RSTS PROFESSIONAL

Volume 3, Number 2

June 1981

10^{00}/issue, 25^{00}/year

INSIDE:

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 __________

*DEC is a TRADEMARK of Digital Equipment Corp

Contents

Coming . . .

- Scott Banks on Disk Directories
- Bob "Macro Man" Meyer on How to Macro
- Technical PDP-11 Internals
- Continued RSTS Internals
- The RSTS Pro Installs Their Computer
- GSYS.BAS
- Disappearing Files
- RSTS Software Hall of Fame
- VAX X.25 Communications
- Modeling the VAX
- EXTEND.BAS
- How to Stretch Your 11/40
- More . . .

The RSTS Professional Magazine, June 1, 1981, Vol. 3, No. 2. Published quarterly. Single copy price \$10.00, \$25.00 per year. Published by M Systems, Inc., 753 Johns Lane, Ambler, Pa. 19002, telephone (215) 542-7008. Send all correspondence and change of address to: Box 361, Ft. Washington, Pa. 19034. Application to mail at second class postage rates pending at Ambler, Pa. 19002.

From the editors...

waiting . . .

Carl Marbach

Version 7.0 was in its time a neat piece of software. It had all kinds of good new things; big (better,faster) FIP, Data caching, QSTATS, lots of new MODES, ANSII magtape routines and lots more. It was a definite improvement over V6C. Sure, it had its problems, but these would be fixed in the next release — wasn't that just a year away?

In the Golden, Olden days, when you purchased a one year support contract for RSTS it guaranteed you at least one new release. We became used to a release cycle of about 1 year. Okay, it slipped once and a while to 18 months, but the clear inference was that we could expect releases at about 1 year intervals. In 1975 we were running V6A. In 1980 we were through 6B, 6C and 7.0 giving us 4 versions in 5 years; pretty close to the cycle we were describing above.

December will mark two (2) years since V7.0 hit the streets and the best DEC GUESS is that the next version is still about 1 more year away! Three years! What's going on? Hardware hasn't exactly been quiet over these 2 years; RM05, 11/24, immediate delivery (!) on LS120's and VT100's, MOS memories and more.

Maybe V7.0 is so solid it doesn't need a new release. But what about the 11/70 small buffer problem, the 11/34 task building problems (it's still task building!), the RM05 support (it costs extra), 11/24's are anyone's guess, Stats that don't work and worse (they give erroneous figures) and more. Why is it that RSX seems to get the new bells and whistles first; DECNET, 2780 support, FORTRAN 4 PLUS.

Of course the commercial marketplace has made RSTS one of the most active operating systems around. There is Word Processing, List processing, Data bases, Queing systems, backup packages, magazines, disk structuring packages, modeling systems, languages and editors — all from sources other than DEC. Just look through this magazine for all the good people working for your money.

Insiders tell me that all this time is being spent figuring out the BEST solution to these problems. That they will be worth waiting for. How do all of you out there feel about paying for three years for support and not seeing a new release? We think that although two women can't make a baby in less than nine months, two RSTS developers could produce a new version of RSTS twice as fast as one. Message: get off INDENT, GIGI and frills; make RSTS work the way it should.

What are you waiting for?

DECUS MIAMI
The Tide Has Turned

Dave Mallery

The big news from Miami is that two years of stone-walling has ended. DEC is talking to us again! One is tempted to attribute the thaw to the effect of tropical breezes and sunny skies on those inhabitants of the frozen north, but I'd rather think that there have been some fundamental changes in policy.

This was obvious right from the start. As soon as the opening salvos of what I had predicted would be "Buffer Wars" were fired, the development folks informed us that they were promising relief in the next release and would tell us more at a later session. The next morning, at a session entitled "Building a RSTS Monitor", Andy Riebs from the development team disclosed two approaches that were in the works to provide the relief.

First, a new memory pool would be established to hold WCB's and FCB's. Secondly, selected code segments would be re-worked to utilize "I and D" space—a hardware feature never before used by RSTS. Basically, this presents the developer with another set of memory mapping registers to use for buffer pools and the like. Please be very **clear** that nothing in this article, as well as nothing said at Miami represents a firm committment by DEC. It is imperative that we accept this information in the spirit in which it was given.

There was also some bad news. The next release is more than six months away.

This symposium was highlighted by many excellent user papers. Mike Mayfield from Northwest Digital Software delivered a six hour marathon on Monitor Tables to large, late night audiences.

Mark Diebert from Squibb gave an excellent paper entitled "So Your Disk Is Irrevocably Corrupt" shedding a great deal of light on one of the more ominous init error messages.

Joyce Hayes and Steve Stepanek gave a three session TECO wonderland tour. It's amazing how some things never die. I have attended more funerals for RSTS and TECO than I care to remember. I heard about one site, in Rochester, NY, that uses about forty

. . . continued on page 28

Andy Riebs spelling the small buffer relief.

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.)

LETTERS to the RSTS Pro...

Dear Dave and Carl,

My subscription to your excellent magazine is now being handled through Pauline [Noakes] in the U.K. As an 'original' subscriber, since you gave me my first free copy at DECUS San Diego, 1979, I would like to say that the articles have been getting better and better with each issue (I like the new VAX section — keep it going!) and it's a shame that you only publish quarterly. Many thanks.

Yours sincerely,
Collin Shaw, Berkshire, England

P.S. I also appreciate the MACRO-11 sections!

Thank you, Collin, it's nice to know that you've been with us since Volume 1, Number 1. We made a wise investment and so did you.

—

Dear Carl,

I was pleased to see my article that you published in the May/June 1980 issue of the *RSTS Professional.* You might be interested to know that since it appeared, I have moved to Memphis, Tennessee, and become a software specialist for Digital.

I think very highly of your magazine. The articles are very informative and especially useful for new RSTS users. I strongly recommend the publication to everyone.

Might I suggest a topic for an article sometime? I'd like to see a review of the word processing packages available that run under RSTS. Technical details such as effect on system load, run time system versus multiple copies of a program, etc., would be useful plus an evaluation of user ease, stability of product and the company marketing the product.

Thanks again.

Sincerely,
Susan Blount Duff, Software Specialist

We like your suggestions, Susan. Stay in touch.

—

Dear RSTS Pro,

Early one Saturday, no, late Saturday nite, now early Sunday morning, I found myself reading back issues. The (obvious) motive: Plagiarism.

Nonetheless, I discoverd a picture of a Datamedia DT-80 in Vol. 1, No. 1, but I couldn't find a later reference. Has anyone collected that T-shirt yet?

We have 3 DT-80's in house, and 2 have blown their ROMS, requiring strapping to provide EIA RS-232 signals for our modems. Did yours (or others) do this?

Steve Suttles, Systems Programmer
Interlake, Inc., Chicago, Ill.

Steve, RSTS MAN says, "Watch static electricity as these terminals seem very sensitive. Non-volatile RAM failures are (unfortunately) common to many of these."

As for the photo contest in our first issue, look no further - there is no later reference. We thought we had everyone "stumped" [in reality, however, most readers probably wanted to see if there would be a second issue]. I suppose you want to be awarded a famous RSTS Professional T-shirt for your answer, well . . . at this late date we'd have to hold a staff conference to decide. Rest assured, we'll take the matter under consideration just as soon as we get the staff to sit still for a few minutes.

—

↑F TTOPNF ERRATA

Dear RSTS Professional:

This is a letter to say thank you, and to point out a bug in the method used to translate source to the printed page.

In the March, 1981 issue you published "Control F, Open Files Support". Which in one swell foop restored my faith in the entire magazine publishing industry. Please convey my thanks to Steven P. Davis for allowing you to publish such a useful patch.

However, the caret characters (∧) were dropped from the two files that made up the article.

TTOPNF.MAC
.
.
TTSYSF:: (etc)
BIC ∧C<63. *2>,R3
.
OUTPNT: (etc)
.
.
BIC ∧C<177>,R2
TTDVR. TEC
EBTTDVR. MAC$$@ ∧A/Found "TTDVR. MAC" (etc)

There seemed to be a late name change. The filenames TTOPNF and TTSYSF seem to be interchangeable so I opted to use what seemed to be the more consistant name of TTSYSF.

Also decoding the TECO source usage of $ or character and $ for <ESCAPE> got a little bit silly. It might have been a good idea to have expanded the occurences of <ESCAPE> in the file to <ESC> or something as distinctive. For your information a copy of the file expanded in this fashion is enclosed. I also took the liberty of adding a carriage return or two.

Paul Ralston, Engineer
Data Node, Inc., Sunnyvale, CA

!!!!!LATE FLASH!!!!!

OOPS: The '#' was also dropped. This is much more serious as MACRO complains not.
MORAL: DO NOT RUNOFF MACRO SOURCE.

```
EBTTDVR MAC<Esc><Esc>
@^A/Found "TTDVR MAC" - working/
Y<Esc>S/AWR<Esc>FR/AWR/SPD<Esc><Esc>

NORG     TTSYST<Esc>LI
         ORG      TTOPNF   ;OPEN FILES CODE

<Esc><Esc>
NTI$CRR  <Esc><Esc>-STI$CTT<Esc>
LI; TI$CFF - F HANDLER
<Esc><Esc>STI$CTT: <Esc><Esc>
LITI$CFF:                  ;;HANDLE CONTROL/F (OPEN FILES)
<Esc><Esc>
SBNE     40$<Esc>S:;<Esc>K
INOPE, IT IS CONTROL/T OR CONTROL/F
<Esc><Esc>
S40$ <Esc>I CMP R2,#'F-100        ;;CONTROL/F MAYBE?
         BEQ      60$             ;;YEP, HANDLE IT
<Esc><Esc>
LI50$ <Esc><Esc>
3LI
60$      CALL     MAPPED,R5,TTOPNF ;;GO TRY FOR OPEN FILES
         BR       50$             ;; DO SAME AS CONTROL/T
<Esc><Esc>
STTSYST <Esc><Esc>
3LI      TMPORG   TTOPNF
TTOPNF:  SEC                      ;; SAY ILLEGAL UNLESS OVERLAID
         RETURN                   ;; AND EXIT
         UNORG

<Esc><Esc>
N.WORD   'G-100<Esc>
OLI       .WORD   'F-100          ;CONTROL/F (OPEN FILES)
<Esc><Esc>
N.WORD   TI$CCG<Esc><Esc>
OLI       .WORD   TI$CFF          ;CONTROL/F (OPEN FILES)
<Esc><Esc>
EX<Esc><Esc>
<Esc><Esc>
```

Our readers keep us on our toes! A t-shirt is going to Paul for pointing out this error to all of us.

