% &
        \ RETURN &
                ! HANDLE ERRORS OPENING THE FILE &

24000     DEF* FNERR$(E%) = CVT$$(RIGHT(SYS(CVT%$(1545%)+CHR$(E%)),3%),4%) &
                ! RETURN RSTS ERROR TEXT FOR SPECIFIED MESSAGE &

30000   ! &
        !       C C L   E N T R Y   P O I N T &
        ! &
        ! &
        ! WE ALLOW TWO CCL'S TO CALL US, 'HIDE' AND 'UNHIDE' &
        &

30100     GOSUB 18000 &
        \ Y$ = CVT$$(SYS(CHR$(7%)),188%) &
        \ GOTO 1000 UNLESS LEN(Y$) &
                ! GO GET INITIALIZED &
                ! GET CORE COMMON AND CLEAN IT UP &

30110     IF LEFT(Y$,4%) = "HIDE" THEN &
                HIDE% = -1% &
        \       Y$ = RIGHT(Y$,5%) &
        \       GOTO 30300 &
                ! IT LOOKS LIKE THE HIDE COMMAND. CLEAN THINGS UP &
                ! AND LET'S CHECK &

30120     IF LEFT(Y$,6%) = "UNHIDE" THEN &
                HIDE% = 0% &
        \       Y$ = RIGHT(Y$,7%) &
        \       GOTO 30300 &
                ! IT LOOKS LIKE THE UNHIDE COMMAND. CLEAN THINGS UP &
                ! AND LET'S CHECK &

30130     PRINT "?Invalid CCL entry" &
        \ GOTO 32767 &
                ! WHAT DID HE SAY? &

30300     GOTO 1100 UNLESS LEN(Y$) &
        \ GOTO 30130 IF ASCII(Y$) <> 32% &
                ! TREAT AS (ALMOST) NORMAL ENTRY IF NO FILESPEC &
                ! BAD CCL COMMAND ALMOST FOOLED US &

30310     FILE.SPEC$ = RIGHT(Y$,2%) &
        \ CCL% = -1% &
        \ GOSUB 23000 &
        \ GOTO 32767 IF ERR.FLAG% &
        \ GOTO 1110 UNLESS IN% &
        \ GOTO 1100 &
                ! THIS IS A VALID CCL ENTRY. FLAG IT AS SUCH. (MAKE SURE &
                ! THAT WE REMOVE THAT LEADING BLANK). THEN: &
                !       CHECK FOR AN INDIRECT COMMAND FILE SPECIFICATION &
                !       PROCEED NORMALLY IF NONE, ELSE &
                !               JUMP INTO THE RIGHT SPOT &

32767     END
```

Appendix B

Listing all Hidden Files For Backup Purposes

The method presented here is very simple, very short, and very crude. It is provided in this form due to a lack of time and space. The release of a very fast and efficient system cataloging program is planned for the near future.

Since none of the standard backup packages recognize the existence of hidden files, some other means of keeping track of them is required. The best way is to have some routine that can create an indirect command file recognizable by the HIDE program that contains all of the currently hidden filenames. Naturally, care should be taken to assure that this file does not get released to the user public.

After the two patches are made to $DIRECT, a hidden file is flagged by having a minus sign ('-') appended to its name. This procedure simply creates a complete directory of the system and then runs a program that looks for the minus sign. Presented here is a sample BATCH control file, and the program that scans for the minus sign.

Following is a sample BATCH control file. Note that this example is set up to process two disks, one with a logical name of 'SYS:' and the other known as 'AUX:'. All output is directed to the 'AUX:' device. Make changes as appropriate to your system.

```
$JOB/CCL/NOLIMIT/ERROR:WARNING
RUN $PIP
AUX:SYSLST.DIR/DE/WO:NOWARN
AUX:AUXLST.DIR/DE/WO:NOWARN
$EOD
RUN $DIRECT
AUX:SYSLST.DIR=SYS:(*,*)
AUX:AUXLST.DIR=AUX:(*,*)
$EOD
RUN HIDDEN
AUX:SYSLST.DIR

RUN HIDDEN
AUX:AUXLST.DIR

RUN $PIP
AUX:SYSLST.DIR/WO
AUX:AUXLST.DIR/WO
$EOD
$EOJ
```

The following is a simple program to process the directory listing. It creates a file that can be supplied to HIDE as an indirect command file.

```
10          EXTEND
100           PRINT "Input directory listing file"; &
            \ INPUT LINE A$ &
            \ OPEN A$ FOR INPUT AS FILE 2%, MODE 8192%
110           DFLT$ = CVT$$(DATE$(0%),32%) &
            \ DFLT$ = MID(DFLT$,1%,2%) + MID(DFLT$,4%,3%) + ".HID" &
            \ Print "Output command file to create <"; DFLT$; "> "; &
            \ INPUT LINE B$ &
            \ B$ = CVT$$(B$,39%) &
            \ B$ = DFLT$ UNLESS LEN(B$) &
            \ OPEN B$+"<188>" AS FILE 4%, MODE 2%
120           ON ERROR GOTO 19000
200           WHILE 1% &
            \ INPUT LINE #2%, IN$ &
            \ IN$ = CVT$$(IN$,4%) &
            \ X% = INSTR(1%,IN$,"[") &
            \ IF X% THEN &
                    X% = X% - 1% UNTIL X% = 1% OR MID(IN$,X%,1%) = " " &
            \       X% = X% + 1% IF MID(IN$,X%,1%) = " " &
            \       PPN$ = MID(IN$,X%,INSTR(X%,IN$,"]")-X%+1%)
210           GOSUB 500 IF MID(IN$,11%,1%) = "-" &
            \ NEXT
300           CLOSE 2%, 4% &
            \ PRINT &
            \ PRINT "Total of"; CNT%; "hidden files" &
            \ GOTO 32767
500           IF PPN$ <> LAST.PPN$ THEN &
                    PRINT #4% &
            \       LAST.PPN$ = PPN$
510           PRINT #4%, PPN$; LEFT(IN$,10%) &
            \ CNT% = CNT% + 1% &
            \ RETURN
19000         IF ERR=11% THEN &
                    RESUME 300 IF ERL=200% &
            \       RESUME 32767
19999         ON ERROR GOTO
32767         END
```

** End of Document ** ♥

The RSTS/E System Manager

By Jeffrey R. Harrow, 485 Creekview Dr., Stone Mountain, GA 30083

Welcome to another issue of the RSTS/E Pro. Let's begin this month by talking about some SPR responses:

In an earlier issue I pointed out that Datatrieve's error messages did not contain the "%" or "?" initial character utilized by the RSTS/E BATCH and ATPK processors. This omission is causing undetermined results while your control file continues processing after a DTR error has occurred.

DEC indicated that:

> "Datatrieve-11 is a layered product that is available on six (6) different operating systems. The same message is used by all. Therefore, it is not appropriate that operating system specific things be included . . ."

Now considering what we pay for these layered products, **purchased specifically for a given operating system**, I continue to feel that whatever operating system dependent modifications are required **should well be made**. How do you feel about this? Send some letters for publication and perhaps we can get DEC's attention.

In any event, I was provided with a site-by-site workaround:

The DTR message file (LB:QUERY.MSG) is an RMS Indexed file and can be modified as you see fit (of course there are a LOT of messages in there and you will have to determine the severity of the event which causes each message and prepend the appropriate "%" or "?" characters).

The record definition follows:

```
01      MESSAGE.
        03      MESSAGE-NUMBER PIC 9(4);
                USAGE COMP;
                EDIT-STRING IS -(5)9.
        03      MODULE PIC IS X(4).
        03      SUBROUTINE PIC X(6).
        03      TYPE PIC X.
        03      TEXT PIC X(60).;
```

Have fun.

EDT Version 2, when put into "C"hange (keypad) mode, sets a VT100 terminal to ANSI mode. When EDT exits, it **does not** return the terminal to VT52 mode if that is how it was originally set.

A large body of older software (including TECO's VTEDIT and several Word Processing Products) assume a terminal in VT52 mode. Now, DEC is advising:

> ". . .The VT100 was not intended to be used in VT52 compatibility mode except to replace VT52s."
>
> ". . .suggest that he leave the VT100s in native mode, and place some code at the front of his programs that detects the presence of a VT100 and switches it to VT52 mode."

Well, that's one way to take care of the problem **if you have the source code (or a hook) for the program.** Of course there are many programs puchased from another vendors where this is not the case. Most of us operate in a mixed environment. [See "Tips & Techniques", page 60.]

As no recommended "standard" for dealing with these bi-modal terminals has been set up, consider the following:

> It is really quite simple for any program which intends to utilize the new "ANSI" functionality of a VT 100 to, during program initiation, **query the terminal** and **remember** what type it actually is, then **change** the terminal to ANSI mode if it is a VT100, and **just prior to termination, change the terminal back** to VT52 mode **if** that was its original configuration.

This methodology would allow users with "older" unchangable software to leave their terminal in VT52 mode, and will allow newer software, with very little additional overhead, to use the newer VT100 features. Remember, a philosophy which attempts to consider the **global** picture, rather than a picture selective to a given program, will help prevent incompatibility and its attendant problems at a later time.

The NFT (DECnet Network File Transfer Utility) documentation indicates that the "/BL" (Block mode) switch must be used for non-ASCII files without recorded "attributes" and indicates, as an example, files with .OBJ extensions.

Well, it turns out that MAC (the RSX assembler under RSTS/E) turns out .OBJ files **with** attributes. DEC's response is:

> "What it (the documentation) should further state is that it should NOT be used for files **with** record attributes (ie. MAC .OBJ files). The problem is that those utilities which rely on the record attributes to indicate the correct EOF pointer with a variable or Fixed length file get confused when you transfer a file that **has** attributes **with** the "/BL" switch. Those utilities (like RT11's MACRO.SAV) which **don't** use record attributes rely on the binary data alone to give it the EOF pointer, and **must** be copied with the "/BL" switch to preserve the binary data block for block. The proper way to transfer .OBJ files is to use the "/BL" switch **only** with **un**attributed files."

A problem has been identified where a program which has opened an RMS Relative or Indexed file and has CTRL/C Trapping (FIP Syscall -7) enabled will, once a CTRL/C is typed, **stop all output (as in CTRL/O mode), even across program CHAINs, until the next INPUT statement is executed** (and then you only get the "?" prompt).

The TSC **was** able to reproduce this, but was unable to provide me with a workaround. I've submitted an SPR and will keep you informed.

I'd like to bring to your attention **The BASIC Primer**. This is a Computer Managed Instruction (CMI) course from DEC which includes various books and workbooks plus a set of programs, written in BASIC, which create, administer, and record results of tests for the modules in the course.

The concept of CMI is **great**, but this software package presents a **very serious** problem:

The installation procedures for the software indicate that the main body of the programs go into a single "CMI" account. You then put a copy of the parameter file (indicating, among other things, the account and name of the program to be run) and a copy of the **initial** program to be run into **each** account from where students will access the CMI package.

The student then runs the CMI program **in his account** which reads the parameter file **in his account**, determines where it must CHAIN to, and does the CHAIN.

Now for the fun . . . If the programs in the central CMI account are NOT given Temporary Priv., all of the files from where the tests are created **and** the student record files MUST BE GIVEN A PROTECTION CODE OF FORTY AND ZERO (respectively) to allow users across the system the necessary read and write access. This means that **anyone** on the system can peruse the test files, can peruse his (or others!) progress in the records files, **and can change** information in these files if he is so inclined! Needless to say prudence, if not the Privacy Act requirements, make this undesirable.

Well, perhaps we can, as indicated (in general terms) in the installation manual, give the programs a <232> Protection Code including Temporary Priv. Consider this: The user now has a program in his account with Temp. Priv. which reads a filename **within a file in his account** and then CHAINs to it. **Your user can then modify the program name in the parameter file in his account and execute ANY program on the system, and retain the Temporary Priv. attribute! In effect, the entire security of your system is compromised!!!!!**

When I pointed this out to Pete McVey of DEC's Educational Service Product Line (the source of this package) he indicated that they "**. . .totally agree with your technical assessment of the condition of the software. . .**", that "**. . .it was not intended to run on a secure system. . .**" and that **they have no intention of changing it at this time.**

. . . continued on page 92

SEVCO

PDP 11

VAX 11/780

WE SPECIALIZE IN DEC HARDWARE, CABLES COMPLETE SYSTEMS

SAVE
SAVE
SAVE
IMMEDIATE DELIVERY

MEMORY OPTIONS MODULES

WE BUY SELL TRADE AND APPRAISE ALL USED COMPUTER EQUIPMENT

NEW AND USED ALL EQUIPMENT CARRIES 30 DAY WARRANTY

SEVCO

MINI-COMPUTER BROKERS
SEVCO, INC. - 200 SW CUTOFF; NORTHBORO, MA 01532
(617) 393-3025 / TWX 7103476527

CIRCLE 45 ON READER CARD

RSTS/E MONITOR INTERNALS, Part 3

. . . continued from page 47

4.2 LOGICAL DEVICE TABLES

Verifying device names and unit numbers and associating these logical names with physical device DDBs when a device is opened is performed using a series of related tables. The entries in each table are ordered so that once an entry is found in one table the associated information in the other tables will be found at the same offset.

The following figure shows the relationship between the device tables:

```
 DEVNAM     DEVCNT     DEVPTR       UNTCNT
 ------     ------     ------       ------
| SY:  |-->| SY:  |-->| SY:  |--   :      :
|------|   |------|   |------|  |  :      :
|      |   |      |   |      |   ->|------|
| Disk |-->| Disk |-->| Disk |---->|  #0  |
|      |   |      |   |      |     |------|
|------|   |------|   |------|     :      :
|  KB  |-->|  KB  |-->|  KB  |     |------|
|------|   |------|   |------|     |  #n  |        DDBs
|      |   |      |   |      |      ------        -------
| Non- |-->| Non- |-->| Non- |               --->| KB n  |
| Disk |   | Disk |   | Disk |      DEVTBL  |     -------
|      |<- |      |   |      |      ------  |     -------
|------| | |------|    ------      |  KB  |-   ->| Dev 0 |
|      |  -|      |                |------|   |   -------
| Syn  |-->| Syn  |                | Non- |---    -------
|      |   |      |                |  KB  |----->| Dev n |
|------|    ------                  ------        -------
|  -1  |
 ------
```

4.2.1 DEVNAM — Device Name Table

The device name table (DEVNAM) contains an entry for every disk type supported by RSTS and an entry for every other device generated into this system. It also contains a list of synonyms for physical device names, such as MT: for MM:.

All of the entries in DEVNAM consist of a pair of ASCII characters stored in a word. Each disk name and the name of each configured device is included. DEVNAM has the following format:

Offset	Symbol	Description
-2		This word always contains the ASCII value for "SY". It is used for specifying the system disk structure, SY:. It is at an offset of -2 to show that it is not a normal device.
0-16	DEVNAM	These nine words contain the names of all the disks supported by RSTS in the order: DC, DF, DS, DK, DL, DM, DP, DR, DB. All disk names are included, even if the device is not configured in this system.
18	DEVNKB	This word contains the device name "KB" and corresponds to all keyboard type devices, including pseudo-keyboards.
20		This word contains the device name "NL", the null device.
22		The device names for all remaining devices configured in this system begin at this word.
+n	DEVSYN	Beginning at this word is a list of synonyms for the devices configured in this system. These synonyms are: TT=KB, CR=CD, MT=MM, MT=MS, DX=DY.
+n+m		This word contains a -1 to show the end of the table.

4.2.2 DEVCNT — Device Unit Count Table

The device unit count table (DEVCNT) contains one entry for each entry in DEVNAM (except the terminating -1). Each entry is the maximum unit number for the corresponding device in DEVNAM. A maximum unit number of -1 indicates that the associated disk device is not configured in this system.

DEVCNT has the following format:

Offset	Symbol	Description
-2		This word contains a 0 to specify that SY0: is the only explicit unit number allowed for SY:.
0	DEVCNT	The nine words starting here correspond to the maximum unit number for each corresponding disk type in DEVNAM. A value of -1 indicates that the corresponding disk type does not exist on this system.
18		This word contains the maximum unit number for KB type devices, including pseudo-keyboards.
20		This word always contains a 0 to specify that only unit 0 may be specified for NL:.
22		Each following word corresponds to the maximum unit number for the corresponding device in DEVNAM.
+n		Each word beginning at an offset equivalent to DEVSYN is a pointer to the appropriate physical name in DEVNAM for that synonym.

4.2.3 DEVPTR — Device Information Pointer Table

The device information pointer table (DEVPTR) contains pointers into one of two other tables. In the case of disk devices DEVPTR points into the disk unit status table, UNTCNT. For non-disk devices DEVPTR points into the device retrieval table, DEVTBL.

In both cases the entry pointed to is the information for unit zero of the associated device. If more than one unit is present for a device, the entries for the additional units immediately follow the unit 0 entry. See section 4.2.4 and 4.2.5 for more information on UNTCNT and DEVTBL.

DEVPTR has the following format:

Offset	Symbol	Description
-2		This word contains a pointer to the entry in UNTCNT corresponding to SY0:, the device from which RSTS was bootstrapped.
0	DEVPTR	The nine words beginning here contain pointers to the entries in UNTCNT for unit 0 of the corresponding disk type. If a disk type does not exist on this system its pointer is to the entry in UNTCNT for the next valid disk type. If no following disk types are valid this word points to a dummy entry at the end of UNTCNT.
18		The words beginning here contain pointers to the entries in DEVTBL for unit 0 of each non-disk device in DEVNAM.

4.2.4 UNTCNT — Disk Unit Status Table

The disk unit status table (UNTCNT) contains information about each disk unit configured in this system. There is one word for each disk type and unit. These words are ordered by disk type in the same order as the disk names in DEVNAM and, within disk type, by unit number. Each word contains the status and current open count for the corresponding disk unit.

The last entry in the table is followed by a word containing 177001(octal), which denotes a non-mounted disk which cannot be mounted. This is used for proper termination of DEVPTR entries for non-existent disk types (see section 4.2.3).

Each word of UNTCNT has the following format:

Bit	Symbol	Description
0-8	UC.CNT	Count of currently open files.
9	UC.TOP	Order the directory with new files first.
10	UC.WLO	The disk is mounted read only.
11	UC.DLW	Update access date with date last written.
12	UC.NFS	Non-file structured processing is currently occurring.
13	UC.LCK	Don't allow non-privileged opens on this disk.
14	UC.PRI	This disk has a private pack mounted.
15	UC.MNT	This disk does not have a pack mounted. Bits 9-14 are meaningless.

4.2.5 DEVTBL — Device Retrieval Table

The device retrieval table (DEVTBL) contains a pointer to the DDB for each unit of each non-disk device on the system. The entries are ordered in the same order as DEVNAM. The pointers for each device with more than one unit are stored in sequential memory locations, beginning with the pointer for unit 0.

DEVTBL has the following format:

Offset	Symbol	Description
0	DEVTBL	The words beginning here are the pointers to the DDBs for the each terminal device and pseudo-keyboard unit, beginning with unit 0.
+n	DEVTBE	The words beginning here are the pointers to the DDBs for each unit of the non-terminal devices, beginning with NL0:.

4.2.6 Examples of Device Table Use

The following examples show how to access the information in the device tables from BASIC-PLUS:

```
10  DEVNAM%=SWAP%(CVT$%(MID(SYS(CHR$(6%)+CHR$(-12%)),5%,2%)))
                !Get address of DEVNAM table
20  DEV0KB%=SWAP%(CVT$%(MID(SYS(CHR$(6%)+CHR$(-12%)),9%,2%)))
                !Get offset to first non-disk device in DEVNAM
```

PDP-11® & VAX® users...

ComDesign's TC-3 Concentrates Multiple Terminals and Printers on a Single Telephone Line Cost-Effectively for the PDP-11 and VAX.

■ CONCENTRATION
Many terminals—one line

■ COST SAVINGS TO 40%
Totally eliminates DZIIs and cables

■ PROVEN RELIABILITY
TC-3s are in use world-wide

■ ERROR CORRECTION
Data checked with all errors eliminated automatically

■ STATISTICAL MULTIPLEXING
Efficiencies to 400%

■ FAST ECHO RESPONSE
—at last !

■ BUILT-IN SYSTEM MONITORS
Real-time data and signal displays

■ POWERFUL DIAGNOSTIC PANEL
Modem loop-back, unit self-tests, and terminal diagnostics

Put a TC-3 on Your UNIBUS® and Get a DZ11 with Stat Muxing too!

"We are impressed . . . have already recommended the unit to a number of other RSTS/E users and shall continue to do so . . ."
Intercontinental Forwarders, Inc.

See us at ACCESS-11, Booth #145

For further information and the sales office for your area
Call **toll-free (800) 235-6935** or in Calif. (805) 964-9852

ComDesign

ComDesign, Inc
751 South Kellogg Avenue
Goleta, California 93117

® Registered Trademarks of DEC

CIRCLE 46 ON READER CARD

BACmac can do it all!

BAC into RTS / BAC into MAC / BAC into BAS

BACmac is a unique software tool, running under RSTS/E, which provides the following conversions:

■ translation from Basic-Plus "compiled" back to Basic-Plus source code (only the comments will be missing)

■ translation from Basic-Plus into Macro source code, which compiled under RSTS runs faster than Basic-Plus

■ translation from Basic-Plus into Macro source code which may be compiled under RSTS for execution under RT11 — a migration facility

■ translation from Basic-Plus into a RUN-TIME-SYSTEM. Now you can write an RTS in Basic-Plus. The ideal solution to memory thrashing due to "multi-copy" applications programs.

RSTS/E, RT11, Macro-11 and Basic-Plus are trademarks of Digital Equipment Corporation.

Please write for more information

TELECOM

Telecom Computer Systems, Inc.
P.O. Box 03285
Portland, Oregon 97203
503/286-5122

CIRCLE 47 ON READER CARD

```
30  DEVCNT%=SWAP%(CVT$%(MID(SYS(CHR$(6%)+CHR$(-3%)),5%,2%)))
\   DEVCNT.KB%=PEEK(PEEK(DEVCNT%+DEV0KB%))
            !Get maximum unit number allowed for KB devices
40  DEVPTR%=SWAP%(CVT$%(MID(SYS(CHR$(6%)+CHR$(-3%)),7%,2%)))
            !Get address of DEVPTR table
50  DDBPTR.KB2%=PEEK(PEEK(DEVPTR%+DEV0KB%)+4%)
            !Get pointer to DDB for KB2:.  Note: DDB pointer
            !for KB2: is at entry in DEVTBL pointed to by KB
            !entry in DEVPTR offset by 2 times the unit number.
60  DK0.MOUNTED%=PEEK(PEEK(DEVPTR%+6%)) >= 0%
            !Determine if DK0: is currently mounted.  Return
            !TRUE if mounted.
```

In the next issue: SEND/RECEIVE and CCL

SPECIAL OFFER: A new, updated wall chart showing the linkages between all tables in the RSTS/E monitor is available for $2.00. Send your requests to Mike Mayfield, Northwest Digital Software, Box 2-743, Newport, WA 99156.

DUMPER-FILE BACKUP UTILITY

- Much faster than BACKUP
- Supports RSTS Large-File feature
- Ideal for big databases
- Proven and reliable, easy to use

etc ENTERPRISE TECHNOLOGY CORPORATION
305 Madison Avenue
New York, N.Y. 10165
(212) 972-1860

CIRCLE 48 ON READER CARD

DEC BEST VALUES

PRE-OWNED DEC EQUIPMENT

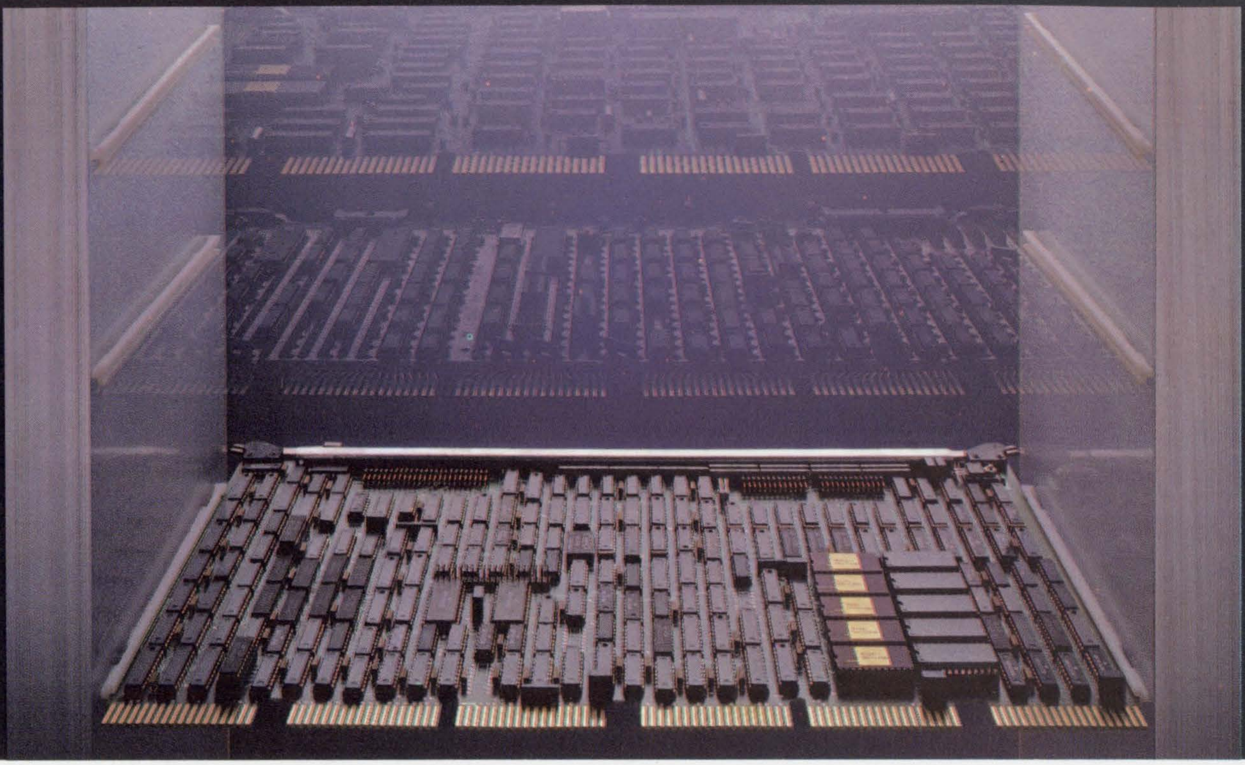
BUYING AND SELLING

SYSTEMS • CPU's • PERIPHERALS • TERMINALS
OPTIONS • MEMORY • COMPATIBLES

CALL DICK BAKER (305) 979-2844

dataware incorporated
Carico Center
2845 NW 62nd Street
Ft. Lauderdale, Florida 33309
Telephone (305) 979-2844

CIRCLE 49 ON READER CARD



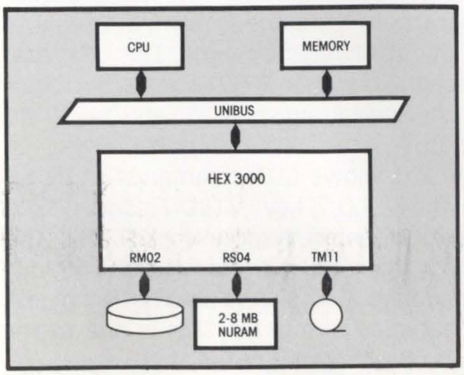
HEX 3000™. The first controller that does the work of three.

National unleashes the new hex-wide DEC®-compatible peripheral controller that does it all from just one slot.

Why waste valuable chassis space on peripheral controllers that can only interface a single device type to DEC's UNIBUS®? This one-to-one approach can only drive your UNIBUS system costs upward and its overall reliability downward.

Now there's no reason to put up with this inefficiency. Now there's the HEX 3000 controller.

The HEX 3000 is our new hex-wide peripheral controller that simultaneously handles up to four RM02 80 MB disk drives, four TU10 ½" tape drives and 8 MB of our NURAM™ semiconductor disk. All from just a single slot.



Cuts costs in the factory and in the field. By using one board to do the work of three or more—with no degradation—you save a bundle in a number of ways.

Your most immediate savings is the elimination of two comparably priced controller boards. But farther down the line, the HEX 3000's upgrade flexibility lets you expand a configuration less expensively because you don't necessarily need to buy additional boards or a larger chassis.

This also means your spares inventory costs are cut way back. Not only in the factory, but in the field as well.

And at the same time, the HEX 3000 increases your system's reliability because it uses far fewer ICs to do the whole job (198 in all).

What these and other cost-benefits all boil down to is a rebalancing of your system operational costs. The HEX 3000 brings the price of I/O processing back in line with the costs of instruction processing and memory. Which makes the old one-to-one approach a thing of the past.

It all stems from our XPU™ architecture. The HEX 3000 is based on our advanced Transfer Processing Unit (XPU) design concept.

The XPU's generalized intelligence emulates the transfer processing logic of DEC's RM02 disk, TU10 tape and RS04 fixed head disk controllers. So it's fully hardware and software compatible with any UNIBUS system.

By consolidating this logic into a single powerful microengine—and with the aid of on-board multi-sector buffers—the HEX 3000 can simultaneously transfer data at the rates of 1.2 MB/sec (80 MB disk), 320 KB/sec (1600 BPI tape) and 500 KB/sec (8 MB NURAM fixed head disk emulator).

And only one UNIBUS load services all three device types.

Practicality prevails. The HEX 3000's XPU architecture further enhances its overall reliability because it utilizes only proven technologies. There are no risky new and exotic parts to complicate the matter. Just good, solid and efficient design. From the same company that brought solid reliability to DEC add-in memory.

Leave it to the Practical Wizards to add a whole new dimension to inner space.

For complete details on our HEX 3000 controller, simply send in the coupon below or call Cindee Forrest toll-free at (800) 538-8510. In California call (408) 736-6994.

HEX 3000, NURAM and XPU are trademarks of National Semiconductor Corporation.
DEC and UNIBUS are trademarks of Digital Equipment Corporation.

The HEX 3000's efficiency and practicality is just what I need to cut my costs down to size.
Please send me:
□ HEX 3000 information.
□ NURAM information.
□ DEC Add-in Memory information.

NAME____________________

COMPANY____________________

ADDRESS____________________

CITY________ STATE______ ZIP______

PHONE____________________

National Semiconductor Corporation
2900 Semiconductor Drive
Mail Stop 7C265
Santa Clara, CA 95051

National Semiconductor
The Practical Wizards of Silicon Valley

CIRCLE 50 ON READER CARD

With this issue we are pleased to present our Editor Editor, David Spencer. David will be writing a series of articles on Editors for RSTS and other operating systems.

VTEDIT.TEC: EDT V2.0 Emulated With TECO

By David Spencer, Infinity Software Corporation

We heavy users of TECO and VTEDIT were both elated and deflated with the arrival of the new EDT version 2.0 (also known as ED2). The speed and capabilities of EDT as a screen editor made it far too attractive to pass up. Unfortunately, the keypad layout of EDT is quite different from the layout of VTEDIT. This would normally cause no problems if VTEDIT was orphaned.

However, TECO has unique editing qualities that EDT can't even come close to emulating in its current implementation. And occasionally returning to VTEDIT to perform some editing would give me a strong case of vertigo.

A Proposed Solution

I have written a new VTEDIT macro to emulate the standard EDT. True, all the EDT keys could be redefined to emulate VTEDIT instead. But that is a short-sighted solution. By knowing the EDT keypad layout, one can move to other operating systems and still be able to instantly use the standard editor on that system; EDT.

Intent of the Rewrite

My intent in rewriting VTEDIT was not to replace EDT, but to allow users of EDT and TECO to smoothly move between the two, with heavier emphasis on use of EDT.

I have attempted to incorporate all of the EDT keypad commands. I have, as well, also tried to mix in some of the more useful VTEDIT keys. As a result, all of the EDT tab level keys have been left in the cold, and quite a few VTEDIT commands have been left out as well.

The VTEDIT described in this acticle is a completely new program, and is similar to the original VTEDIT in name only. However, a deep debt of gratitude is due to the developers of the first VTEDIT. Without their code to examine, my development in TECO would not hardly be as great.

Limitations

The VTEDIT described operates ONLY on VT100 type terminals. This was a conscious decision. The VT52 is old technology, which, by the way I understand DEC no longer manufactures. At the risk of hurting some users, I chose to simplify the task of writing VTEDIT and supported only the VT100. (However, a VT52 only version would be fairly easy.)

Keys That Operate Differently

This is a list of keys that operate differently between the two editors.

1. Delete key
   Delete in ED2 deletes single characters at a time. When used in a GOLD-arg format, delete is used for editing the numeric amount. VTEDIT accepts an argument to the delete key for multiple character deletes.
2. Right and Left arrows