Gentlemen:

May I make an unsolicited recommendation regarding the publishing of source code for programs, subroutines and specifically the monitor code listed for ctrl/F? Your intent is clearly to publish routines which can be used by entering the code listed in the article and following the installation procedure. To wit:

1) Have someone attempt to implement the program from the article.

2) Publish ASSEMBLY listings, not raw macro code or something with a checksum capability so that the reader can easily observe if he/she has entered the listing correctly.

3) Use a printer which prints all ASCII characters. The ∧F article has published without benefit of a printer which had the "∧" and "#" characters. These characters appeared as a CHR$(32%).

Had someone attempted to implement the ctrl/F article he would have quickly found the rather conflicted naming of the module and its entry point as "TTSYSF" and "TTOPNF" in sundry places throughout the article.

The PAT.TSK program will compute checksums for an OBJ module and I presume that PAT.SAV does also. Some reader should also be able to provide a good checksum process for Basic Plus 2, Basic +, or TECO code. This would be invaluable in determining if the code was entered properly by the installe.

Many thanks for interesting and informative articles. With a little more effort you can save the readers many frustrating hours.

Gary ap Kohls, Programming Manager
Star Plan Data Processing
Milwaukee, WI

P.S. My monitor is patched for the moment to treat a ∧F as a ∧T as without this patch RSTS reboots to "OPTION: " when the first ∧F is typed!

See corrected version on page 35.

One more last minute answer to our December 1980 "Best Caption" contest.

64 jobs? 128 keyboards? 1000 small buffers?!!

Dave Kachelmyer
North County Computer Services, Inc.
Escondido, CA

Pretty good, Dave. However, the caption that tickled Dave Mallery the most was:

"Small Buffers?? — Oh, I've got 2 of those."

Our winner, therefore, is Kim Branch, Daniel International Corp. We're sending Kim a small (but valuable) RSTS Professional Tee-shirt.

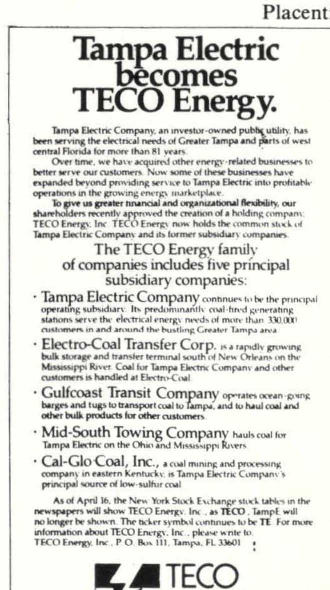
—

How TECO? Why TECO! Who TECO?

Dear Dave:

I thought you might find the attached advertisement from today's Los Angeles Times, amusing.

Sincerely,
Richard A. Marino, Vice President
Data Processing Design, Inc.
Placentia, CA


Tampa Electric becomes TECO Energy.

Tampa Electric Company, an investor-owned public utility, has been serving the electrical needs of Greater Tampa and parts of west central Florida for more than 81 years.

Over time, we have acquired other energy-related businesses to better serve our customers. Now some of these businesses have expanded beyond providing service to Tampa Electric into profitable operations in the growing energy marketplace.

To give us greater financial and organizational flexibility, our shareholders recently approved the creation of a holding company: TECO Energy, Inc. TECO Energy now holds the common stock of Tampa Electric Company and its former subsidiary companies.

The TECO Energy family of companies includes five principal subsidiary companies:

- Tampa Electric Company continues to be the principal operating subsidiary. Its predominantly coal-fired generating stations serve the electrical energy needs of more than 330,000 customers in and around the bustling Greater Tampa area.
- Electro-Coal Transfer Corp. is a rapidly growing bulk storage and transfer terminal south of New Orleans on the Mississippi River. Coal for Tampa Electric Company and other customers is handled at Electro-Coal.
- Gulfcoast Transit Company operates ocean-going barges and tugs to transport coal to Tampa, and to haul coal and other bulk products for other customers.
- Mid-South Towing Company hauls coal for Tampa Electric on the Ohio and Mississippi Rivers.
- Cal-Glo Coal, Inc., a coal mining and processing company in eastern Kentucky, is Tampa Electric Company's principal source of low-sulfur coal.

As of April 16, the New York Stock Exchange stock tables in the newspapers will show TECO Energy, Inc., as **TECO**. TampE will no longer be shown. The ticker symbol continues to be **TE**. For more information about TECO Energy, Inc., please write to: TECO Energy, Inc., P.O. Box 111, Tampa, FL 33601

TECO ENERGY

. . . continued on page 35

VAXination for DEC* Users.

System Industries disk storage. For immunity against high prices, slow delivery and limited choice. If your VAX-11/780* is waiting for RPOX/RMOX drives, you stand to lose a lot more than valuable time. You stand to lose on price/performance, too.

We can help. We're System Industries, the world's largest independent supplier of complete disk systems for DEC users.

Add on or trade out, we'll save you big money. We've been saving DEC users big money on large-capacity disk storage for more than 10 years. And we've made it easy.

We can simply add to your present disk storage. Or we can trade your RPOX/RMOX drive(s) for one of our superior disk systems. Whichever way you choose, you can count on 100% software transparency. And substantial dollars saved.

Get a wide choice, including Winchester-SMD combos. Shared data base with multiple CPUs . . . nonstop operation featuring multiple drives . . . one of our many field-proven system configurations is ideal for your VAX disk storage needs.

We even offer Winchester-SMD combos that pack 900 Mbytes into the same floor space you may be using for 300 Mbytes (RM04/05). By trading your RM04/05 for our Winchester-SMD combo, you can net an additional 600 Mbytes. And pay no more than the cost of an RM04/05 from DEC.

Worldwide service and 30-day delivery. We've built our entire business around fast, effective service. Beginning with responsive delivery—30 days ARO—and following with customized service contracts, we provide the support to get your system up fast. And keep it running.

So why wait for DEC. Take the System Industries VAXination today. It's a sure cure for slow delivery and poor price/performance. For details, contact the sales office nearest you. Or use the coupon.

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

IMMUNIZATION REQUEST

Please send details, including medical case histories, on how your disk storage systems can provide immunity against: ☐ slow delivery ☐ high prices ☐ limited choice

Name____________________

Phone( )____________ ext______

Company____________ Position____________

Address____________________

City__________ State______ Zip______

System Industries
525 Oakmead Parkway, P.O. Box 425
Sunnyvale, CA 94086

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) 741-3502; **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) 953-0315; **Ohio** (513) 771-0075, (513) 874-5503; **Texas** (713) 497-7224; **Washington, D.C.** (703) 734-9700; **West Germany** (06102) 5464/5; **Sweden** 08-63 62 74

Making Basic-2 Programs Sharable

Dave Kachelmyer, Dan Esbensen, North County Computer Services, Inc., Escondido, California

ABSTRACT

With the advent of RSTS/E V7.0, sharable programs provide a simple solution to the problem of large, concurrently used programs degrading system performance.

INTRODUCTION

At our installation, there are several 31K custom data entry programs that are used by 13 to 18 people throughout the day. These programs place a sizable load on our system because of the large memory requirements, RSTS scheduler overhead, and swapping overhead.

The multi-user feature of Task Builder was used to solve this problem. BASIC-2 program pure areas were extracted from the program and placed in a resident library to be shared by all users of the program. With the pure areas removed, the program size dropped to 7K. Converting these programs resulted in increased system performance because more memory was available, and swapping overhead was greatly reduced.

SHARABLE PROGRAMS UNDER RSTS/E

As part of the resident library feature of RSTS/E V7.0, the task builder was upgraded to handle tasks built with read-only code. This feature is called the multi-user task feature. When this feature is used, the task builder takes all read-only code and moves it to the task's upper address space. This code is then marked as read-only sharable code.

RSTS/E supports sharable programs as a task file/resident library combination. A multi-user task must be converted into separate read-write and read-only code in order to run correctly. This conversion is done with MAKSIL.

When run on a multi-user task, MAKSIL splits the task file into an executable task file and a resident library file. The resident library must be ADDed before the task file can be run.

SHARABLE BASIC-2 PROGRAMS

BASIC-2 programs contain two read-only sections, $CODE and $PDATA. The section $CODE contains threaded code generated from the source program. The section $PDATA contains numeric constants and string literals referenced in the program.

BASIC-2 programs also contain a read-write section, called BP2OTS, which is effectively read-only. This section contains BASIC-2 OTS routines referenced by the threads in $CODE. These routines are extracted from the BP2COM library and placed in the task image by the task builder. Because the modules in BP2COM are defined as read-write, task builder places these modules in the task's lower address space, greatly increasing the size of the non-sharable program segment. This can be avoided by having the task builder force the section BP2OTS into the sharable segment.

BP2OTS is made sharable by forcing the section to be defined as read-only. To do this, a PSECT definition is placed in the program's overlay description file, defining BP2OTS as read-only. This definition forces the task builder to include the BP2OTS code in the read-only sharable segment. However, because this definition conflicts with the OTS module definitions, the task builder generates a warning message for each module included in the task image. The number of modules included in the task can be reduced by taskbuilding against a resident library.