   EDT version 2.0 properly maintains the current position in the line. The VTEDIT macro arrow keys sometimes lose the correct position in the line when encountering tabs. The current implementation has a quick routine that could be improved. The trade-off is, of course, the extra overhead required to handle these special cases.
3. GOLD PF3
   The GOLD-PF3 key in both ED2 and VTEDIT allow entry of a search string. In addition, VTEDIT allows editing of the previous search string as in the DEC supplied version of VTEDIT. VTEDIT supports direction changing in search with the keypad "4" and "5" keys. The new VTEDIT, however, will not terminate a search argument on an arrow key, or one of the four "PF" keys.
4. Keypad 6
   In the standard version of ED2, the Keypad-6 key is used to cut a selected region to the PASTE buffer. VTEDIT cuts the selected region to a named register. To use this command, select a region. Next, hit the Keypad-6 key, and then the single letter name of your Q-register. Note VTEDIT does not prevent cutting to the Q-register "I", which houses the VTEDIT macro.
5. GOLD keypad 6
   In ED2, this combination pastes the "PASTE" buffer into the current position in the MAIN buffer VTEDIT operates in the same fashion, but requires the single character register name. This character does not echo. Note that VTEDIT does not allow pasting of the special registers "*" and "—".
6. GOLD Keypad 7
   EDT version 2.0 allows EDT commands to be executed via the GOLD-7 key. VTEDIT allows TECO commands to be entered and executed with this key. Note that input must be terminated with one of the keypad keys. Escape also does not echo until a second character is entered. This is due to the keypad character checking code.
7. GOLD Keypad 8
   In ED2, the GOLD-8 key is used for filling regions of text to a given margin. Because of the complexity involved, the VTEDIT macro does not implement this function. Instead, a page (in TECO, "P") command replaces this key.

8. GOLD Keypad Enter
   The GOLD-Keypad-Enter key is used for replace of selected region with the paste buffer in ED2. VTEDIT uses the Q-register "P" as the replace buffer, but does not assign it. In order to load the replace buffer, use one of the register loading commands on register "P".
9. CTRL/Z
   The CTRL/Z key in ED2 returns to line editing mode. CTRL/Z in VTEDIT returns to normal TECO editing.
10. Word delimiters
    In ED2, the standard word delimiter set is space, carriage-return, and form-feed. VTEDIT uses the old delimiter set of space, tab, comma, carriage-return, and line-feed.
11. Entering Numeric Arguments
    EDT has a twenty-two line window with the bottom two lines used for the display of numeric arguments, search strings, and EDT commands. This is a fairly efficient arrangement. VTEDIT follows the old convention of not echoing entered numeric arguments. The advantage is no screen update at all is required. The disadvantage is the argument cannot be edited or even be seen on the screen.
12. Entering TECO Commands / Search Arguments
    EDT uses the bottom two lines for entering search strings and commands. VTEDIT utilizes the old convention of clearing the top two lines on the screen to enter the text, and then having to refresh those lines when finished. The routine that does this is not as glorious as the one in the original VTEDIT, but it does get the job done.

Changed Keys

This is a list of redefined EDT keys in VTEDIT.

1. CTRL/A
   The EDT CTRL/Z has been redefined to the old VTEDIT CTRL/A that appends from the open file to the text buffer.
2. CTRL/D
   This key has been redefined in VTEDIT to be a synonym for GOLD-Keypad-2.
3. CTRL/E
   CTRL/E is used to execute a register. After entering CTRL/E, type the one character name of the register to execute. Note that if a numeric argument is used, it will be used for iteration and not passed to the macro.
4. CTRL/K
   This key is a synonym of Keypad-6.
5. CTRL/T
   This EDT CTRL/T key is not defined in VTEDIT.
6. Undefined Control Keys
   EDT version two ignores control keys that are not defined to perform any function. VTEDIT inserts any unknown control key into the buffer.

New keys

This is a list of keys not defined in ED2.

How to count your chickens before they hatch.

Now available for PDP-11 and VAX-11.

Surprises can be expensive. Even good news can cost money if your company is not prepared for it.

With financial modeling you can avoid surprises and plan calmly for whatever the future has in store.

FINAR is the latest financial analysis and reporting system. It will help you plan:

- Budgets
- Cash flow
- Capital investment
- Project evaluation
- Forecasts
- Consolidation

All you need is a DEC PDP-11 with RSTS or a VAX-11, and FINAR—the Financial Analysis and Reporting Language.

If you'd like to know how to count your chickens before they hatch, call or write:

Finar Systems Limited
6000 E. Evans, Suite 2-300
Denver, CO 80222 • **(303) 758-7561**

New York • **(212) 222-2784**
Chicago • **(312) 876-1081**
Houston • **(713) 960-0848**
San Francisco • **(415) 956-1178**
Toronto • **(416) 245-8473**

FINAR

CIRCLE 51 ON READER CARD

Request and You Shall Receive Answers. On Line.

Software for Manufacturers

MS-11 Manufacturing

□ Inventory Control
□ Purchasing
□ Bills of Material
□ Shop Routing
□ Work Order Status
□ Manufacturing Cost
□ Material Req. Planning
□ Capacity Planning

FS-11 Financial

□ Order Entry/Invoicing
□ Sales Analysis
□ Accounts Receivable
□ Accounts Payable
□ General Ledger
□ Fixed Assets

System Highlights

● On Line, Interactive
● Fully Integrated
● Easy to Install
● DEC PDP-11 and VAX

Company Highlights

● 130+ Installations
● Regional Branch Offices
● Installation Support
● User Group

Call or write today

nca corporation

388 Oakmead Pkwy.
Sunnyvale, Ca. 94086
(408) 245-7990
TWX#9103399258

CIRCLE 52 ON READER CARD

1. CTRL/Y
   This key operates very much like the old VTEDIT CTRL/Y yank.
2. GOLD CTRL/Z
   The GOLD-CTRL/Z key will close the file being edited and exit TECO to the monitor. This used to be <esc> CTRL/Z in the old VTEDIT.
3. GOLD Q
   Use this combination to abort the current edit and exit TECO. This is equivalent to the old VTEDIT <esc> — CTRL/Z.
4. CTRL \
   Type CTRL \ (Control-backslash) and VTEDIT will insert the ASCII value of next character typed. This operates the same as the old VTEDIT.
5. GOLD ARROW ±
   Using this combination will move upward a screen. This operates like the old VTEDIT.
6. GOLD ARROW v
   Using this combination will move downward a screen (a screen is 24 lines). This operates like the old version of VTEDIT.

I have provided the new VTEDIT macro in a "run"able source program form. I realize that everybody is going to want to stick in some extra feature. So, it is left as an exercise to the student to squish VTEDIT down to a smaller size. (The source program runs in 10K, and can be cut down to at least 7K, and perhaps 6.) Infinity Software Corporation makes no guarantee for the performance of VTEDIT.TEC, nor takes any responsibility for the use of VTEDIT.TEC. However, inquiries, suggestions, bugs, or whatever can be sent to:

David Spencer
Infinity Software Corporation
2210 Wilshire Blvd Suite 801
Santa Monica, California 90403
(213) 820-2702 ♥

AUTHORS!!! Send your articles of interest to the RSTS community to the ***RSTS Professional*** on mag tape in either RNO, PIP or WORD-11 format. Eighty percent of this issue was transmitted via tele-communications from author's mag tapes to phototypesetting equipment and was not retyped.

THE RSTS/E SYSTEM MANAGER

. . . continued from page 85

In my opinion it's one thing to unintentionally put out a product which has a minor problem in its operation and, perhaps, choose not to fix it. It is quite another thing to sell a product **specifically to run under RSTS/E** (see the label on the distribution tape . . . it **says** RSTS/E) and either leave sensitive files open to examination and modification or, far worse, **compromise the security of the system and then**, after being advised of the problem, **choose not to do anything about it!**