BUILDING A SHARABLE BASIC-2 PROGRAM

The following describes the procedure for building a sharable task from a BASIC-2 source program. The procedure for making sharable programs requires that the MAKSIL patch 11.16.1 be installed. Additionally, the procedure for making a sharable BASIC-2 program requires that the BASIC-2 compiler patch 45.2.12 must be installed. The MAKSIL patch may be found in the February Software Dispatch. The BASIC-2 compiler patch is reproduced in Appendix A.

The steps are:

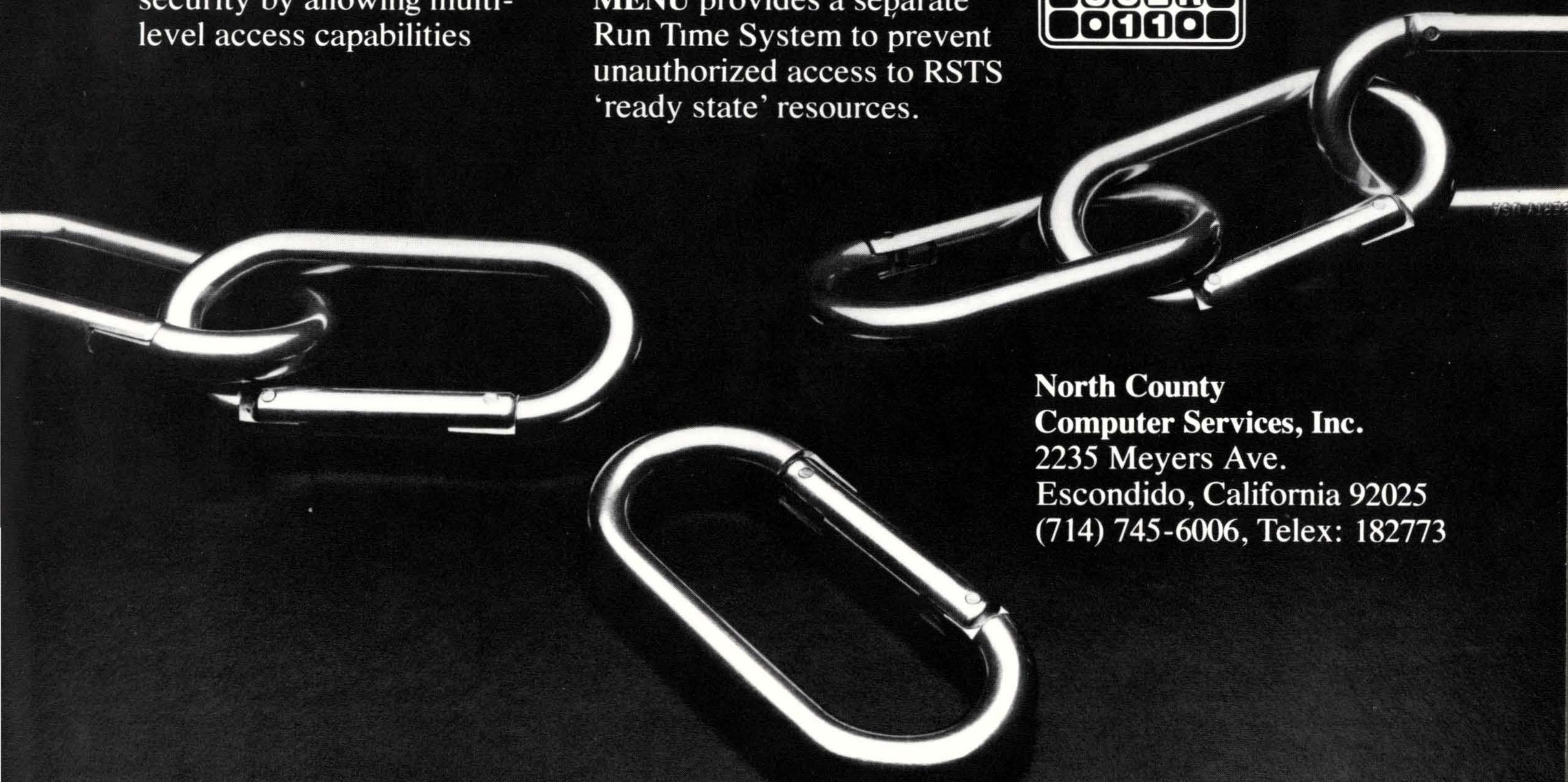
1. Compile program
2. Build task builder control files
3. Edit task builder command file
4. Edit overlay description file
5. Taskbuild program
6. Generate LIB and TSK files with MAKSIL
7. Install Resident Library
8. Test program

The first step involves compiling the program into an object module.

```
OLD TEST
COMPILE TEST/OBJ
```

The next step is to build the Task Builder CMD and ODL files. The program may be built against a BASIC-2 resident library to reduce the number of OTS modules in the task image.

```
HISEG
Name [BP2COM]--NONE
Account [LB:]--
BRLRES
File spec [NONE]--LB:BASICS
BUILD TEST
```



Once the files are built, the Task builder command file is edited to include the /MU switch in the task file specification. The command file should look something like this:

```
SY:TEST/MU = SY:TEST/MP
UNITS = 12
ASG = SY:5:6:7:8:9:10:11:12
RESLIB = LB:BASICS/RO
EXTSCT = 512
//
```

Next, edit the PSECT definition into the overlay description file. This involves adding a PSECT statement and inserting a reference to this statement in the ROOT statement. The ODL file should look something like this:

```
        .ROOT USER-BP2OTS
USER:   .FCTR SY:TEST-LIBR
LIBR:   .FCTR LB:BP2COM/LB
        .PSECT BP2OTS,RO,I,LCL,REL,CON
        .END
```

The next step is to taskbuild the program. Because of the PSECT definition conflicts, the task builder will print a MODULE MULTIPLY DEFINES PSECT BP2OTS message for each BASIC-2 OTS module referenced by the program. These messages (but only these messages) should be ignored.

```
TKB @TEST
MODULE xxxxxx MULTIPLY DEFINES PSECT BP2OTS
.
.
.
TASK EXIT STATUS: ERROR
```

Next, run MAKSIL to split the file into LIB and TSK files. For convenience, the SIL output file can be named as a TSK file.

```
RUN $MAKSIL
MAKSIL V7.0-07 + /MU PATCH
Resident library name? TEST
Task-Build Resident Library input file <TEST.TSK>?
Include symbol table (Yes/No) <YES>? NO
Task Image SIL output file <TEST.SIL>? TEST.TSK
TEST built in 4K-words, 0 symbols in the directory
TEST.TSK renamed to TEST.TSK<104>

PIP TEST.TSK<124>/RE
```

The next step is to add the resident library.

```
UT ADD LIBRARY SY:[1,210]TEST/ADDR:124
```

And then test the program.

```
RUN TEST.TSK
.
.
.
```

Once the test procedure is complete, the program is ready to use. The statements to ADD the resident library segment should be placed in the start-up command files.

ACKNOWLEDGMENTS

This procedure was developed with the aid of information supplied by the Basic-Plus-2 Development Group of Digital Equipment Corporation.

APPENDIX A

RSTS/E V7.0 Software Dispatch, May 1981

BASIC-PLUS-2 V1.60 for RSTS/E V7.0 — Seq 45.2.12 M
BASIC2 Compiler Patches

INCORRECT PSECT ATTRIBUTES

PROBLEM:

$PDATA and $CODE PSECT have incorrect access attributes.

SOLUTION:

This mandatory patch to the Compiler for BASIC-PLUS-2 V1.60 will cure the problem. It must be installed on all versions (EIS, FIS, and FPU) of the Compiler. Article seq. 45.2.7 M must be applied before this patch can be applied.

PROCEDURE:

1. Install the following patch using the ONLPAT program supplied on the distribution media:

```
Command file name? <LF>
File to patch? $BASIC2.TSK
Base address? 742:55724-55054
Offset address? 4
 Base    Offset  Old     New?
000650  000004  002640  ? 2660
000650  000006  002640  ? ^Z
Offset address? 12
 Base    Offset  Old     New?
000650  000012  002440  ? 2460
000650  000014  002640  ? ^Z
Offset address? ^Z
Base address? 574:70456-36734
Offset address? 1562
 Base    Offset  Old     New?
031522  001562  053522  ? 47522
031522  001564  044454  ? ^Z
Offset address? 1720
 Base    Offset  Old     New?
031522  001720  026127  ? 26117
031522  001722  026104  ? ^Z
Offset address? ^Z
Base address? ^Z
File to patch? ^Z
```

2. The compiler is now ready for use.

The RSTS/E Benchmarks Part I

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

benchmark: "A mark made in some durable object, as a wall or other landmark, of known position and elevation, for use as a reference point in surveys or tidal observations."

RSTS/E is certainly a durable object and the RSTS/E Professional a landmark publication, but the goal of this series of articles is to present an examination of how one can develop and use benchmarks as reference points in making evaluations of hardware devices and software techniques. The goal of these evaluations being to improve the performance of your RSTS/E system and your RSTS/E applications.

Our interest is in computer benchmarks — that is benchmarking or measuring the performance of a computer system performing a **specific** task. The term specific task is critical because the goal of benchmarking, like any experiment, is to measure the difference — the difference in performance under differing conditions.

For pure hardware performance this may mean the execution of the same program, typically very scientific and arithmetic in nature, on different hardware configurations and different computer systems. You probably have seen advertisements touting a computer's speed in 'wheatstones per second', one common measure of scientific computational speed. But this type of method neglects one important factor — this benchmark like any benchmark measures one specific condition and this performance cannot typically be generalized to similar tasks. While I am not making an accusation, it would not be difficult to take one of the popular scientific benchmarks and design a FORTRAN compiler or even a hardware instruction set (microcoded as they are) to provide exceptional benchmark performance.

So one of the rules you should follow in reviewing the results of the benchmarks presented in these articles or in any advertisement is:

> Benchmarks are specific but do not necessarily reflect performance in general or in your particular system/application environment.

Another oversight it is easy to make while benchmarking is to be overly concerned with pure hardware speed. As many of you have realized there are many ways to improve system performance — while faster hardware is one way it is often times not the best (and certainly not the cheapest) way. Despite the fact that your sales representative may want to sell you that new fast widget — it may only make a marginal difference in performance.

A for-instance is that disk transfer speed is really less important than you might think. Some benchmarks that we will look at later in this series demonstrate that there are other factors that contribute even more to disk throughput. We will look at many different hardware components and compare them, but more to relate their real performance to their theoretical performance. A second general guideline is:

> Speed does not in itself determine performance.

For example here is one pure speed benchmark. A program that appeared in volume 1, number 1 of the RSTS/E Professional implemented a bubble sort in Macro-11. I modified the program to do the sort fifty times and then executed it standalone on various configurations. The results:

Environment	Time (sec)
11/70 RSTS/E	37
11/44 RSTS/E	55
11/44 RSX11M	56
11/44 RT11	51
11/34 RSTS/E	122
11/34 RSX11M	141
11/34 RT11	111
PDT-150 RT11	236
11/03 RT11	238

I am not going to attempt to draw any conclusions, but I would be interested in hearing from you if you would like to draw some. In fact I would welcome any suggestions or observations you have about it.

A difficulty in benchmarking and one way in which benchmarks mislead is that no benchmark that I develop is perfectly valid for your environment, even assuming it is perfect for mine. One of the goals of this series is to help you discover how to develop your own benchmarks for your own environment. It isn't as hard as you might think.

While my benchmarks may not match your environment I will be showing you the results of some benchmarks you can reproduce on your own system. This will allow you to compare your configuration with those I use. This is one

. . . continued on page 76

RSTS/E System Management

By Jeffrey R. Harrow, 485 Creekview Drive, Stone Mountain, Ga. 30083

Does anyone out there use "forms" (or simulated Block Mode) in their applications to make it easier for a user to interact with his application program?

Certainly, there are a large number of applications written in this manner, but unless you have a CTS-500 (commercial flavor) system and are using DIBOL/DECFORM, you have had to code the forms handling yourself using Echo Control Mode.

The good news is that DEC has come out with an across operating system package called Forms Management System (FMS) which, as the name implies, provides high level forms capability for your applications written in any language which supports a "CALL" verb (NOT BASIC-Plus) without your having to do any of the dirty work.

The bad news is that it won't be available for RSTS/E until around the end of 1981.

The good news is that there is an alternative, Interactive Data Entry (INDENT), available for RSTS/E which does many of the same things as FMS (although in a different manner) and does a few things that FMS does not.

The bad news is that, in some shops, an internal problem between DEC's INDENT and RSTS/E groups which left a special operational requirement (more in a moment) will make INDENT realistically unuseable.

INDENT is composed of an INDENT Run-Time System, an INDENT Compiler, and several INDENT utilities which allow you to work with the forms you design. INDENT is a highly sophisticated package which allows easy use of the many special features of RSTS/E (such as multi-terminal I/O) and the VT100 (all attributes).

In brief, to create an INDENT application, you first decide what the form should look like. You then, using an editor, create a text file which is actually the source code for the INDENT compiler. This is NOT an "on-screen" forms editor (like FMS provides), but, like DECFORM, requires you to create the forms description, then compile it, then display it to see what you have done.

Once you have a compiled INDENT form, you can exercise it (and I don't just say "display" it, because INDENT allows **extremely** sophisticated and versatile forms which can have many parts, chain between forms, do scrolling within a form, etc.) with some of the supplied INDENT utilities. You then put this aside, and create your application program which will "drive" the form. The application program essentially does Field **I/O** to/from the form using a few CALLS (which are resolved from the INDENT object library at Task Build time).

All fairly simple, except for this one problem: The first thing that your application program must do is "fire up" the INDENT form job. This is necessary because your application program runs in JOB A, linked to whatever Run-Time System is appropriate (BASIC2, RSX, etc.) and your first CALL to INDENT **SPAWNS** JOB B, which runs the compiled INDENT form and is linked to the INDENT Run-Time System. (When you do the CALLS to do the Field I/O to/from your form, the INDENT routines linked into your application program are actually doing Message Send/Receives to the associated INDENT form job (JOB B).)

And here is the crux of the problem ... the SPAWN Syscall requires Privilege, and that means that your **application program** must have Permanent (shudder) or Temporary Privilege to start its INDENT job.

In any sanely run RSTS/E shop, VERY FEW privileged passwords are up for grabs, and this means that **each** application programmer who is writing INDENT application programs must go to one of the Systems Staff **each** time that he re-task builds his application program (which happens quite a bit during program development/testing). If the Systems Staff is conscientious, they must then go over the application program **with a fine toothcomb** during each of these iterations, to make sure that the program will not compromise the system once he gives it the Temporary Privilege attribute. This can easily constitute an unacceptable level of effort for the Systems Staff (or could cause them to shirk their duty and, perhaps, give the Temporary Privilege attribute to a program with a bug which, while running Privileged, could cause harm to the system or other users).

During the course of my SPRs with DEC on this subject, I received an interesting "finger pointing" response from the INDENT group, indicating that they had requested a Feature Patch to RSTS/E allowing an installation to make the SPAWN Syscall non-privleged, however the RSTS/E group would not commit to providing such a patch, much less provide an expected date. A later communication suggested writing a simple utility which would allow a non-privileged user to make a program privileged himself ... anyone willing to have such a utility?

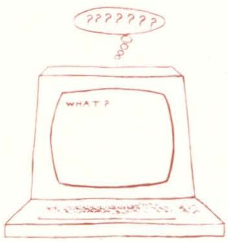
Therefore, although INDENT is an extremely versatile forms package, if you are thinking of purchasing it for your shop consider two items: Can you afford (you can't afford NOT to) the resources to scan your application programmers' INDENT application programs **each time** a change is made; and do you have any expectation of moving the application to a non-RSTS/E system such as RT11, VAX, or RSX. (Note that INDENT forms bear NO relation to an FMS equivalent, **nor will they ever do so.**) If these are not problems to you, INDENT may well provide the forms capabilities we've all been waiting for.

but if these **are** problems, prepare to wait for FMS-RSTS/E.

Concerning the Datatrive problem which I discussed last issue, DEC's response was a change to the installation procedure to **remove all protection from the central data dictionary**, or use private dictionaries. Come on, DEC! A central precept of the use of DTR in many shops **is** the use of a central dictionary, and it is against all conceivable security considerations to leave such a central dictionary completely unprotected! I realize that this requires some changes to DTR for RSTS/E (implementing the Temporarily Drop Temporary Privilege Syscall in a few places and giving DTR.TSK a protection code of < 232 > and LB:QUERY.DEC a protection code of < 60 >), but we're paying for a product that works correctly on our RSTS/E systems, as well as on RSX11M, etc!

Speaking of DTR, there's another RSTS/E related problem. The RSTS/E standard for error messages is for the first character of an error message to contain either a "?" or a "%' to indicate that it is a FATAL or WARNING message, respectively. DTR does **NOT** preface its error messages with these characters, which means that you may get quite unexpected results when running DTR as part of a BATCH or ATPK stream. I've submitted another SPR on this subject, and I'll keep you informed on its response. In the meantime, examine any such job streams to be sure that if the DTR portion has a fatal error, the rest of the job stream, which WILL continue, does not do anything which would prevent you from fixing the DTR problem and re-running the job stream. See you next issue. ♥

DEAR RSTS MAN:



DEAR RSTS MAN:

1) The RSTS Professional contains more useful information on one page than DEC could ever hope to fit into one of its manuals.

2) There is a bug in PIP.SAV (at least in V06C). As far as I can tell, PIP.SAV uses the following routine:

COPY FILESPEC IN-NAME TO OUT-NAME
1) FOR FIL-NUM = 0 TO 32767
 1.1) DO A DIRECT LOOKUP ON IN-NAME. GET OCCURRENCE NUMBER FIL-NUM TO IN-NAME-1 (LIKE THE DIRECT LOOKUP ON WILD CARD SYSCALL, FIP = 17.)
 1.2) ON ERROR FROM ABOVE, GOTO 2
 1.3) CHANGE WILD CARDS IN OUT-NAME AS NEEDED. PUT RESULT IN OUT-NAME-1
 1.4) COPY IN-NAME-1 TO OUT-NAME-1
 1.5) NEXT FIL-NUM
2) END

This works fine most of the time. If you try to copy files within an account the following happens:

PIP SY0: *.* = SY0: *.*

(DIRECTORY CONTAINS 1.FIL, 2.FIL, AND 3.FIL)

(PIP COPIES 1.FIL TO 1.FIL)

(DIRECTORY CONTAINS 2.FIL, 3.FIL, 1.FIL IS NEW FILES LAST)

(PIP COPIES THE SECOND FILE IN THAT ACCOUNT, 3.FIL, TO 3.FIL)

(DIRECTORY CONTAINS 2.FIL, 1.FIL, 3.FIL)

1.FIL will not get copied. This causes problems when:

PIP SY0:*.*/CL:16 = SY0:*.*	:MAKE ALL FILES :CONTIGUOUS
PIP SY0: = SY1:	:COPY SYSTEM :DISK 1 TO :SYSTEM DISK 0

3) I *think* there is a bug in REORDR.BAC— it uses the following routine:

REORDER USER FILE DIRECTORY [4,5]
1) OPEN A SCRATCH DIRECTORY
2) FOR FIL-NUM = 0 TO 32767
 2.1) GET STATISTICS (NAME,EXT, LAST ACCESS DATE, etc.) ABOUT FILE NUMBER FIL-NUM
 2.2) IF NOT THERE, GOTO 3
 2.3) IF IT IS OPEN, THEN ABORT
 2.4) COPY STATISTICS TO SCRATCH DIRECTORY
 2.5) NEXT FIL-NUM
3) COPY ENTIRE SCRATCH DIRECTORY TO DIRECTORY OF (4,5)

Imagine the following happens:

1) User (1,10) is reordering account (4,5) which has 200 files in it. The reorder takes a long time.