Come on, DEC! If you truly didn't intend for this to run on a secure system, yet you sell it for what is highly touted as a secure, multi-user system (see your publication "RSTS/E Technical Summary", pages 6-1 and 6-2), **it should be so stated, with appropriate details, so this can be determined by the potential buyer prior to purchase!**

Let me close for this month with a warning. I've been working for several weeks with S.E. Regional and Corporate DEC engineers on a problem with my TU77 tape drives connnected to PDP-11/70s.

Although we do not yet know why, we **have** determined that, under as yet unknown circumstances which are greatly increased when the capstan ramp-up speed **is set to factory specs, the drive is writing corrupt data onto the tapes and is *NOT* flagging an error condition!**

This means that if you do tape operations which give you the option of a "verify" pass, **use it** (you should, anyway) and be prepared to distrust any tapes written without a "verify" pass!

More information next month.

See you then. ♥

SPECIAL OFFER

MACHINE READABLE
800 BPI DOS LABEL
OF MAJOR PROGRAMS
IN THIS ISSUE

Send $50 to:

M SYSTEMS, INC.
Box 361
Ft. Washington, PA 19034-0361

CLASSIFIED

Send Classified Ads to: RSTS Classified, P.O. Box 361, Fort 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.]

POSITIONS AVAILABLE

BASIC-PLUS PROGRAMMER WANTED. Call 405-772-6611 X4108 or 4109

Rapidly expanding staff needs RSTS/E expertise. Send resume to: DEC Project Manager, 2170 Piedmont Rd., Atlanta, GA 30324.

RSTS/E PROFESSIONALS

Positions now open in exciting WALL STREET projects.

-TOP MONEY-
-SUBCONTRACTORS WELCOMED-

Send resume or call:

LINK SOFTWARE INC.
P.O. Box 208
Wall Street Station
New York, 10268
(212) 426-8921

BUY/SELL

BEST ELECTRONIC MAIL SYSTEM available for RSTS. Money back guarantee. 714-980-0046

Wanted: OLD PDP-11 System Manuals. Write: Landino 6 Soundridge Rd., Shelton, CT 06484

NEEDED — RSTS/E and IAS Machine Time in Houston Area. Call Gary at (713) 772-1927

ATAC are Australian. We buy 240 V.50 HZ. Equipment. TEL.(02)436-2477 TELEX AA23862

Buy and Sell DEC Equipment, Midwest Systems, Inc., (612) 894-4020.

EXIDY SORCERER-I 16K $795. M. MORELL, 6820 Whiteshell, Las Vegas, NV 89108

BASIC+2 TAPE FOR SALE. LIFEBOAT ASSOCIATES. CALL EILEEN (212) 860-0300

CONSULTANTS

ACC&SS for contract programming and consulting work. 8112 Pinewood Dr., Juneau, AK 99801

Contract programmer. $38/HOUR - BP2, RMS11, Heavy Applications Experience, Financial, Medical, Dental...Will Travel, Don Schwirtlich (512)852-8824

RSTS/E CONSULTANT - Sysgens, Modifications, Systems Programs. Call Russell Dietz, (516) 744-7242 or 929-8500 Ext 238

GENERAL

Infinity Software Corporation happily announces the addition of TECO FREAKO. David Spencer, (213) 820-2702.

To Betsy, Dick, Gerry, Greg, Kathy, Joe, Judy, Pat, Rick, Rod & Tom. Thanks for the educational intellectual and emotional impact in Winona, 1981. Hope to see you all in Bermuda, 1982. (Names sorted using Macro 11 subroutine.) Les.

Sort RSTS/E files in a flash with CSORT. $200 on magtape. Kent Harris, 6687 Green Way, Apt. 4, Greendale, WI 53129.

RSTS/E TIME AVAILABLE — FLORIDA FCI, 99 NW 183rd Street, N. Miami, FL (305) 652-1710.

Network Computer Services, Sydney, Australia, distributor for Oregon Software DPD Dymanion Research.

NCR 315, 3 CRAMS, Paper Tape, 1200 LPM Printer. Works! (513) 767-7717

RSTS/MICRO XSUPORT 6809 Z-80, Ring UK 01-643-5713.

Digital Tax Systems, Inc., 20 Court St., Hackensack, N.J. 07601, TEL. (201)488-6699

Midcom Corp., Serving the Dec User Community. Call for info. (714) 998-6041

Management Information Systems: Authorized POISE DMS Distributors, DEC Commercial OEM. CALL (304) 472-9520

Paco Systems (713) 977-3283, General Accounting, Oil and Gas, Pension Administration.

RSTS Consulting, Contract Programming, Structured BASIC-PLUS Instruction. COSAP Consulting and Seminars. L.E. Walker; 310 N. Blackhawk Ave; Madison, WI 53705 (608) 231-3712.

LEX-11 Word Processing for PDP-11, VAX. All DEC operating Systems. Fast, Flexible, 'Friendly', Comprehensive. EEC Systems (617) 358-7781/2

OVER 100 man years of problem solving using RSTS. For assistance call CIBAR, Colorado Springs (303) 574-4050.

MORE MACRO MAN from the Washington Connection.

RSTS RESCUE SQUAD

We salvage all kinds of disasters:

- unreadable disks
- ruined UFDs and MFDs repaired
- immediate response
- telephone DIAL-UP
- on-site
- software tools
- custom recovery
- 90% success to date
- more than 1 GB rescued to date

Brought to you by
On Track Systems, Inc.
and a well known (and read)
Disk Directory expert.
CALL 24 HOURS
215—542-7113

SAVE YOUR quarters, games for RSTS/E are here. Interactive, real-time games for VT52 and VT100 terminals. Some of the games available:

BLKADE: As many as eight players try to force each other to collide with their growing tails on the screen.

STRWRS: You have five minutes to destroy the Death Star with your single X-wing fighter.

SUBS: Two players manuever around islands in an attempt to sink the other player's submarine.

Games come on 9-track magtape with their own high-quality user manuals. Order the first game for $39.95, $29.95 for each additional game. Individual manuals available for $9.95 each.

Send your check, or write or call:

INFINITY SOFTWARE CORPORATION
2210 Wilshire Blvd
Suite 801
Santa Monica, California 90403
(213) 820-2702

COSAP™

Conversationally Oriented Statistical Analysis Package, Version 2B

Coherent, versatile package written in BASIC-PLUS for RSTS/E featuring low overhead per user, a powerful select mode, and flexible commands to edit, transform, and analyze data.

Software product announcements for COSAP, the Lawrence Plotting Package for BASIC-PLUS, and Lawrence RUNOFF are available from:

Computer Center Publications
Lawrence University
Box 599
Appleton, WI 54912

REPAIRS

REPAIR SERVICES

Centronics Printers
All Series - All Models
DEC Computers & Terminals
All Makes - All Models

AAA COMPUTER SERVICE COMPANY

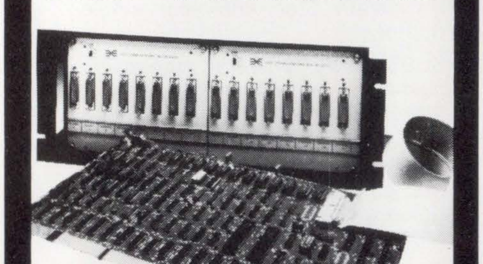
509 Springarden Mall
Philadelphia, PA 19123
215-382-3457

DEC

WHY PAY MORE???

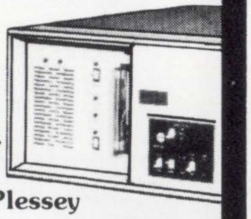
$AVE WITH DEC COMPATIBLE PRODUCTS

8-64 LINE DH11 ON ONE HEX CARD
½ PRICE - TWICE THE PERFORMANCE



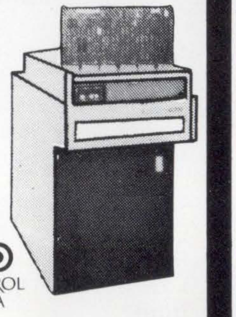
EMULEX

LSI-11/23
MSV11, DLV11
28 MB Winch
10 MB Tape



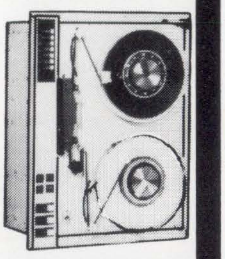
Plessey

DISK SUBSYSTEMS
10-675 MB
LSI-11,
PDP-11, VAX



CONTROL DATA

MAG TAPE SUBSYSTEMS
45-125 IPS
LSI-11,
PDP-11, VAX



KENNEDY
Subsidiary Magnetics & Electronics Inc

Also Available: DL11, DD11, MS11, DZ11, MA20

CCG CALIFORNIA COMPUTER GROUP

Toll Free
800-854-7488

In California
714-966-1661

TELEX: 183519 CCG CSMA

3303 Harbor Blvd., #K-11, Costa Mesa, CA 92626

CIRCLE 53 ON READER CARD

DEC BEST VALUES

PRE-OWNED DEC EQUIPMENT
BUYING AND SELLING
SYSTEMS • CPU's • PERIPHERALS • TERMINALS
OPTIONS • MEMORY • COMPATIBLES

CALL DICK BAKER (305) 979-2844

dataware incorporated
Carico Center
2845 NW 62nd Street
Ft. Lauderdale, Florida 33309
Telephone (305) 979-2844

GROSSMAN G GRAPHICS
215537-0782

Lifeboat Associates

Pre-Eminent Publisher
Of Computer Software
Introduces
AMCOR Software for RSTS/E
Including:
◇ General Ledger
◇ Accounts Payable
◇ Accounts Receivable
◇ Business Control Systems
and
XENIX
Version 7 Unix Operating System Configured for 11/23 and Unibus models of PDP-11 Computers
Contact:
Lifeboat Associates
Minicomputer Systems Division
1651 Third Avenue, Dept. M
New York, New York 10028
Tel:(212)860-0300 TWX:710-581-2524
Telex:640693

UNIX is a trademark of Bell Laboratories
XENIX is a trademark of Microsoft
PDP-11 and Unibus are trademarks of Digital Equipment Corporation

DELAWARE VALLEY
TIMESHARING

Eastern Pennsylvania
Philadelphia
South Jersey

DEC 11/70 RSTS/E

24 hours - 7 days
Most commerical packages
Custom work, Raw time

NATIONWIDE DATA DIALOG
70 James Way
Southampton, Pa. 18966
215-364-2800

The *FAMOUS RSTS PROFESSIONAL TEE-SHIRT* is now for sale!

Send size desired and $6.95 for each shirt to:
RSTS TEE-SHIRT, P.O. Box 361, Ft. Washington, PA 19034-0361.
Shirts available in adults sizes only:
Small — Medium — Large — X-Large

FOR SALE:

2 RP04
1 RS04
2 RH70
3 DH11/AE in Expander box in H960

11/40 System including
CPU w/RSTS options
ABLE CACHE & EMULOADER
128KB CDC MOS
BACKPLANE (ABLE)
CDC 45 IPS TAPE
W/EMULEX CTLR
ABLE DH/DM's

DAVE MALLERY
(215) 364-2800

DEC

SYSTEMS
&
COMPONENTS

C.D. SMITH & ASSOCIATES, INC.
12605 E. Freeway,
Suite 318
Houston, TX 77015
(713) 468-2384

CIRCLE 54 ON READER CARD

NEWS RELEASES

Occasionally we are requested to print news that may be of interest to the RSTS community. We are happy to offer this feature to our readers. We reserve the right to print only as time and space permit. We cannot return photos or manuscripts. Send news releases to: RSTS NEWS RELEASE, P.O. Box 361, Fort Washington, PA 19034-0361.

September, 1981
VT100 USERS GET SESSION PLAYBACK SOFTWARE PACKAGE

Bedford, Mass. — Clyde Digital Systems has just introduced a session playback package for Digital Equipment Corporation's PDP11s running under the RSTS/E operating system.

This new package, called IMAGE, is able to run against session documenting log files produced by the recently announced program DOC. The VT100 screen terminal exchanges a number of control codes with the processor when it is used in screen formatting applications. The combination of DOC, to create a log file of a complete session of any given interactive application, together with IMAGE to replay the session back to the screen, addresses an important problem. Specifically, that of getting dynamic user documentation of screen formatting applications to playback accurately whe needed for subsequent reference and training.

With this new capability, the dynamic application run example may be played back one frame at a time. The various unwanted control characters that were generated at actual run time are eliminated. The result is a simple play back of the run example that looks exactly as it should with frame by frame inspection at the trainee's leisure.

In addition, an accurate mapping of the dynamic session may be put out as a printed document. The results are the same as those achieved by the copy option on a VT100 with a printing terminal connected to its auxiliary printer port. The difference is that a single copy of the IMAGE package is only $295.00 and may be used in association with any number of VT100s connected to the system.

Installation of both DOC and IMAGE consists of reading these programs onto the system. A typical user learns how to run them in about 20 minutes. They are available for immediate delivery. The standard distribution media is 800 or 1600 BPI, 9-track magnetic tape.

Product information may be obtained from: Janet (617) 275-6642, Clyde Digital Systems, P.O. Box 348, Bedford, MA 01730.

———————

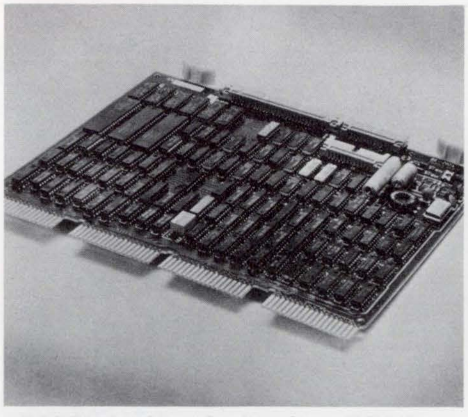
July 3, 1981
NEW SINGLE BOARD SMD I/O COMPATIBLE DISC CONTROLLER MIXES OR MATCHES DRIVES, for Easy LSI-11 Data Base Expansion Without any Component Changes, and Offers Two Methods of Preventing Media-Flaw Data Errors.

Garden Grove, CA — A new single quad size SMD I/O compatible Disc Controller, featuring mix or match interface of one or two 8″ or 14″ Winchester/ SMD pack/ or CMD cartridge hard disc drives with 8-300 Mbyte storage to LSI-11, 11/2, 11/23 computers, is now available from DILOG (Distributed Logic Corp.), Garden Grove, California for easy up-grade or expansion to a larger capacity drive during or after installation.

Designated the Model 202A, which is based on a field proven proprietary DILOG design, the controller employs a universal firmware set that doesn't retain drive parameters in on-board components, permitting mixing or matching drives without controller or component changes. It also offers both bad-sector mapping when formatting the disc and automatic media-flaw compensation as an alternate to ECC used on other DILOG controllers.

The uP based controller, which uses up to 60% less power than similar controllers, plugs into one slot of any LSI-11 based quad backplane and is compatible with DEC RP02/RP03 software drivers (switch selectable) in RT-11 and RSX-11 operating systems.

Other standard features include * automatic power-down protection * on-board bootstrap loader for RP02/RP03 and TM-11 support, with jumper selectable bootstrap address * automatic self-test mode incorporaing built-in microdiagnostics and a data protect feature with status indicator . . . similar to other DILOG family products for ANSI/SA1000/SA4000/ST506 interface as-well-as ¼″ 3M cartridge/½″ NRZI-PE and floppy controllers for LSI-11 and PDP-11 CPUs.



DILOG (Distributed Logic Corp.) introduces a SMD I/O compatible hard disc controller. For LSI-11 computers, which permits expanding data base by mixing or matching 8″ or 14″ Winchester/SMD PACK/CMD cartridge drives having 8-300 Mbytes storage ... without component changes.

Price: $2,775.00 ea. Delivery: 30-45 days A.R.O. Direct inquiries to: Mr. Dennis Edwards-,Natl. Sales Mgr., DILOG, 12800 Garden Grove Blvd., Garden Grove, CA 92643, phone: (714) 534-8950. ♥

LIST OF ADVERTISERS

Aardvark Software Inc.	p. 77
Able Computer	pp. 61, I.B. Cover
Amcor	p. 2
American Used Computer	p. 31
C.D. Smith & Associates, Inc.	p. 94
California Computer Group	p. 94
Clyde Digital Systems	p. 10
ComDesign, Inc.	p. 87
Computer & Terminal Exchange	pp. 18-19
Data Node	pp. 41, 43
Data Processing Design, Inc.	B. Cover
Data Terminals Corp.	p. 13
Dataram Corp.	p. 96
Dataware Inc.	p. 88
Digital Information Systems Corp.	p. 55
Distributed Logic Corp.	p. 23
EEC Systems	p. 81
Emulex Corp.	p. 5
Enterprise Technology Corp.	p. 7, 88
Evans Griffiths & Hart, Inc.	pp. 47, 57
Finar	p. 91
Florida Computer, Inc.	p. 71
George Foldvary & Associates, Inc.	p. 50
Hamilton Rentals	p. 15
Manus Services Corp.	p. 69
McHugh, Freeman & Associates, Inc.	p. 27
Midwest Systems, Inc.	p. 33
N.S.D. Sales	p. 62
NCA Corp.	p. 92
Nassau Systems	p. 45
National Public Radio	p. 33
National Semiconductor Corp.	p. 89
Nationwide Data Dialog	pp. 12, 16
North County Computer Services, Inc.	pp. 48-49, 67
On-Track Systems, Inc.	pp. 16, 17
Plycom Services, Inc.	p. 79
Raxco Inc.	p. 37
Real Time Products	p. 9
Ross Systems, Inc.	I.F. Cover
Saturn Systems, Inc.	p. 44
Scientific Placement, Inc.	p. 65
Sevco, Inc.	p. 85
Software Results Corp.	p. 82
Software Techniques, Inc.	p. 61
Southern Systems, Inc.	p. 1
System Industries	pp. 34-35
System Associates, Inc.	p. 29
T.F. Hudgins & Associates, Inc.	p. 14
Telecom Computer Systems, Inc.	p. 88
TransNet Corp.	p. 45
Tymshare Inc.	p. 63
Unitronix Corp.	p. 44
Wireworks Corp.	p. 39

♥

LETTERS to the RSTS Pro . . .

. . . continued from page 6

Dear Sirs,

Once again I am entering your fiendish photo contest. The new photo, entiled "Garbage Out — Garbage In" is obviously a picture of the latest in forms handling and disposal equipment. The garbage can neatly stacks the forms as they come pouring out of the printer, so they can be hauled away with very little effort (nobody reads that computer stuff anyway!). After what I went through trying to track down the elusive TECO, I think I deserve a tee-shirt for this one.

Sincerely, Mark Anacker
Information Systems Analyst
General Telephone of the Northwest

Hello, Mark. Your answer is close but not right, however, you are right in one respect — you ***do*** *deserve a tee-shirt — for persistence! (And, we want you to make a good impression at General Telephone — what's your size?)*

I would like to congratulate you on another fine issue. Each and every one is stuffed with swell stuff. The magazine has excellent quality, keep up the good work (and check all program listings you publish, even if Steve Davis *did* write it).