2) User (4,5) logs in, deletes the first 10 files in his account, and logs off.

3) REORDR.BAC reordered files 150 to 180 while the above was going on. It was not paying attention to files 1-10 which have now been deleted.

4) The scratch directory is copied into the real directory.

I hope I missed something in REORDR.BAC which checks for this, as the results could be grim.

4) The article on RSTS V7 internals was very informative. Most of what was said (almost all of the useful information) is applicable to RSTS V06C.

5) ??? How can you tell if a printer is printing or if it is just sitting there? ??? Is there any way you can PEEK into its ??? Buffer Control Area somehow and ??? find out if there are any small ??? buffers which are to be printed on ??? the printer once it finishes up? ???

Sincerely, Kenneth Boorom
Grade 10, Brunswick School
Riverside, CT

Dear Ken: *I would not call #2 a bug, but a feature. A very dangerous feature! However, this will work on DECTAPE! In the very old days DECTAPES did not free up space when a file was deleted . . . a "Copy and Clean" operation was required and if you had lots of time only one DECTAPE was needed.*

Number 3, when running REORDER always SET NO LOGINS or Else!

Number 5 was submitted to a monitor Guru. His answer:

```
10000   !                                     &
        !FNWAIT% -- WAITS FOR PRINT           &
        !HEAD TO STOP ON LA36'S               &
        !
10010   DEF FNWAIT%
10020   T%=PEEK(PEEK(PEEK(520%)))             &
        \GOTO 10090 IF PEEK(T%+10%)=PEEK(T%+12%)
        \SLEEP 1%                             &
        \GOTO 10020
10090   FNWAIT%=0%                            &
        \SLEEP 3%                             &
        \FNEND
```

How come it took until 10th grade for you to think of all this? How about writing us an article on how and what you and your school do on RSTS. Keep up your interest!

—

DEAR RSTS MAN:

Problem: A Fortran user (under RSTS/E) has a large program that manages extensive files. During certain points in the program execution he wishes to interrupt the program without terminating its execution. Having so interrupted it, he will handle the situation within his program — i.e., he can tidy his files files and make an orderly exit or resume processing.

It's easy enough to handle this problem if we were dealing with Basic-plus, but in Fortran it's not so simple. The idea was to be able to type a single character at the terminal and the program would test for the existence of it. If it wasn't there, it would go on about its business.

After much trial-and-error coding, the program KBT, attached, worked out but not very well. With a simple Fortran calling program which did nothing but loop on the call to KBT and test the flag, the real time between character input and reaction of the program to it averaged 45 seconds! Secondly, KBT seems only to react to a carriage-return; other characters (without being followed by a return) have no influence on it.

```
        TITLE KBT
;
;SUBROUTINE TO TEST KEYBOARD FOR EXISTENCE OF TYPED CHARACTER
;
;CALLING SEQUENCE:   CALL KBT(IFLAG)
;
;                IFLAG=13 IF NO DATA HAS BEEN ENTERED
;
XRB=000442
FIRQB=000402
 READ=104002
 rsts    =      104377
KBT::    CLR    @2(R5)
         MOV    #XRB,R0               ;PUT XRB INTO R0
         MOV    #128.,(R0)+
         CLR    (R0)+                 ;BYTE 2 IS 0
         MOV    #BUF,(R0)+            ;ADDRESS OF BUFFER
         MOV    #0 ,(R0)+             ;CHAN ZERO
         CLR    (R0)+                 ;BLOCK #
         CLR    (R0)+                 ;WAIT TIME
         MOV    #8192.,(R0)+          ;MOD FOR NO DATA
          rsts                        ;tell RT11 to let rsts handle the EMT
          READ                        ;ISSUE NO-WAIT READ TO TERMINAL
         MOV    @#FIRQB,@2(R5)        ;PUT STATUS IN IFLAG
OUT:     RTS    PC

BUF::     BLKB  128
          END
```

I have two questions: (1) How can I make this program behave as it should, and (2) Will somebody please explain why this program (or for that matter, any MACRO program accessing the monitor data or using the directives) only works if you include the directive ".RSTS"? It's almost like magic, especially since no one anywhere, including your authors, ever mentions it.

Ms. E.G. Johnston
Director, Computer Center
Western Maryland College

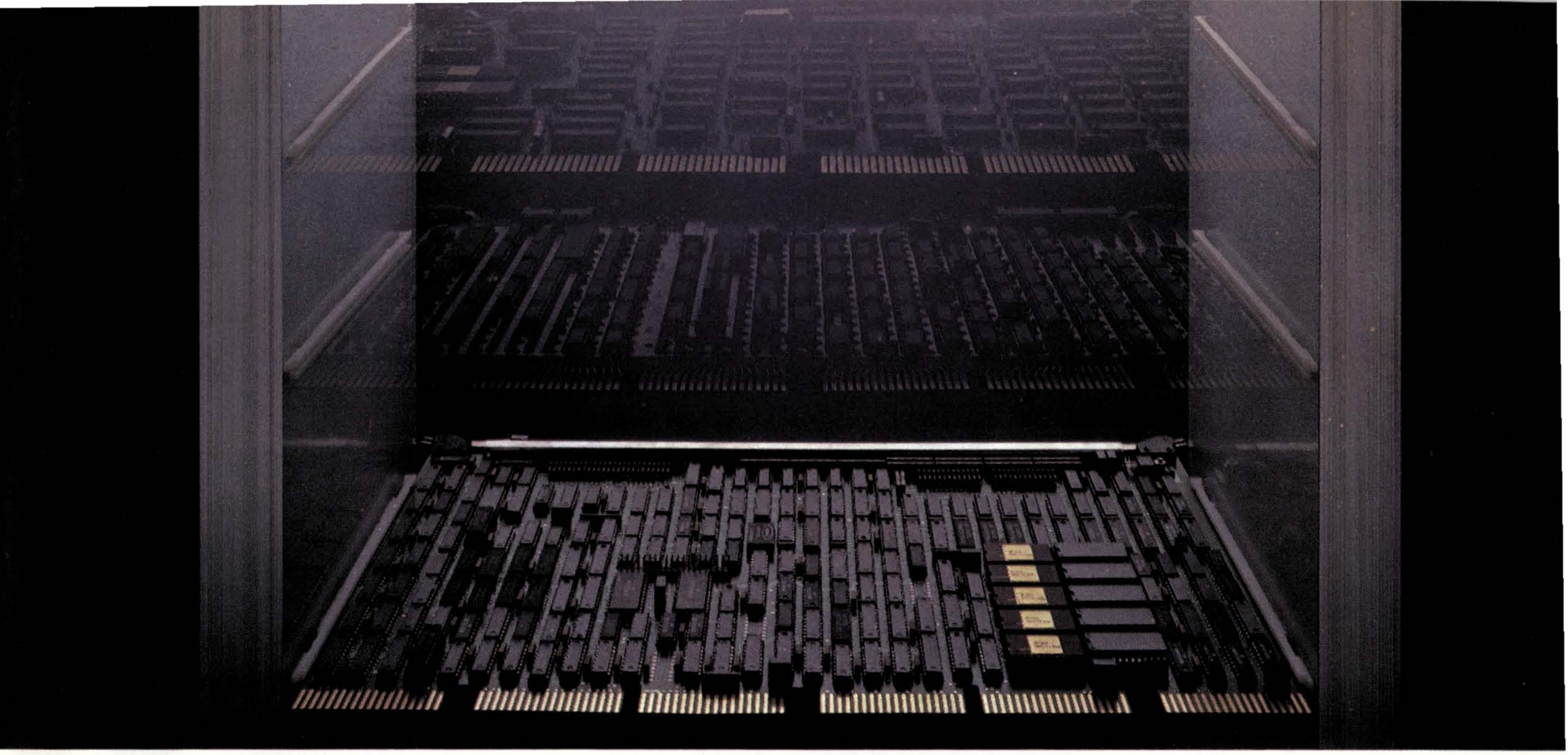
Dear Ms. Johnston: *1) We don't know, but a BASIC+2 version follows that works fine!.*

```
10      call Kbt(t%)
20      goto 10 if t% = 13%
30      print t%
90      end
```

```
>ty Kbt.mac
.title  Kbt
.ident  /test/

xrb     =       442
firqb   =       402
.read   =       104002
buff:   .blkb   128.                    ;buffer for holding things

kbt::   clr     @2(r5)                  ;clear return code
        mov     #xrb,r0                 ;point to xrb
        mov     #128.,(r0)+             ;setup buffer size for read
        clr     (r0)+                   ;'must be zero'
        mov     #buff,(r0)+             ;address of buffer
        clr     (r0)+                   ;chan/msb block
        clr     (r0)+                   ;lsb block
        clr     (r0)+                   ;wait time
        mov     #8192.,(r0)+            ;modifiers
        .read                           ;do the read
        mov     @#firqb,@2(r5)          ;return the error code
        rts     pc                      ;return to caller

        .end
```

2) The .RSTS directive is required to prevent the RT11 emulator from intercepting the EMT, but to pass the next monitor call to RSTS.

. . . continued on page 66



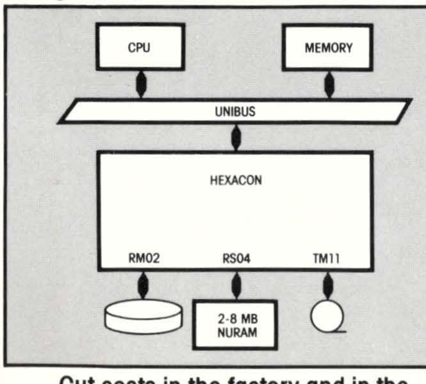
HEXACON from National. A new dimension in inner space.

National unleashes the first DEC®-compatible peripheral controller that does the work of three.