I found this issue's contest quite interesting. It looks to me like a solution to a problem I have had. With all the exhaust fans and air conditioners blowing the output printer paper just won't stack (this can be a real pain when printing large reports). The solution is simple, put up a wind screen — like a trash can.

Since I missed out on "TECO", I hope this wins me a tee-shirt (make it XL).

Mike Gacsaly
Coca-Cola Bottling of Los Angeles

Well now, Mike, that's a pretty good solution to the problem you mention, but it doesn't apply here. We'll put an XL aside for you and maybe you'll get to claim it next time.

Dear Dave,

I just received your latest "What is it" in the mail today and can appreciate what the system manager at that site is going through. We have an LP05 and have had no end of trouble with static and paper stacking. I may take your suggestion and try a plastic bag since our stacker could only work better.

I enjoy your magazine and am especially enjoying Mike Mayfield's articles on Monitor Internals. One thing that I would like to see is an article that explains the BASIC+ PUSH/POP code and gives a list of the codes and what they do.

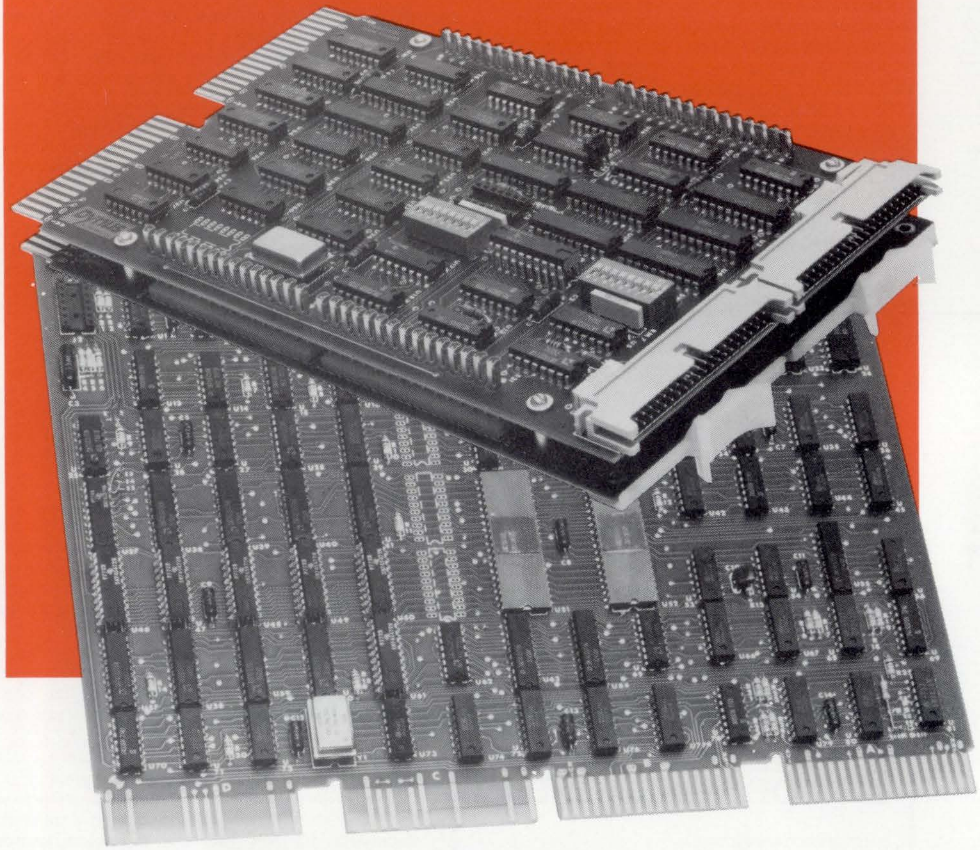
I will be sending in an article on Program Extraction and ways to speed your program generation. If I am correct on the photo, please either send me a free issue or an XL tee-shirt. (I have already received a tee-shirt for Bounty Hunting [New readers, see insert cards.] and don't really know what I could do with another one.

Stephen Munyan, Holden, Mass.

Stephen, we look forward to receiving your article. Also ***we'd*** *like an article on BASIC Internals, too. Readers!*

Your answer is incorrect so we can't send a prize, but maybe we helped solve a problem for you. By the way, if you do win another tee-shirt you could always send it to Mike, it would fit perfectly. ♥

DEC®-COMPATIBLE PERIPHERAL CONTROLLERS



Now, from the company that delivers the industry's widest range of DEC-compatible memory products, a family of peripheral controllers that's second to none. From comparatively simple cartridge disk controllers to complex 300MB storage module drive (SMD) controllers.

An impressive array of state-of-the-art controllers, all built around high-speed bipolar microprocessors. All software compatible with the host LSI-11® or PDP®-11 minicomputer...and all available now.

And Dataram's controllers are designed to save you money, and a lot more. Like space — our controllers typically occupy half the space required for the comparable controller from DEC. Doing it with a level of performance that makes any member of this family worth looking at.

Look at the chart of our current family of peripheral controllers, growing every day. If you don't see the controller you need, we're probably working on it right now. Call us and discuss your requirements.

DEC, LSI-11 and PDP are registered trademarks of Digital Equipment Corporation.

DATARAM CORPORATION

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

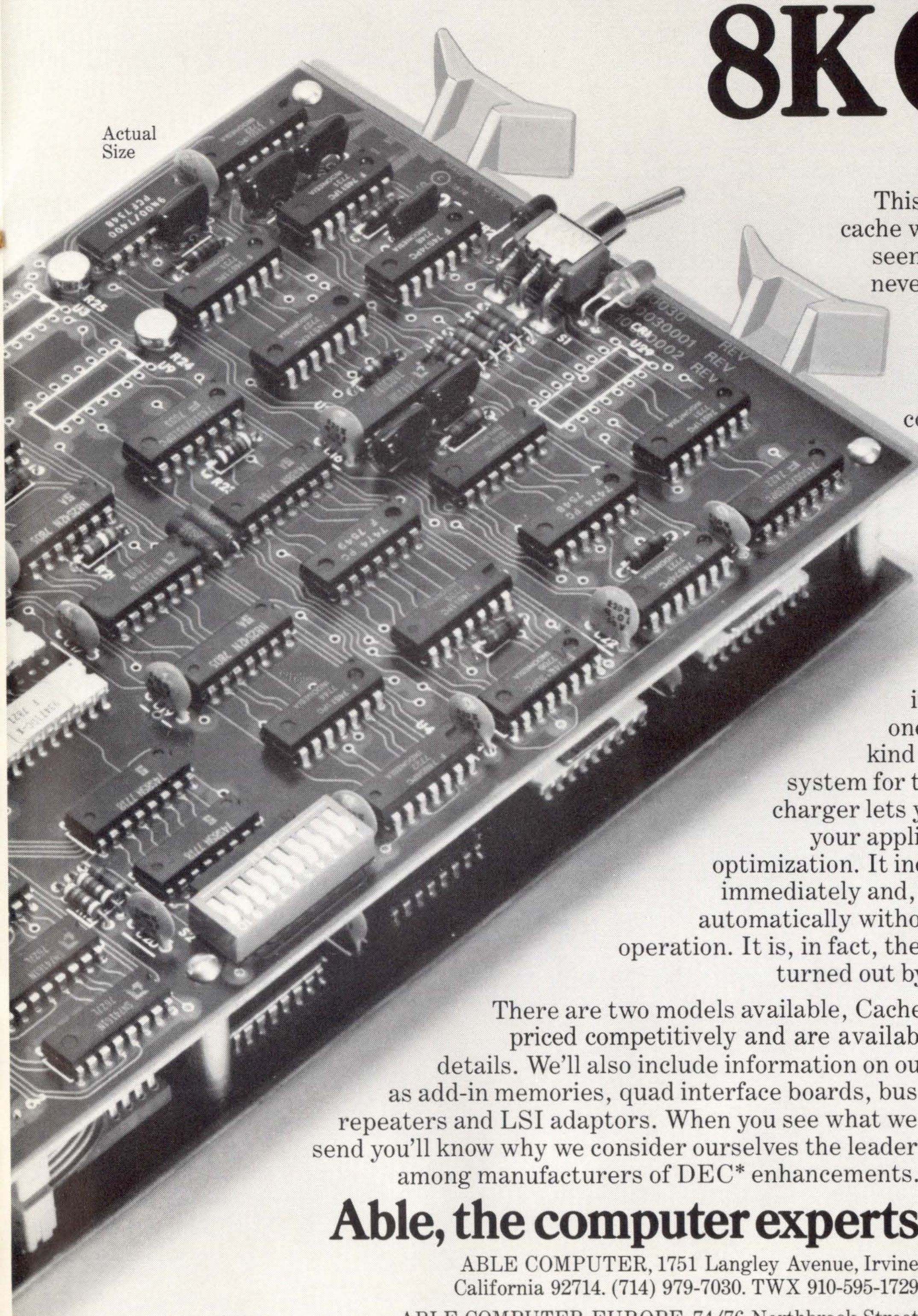
DATARAM CONTROLLER	Magnetic Tape Controller			Cartridge Disk Controller		SMD Controller					
	T03	T04	T34	C03	C33	S03/A	S03/B	S03/C	S33/A	S33/B	S33/C
MINI	LSI-11	LSI-11	PDP-11	LSI-11	PDP-11	LSI-11	LSI-11	LSI-11	PDP-11	PDP-11	PDP-11
COMPATIBILITY	TM11/TU10			RK05		RM02	RK07	RP06	RM02	RK07	RP06

Canada: Ahearn & Soper Ltd., 416-245-4848 • **Finland:** Systek OY, (80) 73 72 33 • **France:** YREL, (03) 956 81 42 • **Hungary/Poland/Rumania:** Unitronex Corporation, WARSAW 39 6218 • **Italy:** ESE s.r.l., 02/607 3626 • **Netherlands:** Technitron b.v., (020) 45 87 55 • **Sweden:** M. Stenhardt AB, (08) 739 00 50 • **Switzerland:** ADCOMP AG, 01/730 48 48 • **United Kingdom:** Sintrom Ellinor Ltd., (0734) 85464 • **West Germany:** O.E.M.-Elektronik GmbH, 07 11-79 80 47 • **Yugoslavia:** Institut "Jozef Stefan", 263-261 • **Australia/New Zealand:** Anderson Digital Equipment, (03) 544-3444 • **India:** Infosystems Private Limited, 79281 • **Israel:** Minix Computers & Systems Ltd., 03-298783 • **Japan:** Matsushita Electric Trading Co., Ltd., 06 (282) 5111 • **Taiwan:** Rabbit Associates, Ltd., 7219573-5 • **Hong Kong:** Automated Systems (HK) Ltd., 5-630256-9 • **Malaysia:** Automated Systems (M) Sdn Bhd., 773777 • **Indonesia:** P. T. Daya ASL, 584306 • **Singapore:** Automated Systems (PTE) Ltd., 2354133

CIRCLE 55 ON READER CARD

Able offers a magnificent memory for the full PDP-11/34 series.

The Incredible 8K Cache

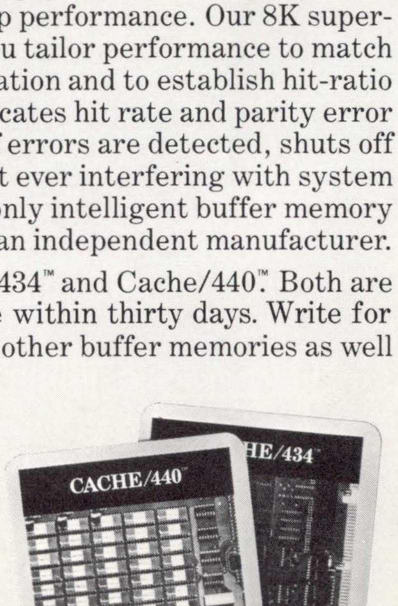


Actual Size

This is the ultimate version, the 8K cache with everything. You have never seen one like it before, and you may never see another. Designed for the 11/34, 11/35 and 11/40, it has byte and address parity, upper/lower limit switches, on-line/off-line manual switch control, activity indicator lights, 8K bytes of memory (4K words) providing a capacity four times that of competitive units and an interconnect board design which consumes no extra space in the computer.

That's why there is always room in your computer for this incredible cache. You should put one there, especially if you are the kind of guy who likes to optimize his system for top performance. Our 8K supercharger lets you tailor performance to match your application and to establish hit-ratio optimization. It indicates hit rate and parity error immediately and, if errors are detected, shuts off automatically without ever interfering with system operation. It is, in fact, the only intelligent buffer memory turned out by an independent manufacturer.

There are two models available, Cache/434™ and Cache/440™. Both are priced competitively and are available within thirty days. Write for details. We'll also include information on our other buffer memories as well as add-in memories, quad interface boards, bus repeaters and LSI adaptors. When you see what we send you'll know why we consider ourselves the leader among manufacturers of DEC* enhancements.



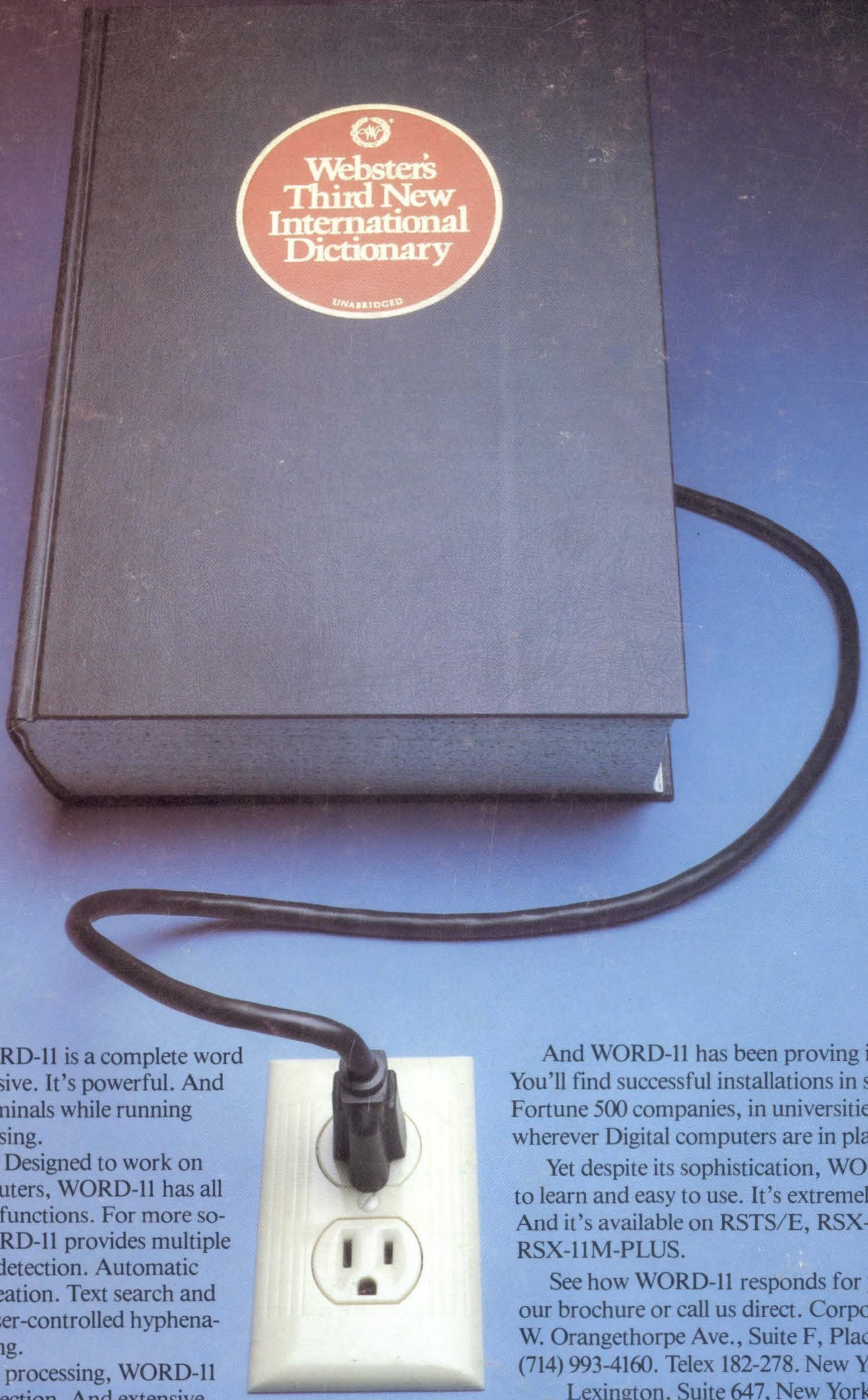
Able, the computer experts

ABLE COMPUTER, 1751 Langley Avenue, Irvine,
California 92714. (714) 979-7030. TWX 910-595-1729.

ABLE COMPUTER-EUROPE, 74/76 Northbrook Street,
Newbury, Berkshire, England RG13 1AE. (0635) 32125.
TELEX 848507 HJULPHG.

CIRCLE 56 ON READER CARD

Responsive Word Processing Take Our Word For It.



WORD-11. WORD-11 is a complete word processing system. It's responsive. It's powerful. And it's sharable on up to fifty terminals while running concurrently with data processing.

WORD-11 is talented, too. Designed to work on Digital's family of mini-computers, WORD-11 has all the standard word processing functions. For more sophisticated requirements, WORD-11 provides multiple dictionaries for spelling error detection. Automatic index and table of contents creation. Text search and replace. User defined keys. User-controlled hyphenation. And automatic footnoting.

In addition to standard list processing, WORD-11 offers fast sorting. Flexible selection. And extensive math functions.

And WORD-11 has been proving itself for years. You'll find successful installations in small businesses, Fortune 500 companies, in universities and in banks—wherever Digital computers are in place.

Yet despite its sophistication, WORD-11 is easy to learn and easy to use. It's extremely cost effective. And it's available on RSTS/E, RSX-11M and RSX-11M-PLUS.

See how WORD-11 responds for you. Write for our brochure or call us direct. Corporate Office: 181 W. Orangethorpe Ave., Suite F, Placentia, CA 92670, (714) 993-4160. Telex 182-278. New York Office: 420 Lexington, Suite 647, New York, N.Y. 10170, (212) 687-0104.

CIRCLE 59 ON READER CARD

Data Processing Design, Inc.
AUTHORIZED digital COMPUTER DISTRIBUTOR

Overseas Distributors:

Management Information Services PTY. LTD.
Melbourne, Australia

Jenson, LTD.
Bristol, England

Network Computer Services PTY. LTD.
Sydney, Australia

Systime, LTD.
Leeds, England

On-Line Computing PTY. LTD.
Subiaco, W. Australia