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 HEXACON™ controller.

HEXACON is our new hex-wide peripheral controller that simultaneously handles up to four RM02/RM03 80 MB disk drives, four TU10 ½" tape drives and 8 MB of our NURAM™ semiconductor disk. All from just a single slot.



Cut 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, HEXACON'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, HEXACON 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. HEXACON 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. HEXACON 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/RM03 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—HEXACON 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. HEXACON'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 HEXACON controller, simply send in the coupon below or call Bill LeDuc toll-free at **(800) 538-8510** or **(800) 538-8514.** In California call **(408) 736-6994.**

HEXACON's efficiency and practicality is just what I need to cut my costs down to size.
Please send me:
☐ HEXACON information.
☐ NURAM information.
☐ DEC Add-in Memory information.

NAME ______

COMPANY ______

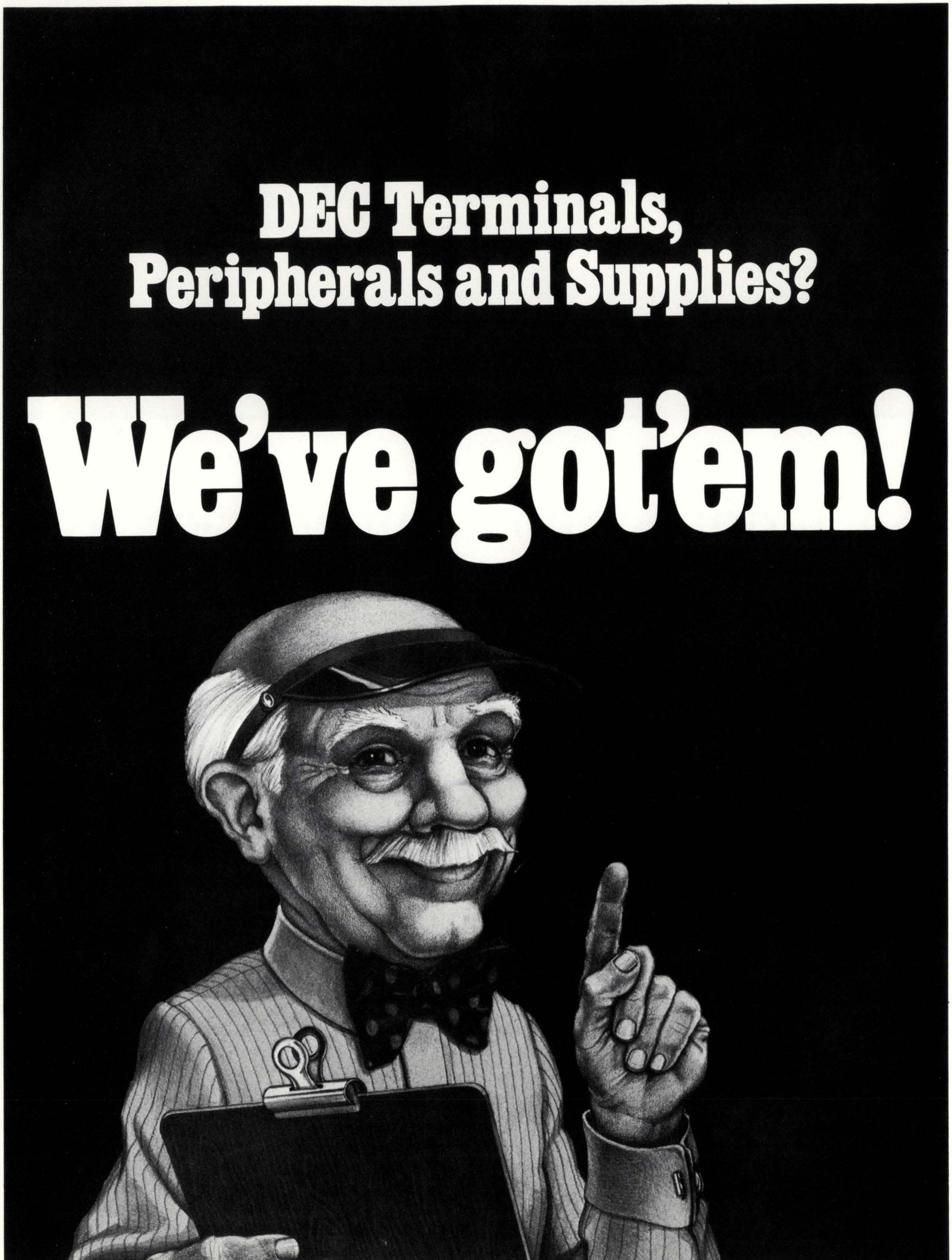
ADDRESS ______

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


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

DEB, a Basic-Plus 2 Mini-debugger

By Steven L. Edwards, Software Techniques, Inc.

1.0 Abstract

The Basic-Plus 2 language provides the applications programmer with a very powerful program debugging facility. Unfortunately, there are two serious problems with the supported debugging facility: size, and lack of features. The debugging facility described in this article was written with these problems in mind.

2.0 Disclaimer

This article describes the author's experience with RSTS/E V7.0 and Basic-Plus 2 V1.6 and may not be accurate in other operating environments. This document may contain information that is not part of the supported functionality of RSTS/E or Basic-Plus 2 and therefore is subject to change without notice.

3.0 Description

DEB is a tiny Basic-Plus 2 debugging routine, stripped down to the bare minimum functionality to provide some semblance of debugging facility to programs that are too large to make use of the supported Basic-Plus 2 debugging facility. Gone are variable display (PRINT), variable assignment (LET), and most of the breakpoint features. In their place are GOTO's, RESUME's, and executing a trap to ODT (if linked into the program).

In use, DEB replaces the LIN$ (line number routine). Thus, the resolution of our breakpoint and tracing control is at the line number level, not at the statement level.

4.0 History

Software Techniques is a software house specializing in RSTS/E and Basic-Plus 2/RMS-11K applications. Because of the 16 bit addressing limitation (31KW) on PDP-11's, we found ourselves without any facility to debug large applications programs. DEB was written to give us a trace of the line numbers being executed. However, just watching the trace was not enough, so breakpoints were added. GOTO and RESUME were soon added in response to a dare (who can leave a challenge un-challenged?). We develop our programs in a mix of Basic-Plus 2 and MACRO (see RP, V2 #4), so the 'O' command was added to allow the use of a low level debugging tool like ODT.

5.0 Command Set

The command set available to the applications programmer, while admittedly a restrictive subset of the supported debugger command set, includes a few tricks that belong in the supported debugger. Listed below is the DEB command set. The command parser is very simplistic (less memory), so all commands consist of a single character and a line number if applicable.

- B linenumber—set breakpoint.
- C—continue until next breakpoint.
- E—exit program.
- G linenumber—GOTO linenumber.
- O—enter ODT.
- R linenumber—RESUME linenumber.
- S—step.
- T—enable tracing.
- U linenumber—un-set breakpoint.
- <CR>—step.

6.0 Linking Instructions

Since DEB replaces the LIN$ routine, compile your program with "/LIN/NODEB" to insure that this routine will be executed at each line number (LIN) and that we have enough memory to execute our program (NODEB).

Do not link to BASICS because references to resident libraries are resolved before references to object modules or libraries.

When you link your program, you will get the 'error' message:

```
%TKB — *DIAG*-MODULE $ERROR MULTIPLY DEFINES SYMBOL LIN$
```

Since our module is referenced first, the task-builder will ignore the Basic-Plus 2 routine.

7.0 Linking Example

```
>BP2

PDP-11 BASIC-PLUS-2 V1.6 BL- 01.60

BASIC2

OLD DEBTST

BASIC2
```

"I (for once) was speechless."

-- Dave Mallery, March, 1981 issue of
RSTS PROFESSIONAL

To tell the truth, so are we. We knew DISKIT would amaze RSTS users, but, frankly, we were unprepared for the response. Phone calls, letters, and now the RSTS PROFESSIONAL -- all saying what we want you to know:

DISKIT is a remarkable software tool!

Listen to what else Dave has to say:

"...using DISKIT, I created 130 accounts and fully extended their centered UFDs in 3 minutes and 40 seconds (a job that used to take 4 to 8 hours.)"

"I then copied the full contents of a 300 MB RM05 equivalent to this new 'well-structured' disk in 45 minutes, optimizing clustersize and contiguity in the process..."

DISKIT IS A DISK STRUCTURING UTILITY

As Dave discovered, DISKIT's disk structuring utility, DSU, is fast. It also:

- Optimizes file clustersizes
- Places and pre-extends UFDs
- Performs transfers between unlike disks
- Saves all accounting data
- Allows manual file placement
- Provides full logging and statistics
- Includes sophisticated error handling and recovery

DISKIT IS A DIRECTORY PROGRAM

But DISKIT is more than a disk structuring utility. DISKIT's Macro-11 directory program, DIR, displays directories 12 times faster than before -- looking up files by name, extension, and date (with wildcards) at the incredible rate of 250 files/second.

And DIR is smart. It supports all standard DIRECT switches (including backwards, up to 1,000 files) with features you won't find elsewhere - like password lookup, UFD placement, and UFD size.

DIR even works as a diagnostic tool on dismounted disks, detecting bad directory structures and identifying them with comprehensive error messages.

DISKIT IS AN OPEN FILES DISPLAY PROGRAM

DISKIT's Macro-11 OPEN program displays open files by job – with complete job and file statistics. It even has a "sleep switch", allowing you to dynamically update information at any desired interval.

DISKIT LETS YOU WRITE YOUR OWN DISK HANDLING ROUTINES

Best of all, the very same routines used in DISKIT are included, with documentation, so you can write your own disk handling routines. In minutes.

DISKIT IS THE FIRST SOFTWARE TOOL KIT FOR COMPLETE DISK MANAGEMENT

The DISKIT package provides all the tools and utilities you need to create and manage a well-structured disk. The entire DISKIT package, with extensive documentation is available now for only $1250.

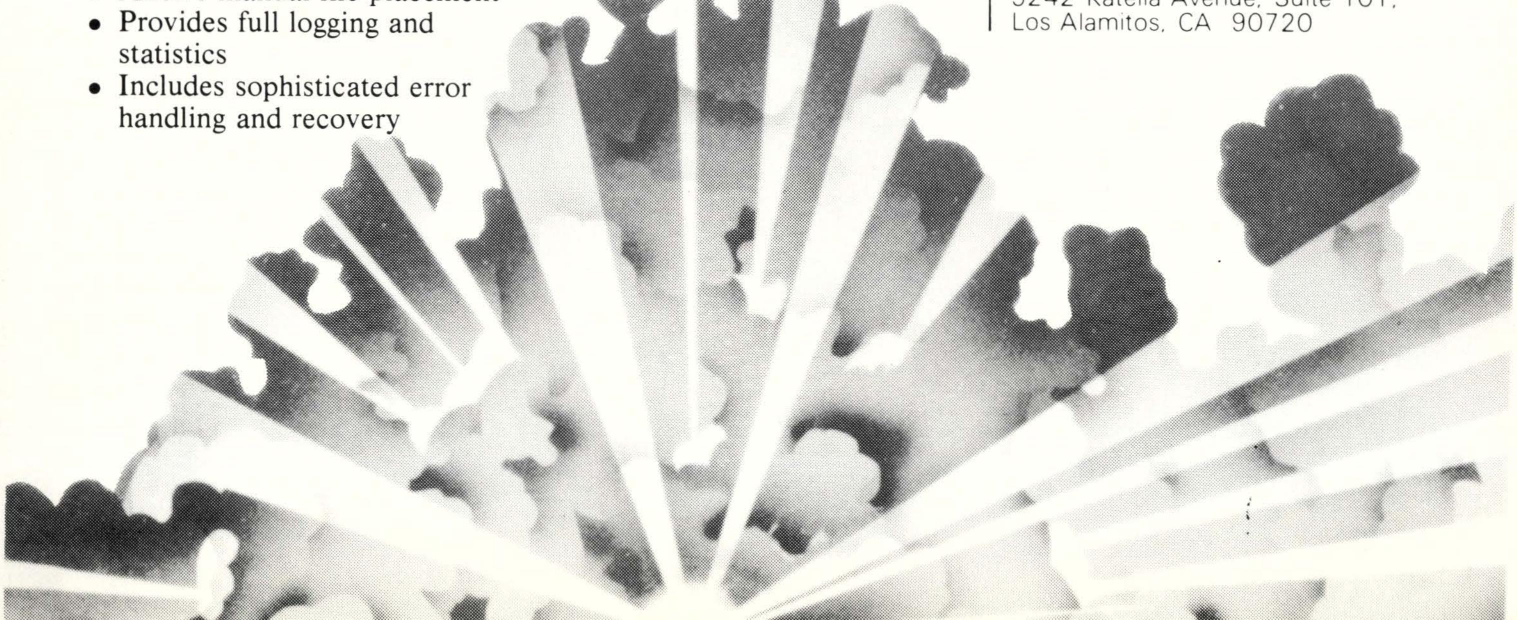
DISKIT, Dave says, "...is the 'final solution' to structured disks, eliminating all of the time and complexity and reducing the job to one of a SAVRES."

What more could we say?

Once again, we've got the answer.

Software Techniques, Inc.

5242 Katella Avenue, Suite 101,
Los Alamitos, CA 90720



```
LISNH
1000        ONERROR GOTO 19000
1010        PRINT "THIS IS LINE 1010"
1020        TEMP.0% = VAL("OOPS!")
1030        PRINT "THIS IS LINE 1030"
1040        OPEN "" FOR INPUT AS FILE 1%
1050        PRINT "THIS IS LINE 1050"
1060        GOTO 32767
19000       PRINT "THIS IS LINE 19000"
19010       PRINT "ERROR!, ERR = "; ERR; & "ERR = "; ERL
32767       END

BASIC2

COM/OBJ/LIN/NODEB

BASIC2

HIS
Name [NONE]-NONE
Account [LB:]-

BASIC2

BRL NONE

BASIC2

BUI DEBTST.DEB

BASIC2

TKB @DEBTST
%TKB - * DIAG* -MODULE $ERROR MULTIPLY DEFINES SYMBOL LIN$

%Task exit status: ERROR
```

8.0 Sample Execution

When the program is run, DEB displays a header, and the first line number awaiting your command. As the program executes, DEB displays the current line number at each breakpoint or step. If the program has an error pending (has not beed RESUME'd), the relevant information is displayed before the current line number.

```
>RUN DEBTST

Deb:DEBTST V7.0-05 Software Techniques

1000> ; STRAIGHT EXECUTION
1000> ;
1000> C
THIS IS LINE 1010
THIS IS LINE 19000
ERROR!, ERR = 52 ERR = 1020

>RUN DEBTST

Deb:DEBTST V7.0-05 Software Techniques

1000> ; BREAKPOINT DEMO
1000> ;
1000> B 1020
1000> B 19000
1000> C
THIS IS LINE 1010
1020> C
Ern:DEBTST, Err:00052, Erl:01020
19000> C
THIS IS LINE 19000
ERROR!, ERR = 52 ERR = 1020

>RUN DEBTST

Deb:DEBTST V7.0-05 Software Techniques

1000> ; GOTO DEMO
1000> ;
1000> G 1050
1050> S
THIS IS LINE 1050
1060> G 1000
1000> B 19000
1000> C
THIS IS LINE 1010
Ern:DEBTST, Err:00052, Erl:01020
19000> G 1000
Ern:DEBTST, Err:00052, Erl:01020
1000> C
THIS IS LINE 1010
?Illegal number at line 1020 in "DEBTST"

>RUN DEBTST

Deb:DEBTST V7.0-05 Software Techniques

1000> ; RESUME DEMO
1000> ;
1000> B 19000
1000> C
THIS IS LINE 1010
Ern:DEBTST, Err:00052, Erl:01020
19000> R 1000
1000> C
```

```
THIS IS LINE 1010
Ern:DEBTST, Err:00052, Erl:01020
19000> R 1030
1030> C
THIS IS LINE 1030
Ern:DEBTST, Err:00010, Erl:01040
19000> R 1050
1050> C
THIS IS LINE 1050

>RUN DEBTST
```

Deb:DEBTST V7.0-05 Software Techniques

```
1000> ; TRACE DEMO
1000> ;
1000> T
1000> C
At line 1000
At line 1010
THIS IS LINE 1010
At line 1020
At line 19000
THIS IS LINE 19000
At line 19010
ERROR!, ERR = 52 ERR = 1020
At line 32767

>RUN DEBTST
```

Deb:DEBTST V7.0-05 Software Techniques

```
1000> ; BREAKPOINT DEMO
1000> ;
1000> B 19000
1000> C
THIS IS LINE 1010
Ern:DEBTST, Err:00052, Erl:01020
19000> U 19000
Ern:DEBTST, Err:00052, Erl:01020
19000> R 1030
1030> C
THIS IS LINE 1030
THIS IS LINE 19000
ERROR!, ERR = 10 ERR = 1040
>
```

9.0 Summary

DEB is a small (under 500 words) limited function debugging routine for Basic-Plus 2 programs. In many cases, the functions available will allow the applications programmer to debug the program at hand. DEB is not intended to be a functional replacement for the supported debugging facility, but it does have a place in the applications programmer's toolbox.

```
        .Enabl  LC

Title   DEB,<Mini-debugger/tracer>,05,02-MAY-81,<Sle>

;       Package:        In-House

;       Written by:     Steven L. Edwards

;       Date:           23-Aug-80

;       Copyright (C) 1980
;       Software Techniques
;       Los Alamitos, California 90720

;       This  software  is  furnished  under  a license for use only on a
;       single  computer system 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 except for use on such system and to one who
;       agrees  to  these  license  terms.  Title to and ownership of the
;       software shall at all times remain in Software Techniques.

;       This  information  in  this document is subject to change without
;       notice  and  should  not be construed as a commitment by Software
;       Techniques.

;       This  software  is  un-released  and  Software  Techniques has no
;       commitment  to  support  it at this time, unless stated elsewhere
;       in writing.
```

```
;
;       Modification History
;
;       Ver/Edit        Date            Reason
;       --------        ----            ------
;
;       V7.0-01         23-Aug-80       Initial conception.
;                                       (SLE)
;       V7.0-02         30-Aug-80       Add debugger stuff.
;                                       (SLE)
;       V7.0-03         03-Sep-80       Add ERN, ERR, ERL.
;                                       (SLE)
;       V7.0-04         23-JAN-81       FIX MISC. BUGS.
;                                       ADD "O" COMMAND.
;                                       (SLE)
;       V7.0-05         02-MAY-81       TEST FOR ODT BEFORE BPT.
;                                       TEST FOR EXACT MATCH ON LINENUMBER.
;                                       (SLE)


;
;       General Description
;
;
;               This routine replaces the Basic-Plus 2 LIN$ routine.
;       This routine allows you to do some minor debugging, and display
;       some tracing info.
;
;       Debugging commands available are:
;
;               B linenumber            ; Set breakpoint at linenumber.
;               C                       ; Continue until next breakpoint.
;               E                       ; Exit program.
;               G linenumber            ; Goto linenumber.
;               O                       ; Execute BPT (enter ODT.)
;               R linenumber            ; Resume linenumber.
;               S                       ; Set step mode.
;               T                       ; Set tracing flag.
;               U linenumber            ; Un-break linenumber.
;               X                       ; Exit program.
;               <CR>                    ; Advance to next step/breakpoint.
;
;       Assembly instructions:
;
;               MAC DEB = PRE, COMMON, DEB
;                               ^       ^
;                               !       !
;                               !       ------- is on the SYSGEN tapes
;                               -------------- is on the BP2 tapes
;
;       Linking instructions:
;
;               Edit the ODL, CMD, or TKB command line to include a
;       reference to the object module DEB. Note: do not link to BASICS
;       because references to resident libraries are resolved before
;       object modules. Ignore the TKB error message:
;
;               %TKB -- *DIAG*-MODULE $ERROR MULTIPLY DEFINES SYMBOL LIN$
;
;               Since our module is referenced first, the task-builder
;       will ignore the Plus-2 routine.


;
;       Global Symbols
;

        .Globl  Lin$                    ; Entry points.
        .Globl  $Cbdsg                  ; Number conversion routine.
        .Globl  $Cdtb                   ; Number conversion routine.
        .Globl  $Otsv                   ; OTS pointer.
        .Mcall  Exit$s                  ; Exit macro.


;
;       Macros
;

.Macro  Cmd     Char,Rtn
        .Byte   ''Char,0                ; Character command.
        .Word   Rtn                     ; Routine address.
.Endm   Cmd

.Macro  Num$    Num
        Push    R1                      ; Save R1.
        Push    R2                      ; Save R2.
        Mov     Num,R1                  ; Number in binary.
        Mov     #377,R2                 ; No zero suppression.
        Call    $Cbdsg                  ; Convert Binary to Decimal, SiGned
        Pop     R2                      ; Restore R2.
        Pop     R1                      ; Restore R1.
.Endm   Num$

.Macro  Num1$   Num
        Push    R1                      ; Save R1.
        Push    R2                      ; Save R2.
        Mov     Num,R1                  ; Number in binary.
        Clr     R2                      ; Zero suppression.
        Call    $Cbdsg                  ; Convert Binary to Decimal, SiGned.
        Pop     R2                      ; Restore R2.
        Pop     R1                      ; Restore R1.
.Endm   Num1$

.Macro  Print   Addr, Len
        Call    Clrxrb                  ; Clear the XRB.
        Mov     Len,(R0)                ; Length of text string.
        Mov     (R0)+,(R0)+             ;    "     "    "     "
        Mov     Addr,(R0)+              ; Address "    "     "
        .Write
.Endm   Print


;
;       Variable Description and Initialization
;

        .Bsect                          ; Rsw status bits.
A:      .Blkb   .                       ; Advance to next step/breakpoint.
S:      .Blkb   .                       ; Step pending.
T:      .Blkb   .                       ; Trace requested.

        .Psect  $CODD
$CODD:                                  ; Start of $CODE.
        .Psect  $CODF
$CODF:                                  ; Start of $CODF.

        .Psect  $$$ODS
$$$ODS:                                 ; Start of $$$ODS.
        .Psect  $$$ODU
$$$ODU:                                 ; Start of $$$ODT.

        .Psect  Deb                     ; Data area.

;       Byte aligned data.

Etext:  .Ascii  /Ern:/
E1:     .Blkb   6
        .Ascii  /, Err:/
E2:     .Blkb   5
        .Ascii  /, Erl:/
E3:     .Blkb   5
        .Byte   15, 12, 0
Elen    =       .-Etext

FTEXT:  .ASCII  /%BPT TBL FULL./
        .Byte   15,12
Flen    =       .-Ftext

Gtext:  .ASCII  /%LINE NOT FND./
        .Byte   15,12
Glen    =       .-Gtext

        .Even;  in the quietest moments...
KBUF:
Itext:  .Byte   15,12
        .Ascii  /Deb:/
Name:   .Blkb   6
        .Ascii  / V/
        .Byte   Sysvel/400&377, Sysvee&377
        .Byte   Sysvee/400&377, '-
        .Byte   $$$ver&377, $$$ver/400&377
        .Ascii  / Software Techniques/
        .Byte   15, 12, 12
Ilen    =       .-Itext
Klen    =       .-Kbuf                  ; Length of keyboard buffer.
Otext:  .Ascii  /%ODT NOT FND./
        .Byte   15,12
Olen    =       .-Otext

Ntext:  .Ascii  /%BPT NOT FND./
        .Byte   15,12
Nlen    =       .-Ntext

Ttext:  .Asciz  /At line /              ; Tracing text.
        .Byte   15,12,0

Wtext:  .Ascii  /?What?/
        .Byte   15,12
Wlen    =       .-Wtext

        .Even

;       Word aligned data.

Bkptbl: .Blkw   12                      ; Breakpoint table.
Bkplen  =       <.-Bkptbl>/2            ; Size of breakpoint table.

Caltbl: Cmd     B,Bre                   ; Set breakpoint.
        Cmd     C,Con                   ; Continue.
        Cmd     E,Exi                   ; Exit.
        Cmd     G,Got                   ; Goto.
        Cmd     O,Odt                   ; Execute BPT.
        Cmd     R,Rsu                   ; Resume.
        Cmd     S,Ste                   ; Step.
        Cmd     T,Ett                   ; Enable tracing info.
        Cmd     U,Unb                   ; Un-break.
        Cmd     X,Exi                   ; Exit.
Callen  =       <.-Caltbl>/4            ; Number of entries in call table.

Curlin: .Word   0                       ; Current line number.

Rsw:    .Word   S&^C<T>                 ; Routine status word.

;
;       Mainline Program
;

Ini:    Mov     #NOP,Lin$               ; So that we will only do this once.
        Getots  R0                      ; Get OTS pointer.
        Add     #Nmptr,R0               ; Point to module name pointer.
        Mov     (R0),R0                 ; Point to module name.
        Mov     #Name,R2                ; Point to name area.
        Mov     (R0)+,(R2)+             ; "NO
        Mov     (R0)+,(R2)+             ;   NA
        Mov     (R0)+,(R2)+             ;     ME"
        Print   #Itext,#Ilen            ; Print out the identifier.

Lin$:   Br      Ini                     ; Only the first time.
        Mov     R4,@$Otsv               ; Store the pointer to the line
        Mov     (R4)+,Curlin            ; Save the current line number.
        Push    R4                      ; Save R4.

        Mov     #Bkptbl,R0              ; Address of breakpoint table.
        Mov     #Bkplen,R1              ; Length of breakpoint table.
10$:    Cmp     Curlin,(R0)+            ; Is this it?
        Beq     20$                     ; Sure enough.
        Sob     R1,10$                  ; Loop until done.

        Bit     #S,Rsw                  ; Step pending?
        Beq     70$                     ; Don't stop the task.

20$:    Bis     #S,Rsw                  ; Set step mode.

30$:    Bit     #A,Rsw                  ; Advance?
        Bne     70$                     ; Don't stop the task.

        Bit     #S,Rsw                  ; Step pending?
        Beq     70$                     ; Don't stop the task.

        Getots  R0                      ; Point R0 at $Otsv.
        Bit     #Epend,Bitwrd(R0)       ; 'Error pending'?
        Beq     40$                     ; Good.
        Mov     #E1,R2                  ; Point at ERN$ text output area.
        Mov     Ern1(R0),(R2)+          ; "NO
        Mov     Ern2(R0),(R2)+          ;   NA
        Mov     Ern3(R0),(R2)+          ;     ME"
```

```
        Mov     Errnum(R0),R1           ; ERR.
        Mov     @Erlptr(R0),R2          ; ERL.
        Mov     #E2,R0                  ; Point at ERR output area.
        Num$    R1                      ; Convert ERR to ASCII.
        Mov     #E3,R0                  ; Point at ERL output area.
        Num$    R2                      ; Convert ERL to ASCII.
        Print   #Etext,#Elen            ; Print out the error text.

40$:    Mov     #KBUF,R0                ; Address of output area.
        Num1$   Curlin                  ; Convert to ASCII.
        Movb    #'>,(R0)+               ; ">"
        Movb    #40,(R0)+               ; <SP>
        Mov     R0,R1                   ; Save the final address.
        Sub     #KBUF,R1                ; How long?
        Print   #KBUF,R1                ; Print prompt.

        Call    Clrxrb                  ; Clear XRB.
        Mov     #KLEN,(R0)+             ; Length of input buffer.
        Tst     (R0)+                   ; Skip XRB+XRBC.
        Mov     #Kbuf,(R0)              ; Address of input buffer.
        .Read                           ; Do the get.

        Call    Clean                   ; Clear up the command
        Cmpb    (R0),#'!                ; Basic-Plus (2) comment?
        Beq     40$                     ; Go get another command.
        Cmpb    (R0),#';                ; RSX style comment?
        Beq     40$                     ; Go get another command.
        Tstb    (R0)                    ; See what we got.
        Bmi     70$                     ; If minus, then advance.
        Mov     #Caltbl,R2              ; Address of call table.
        Mov     #Callen,R3              ; Number of entries in table.
50$:    Cmpb    (R0),(R2)               ; Right command?
        Bne     60$                     ; Too bad.
        Tst     (R2)+                   ; Point to the routine address.
        Call    @(R2)                   ; Dispatch to the routine.
        Br      30$                     ; Branch back for next command.
60$:    Add     #4,R2                   ; Point to next table entry.
        Sob     R3,50$                  ; Loop until done.
        Print   #Wtext,#Wlen            ; Tell the poor abuser.
        Br      30$                     ; Branch back for next command.

70$:    Bic     #A,Rsw                  ; Clear advance bit.
        Bit     #T,Rsw                  ; Tracing requested?
        Beq     80$                     ; ?Esta muy loco?
        Call    TRA                     ; Print the tracing info.
80$:    Pop     R4                      ; Restore R4.
        Jmp     @(R4)+                  ; Exit. (Back to Plus 2)

;
;       Subroutines
;

;
;       Set a breakpoint.
;

Bre:    Mov     #Bkptbl,R4              ; Address of breakpoint table.
        Mov     #Bkplen,R1              ; Length of breakpoint table.
10$:    Tst     (R4)+                   ; Empty?
        Beq     20$                     ; Good.
        Sob     R1,10$                  ; Loop until done.
        Print   #Ftext,#Flen            ; Tell the poor abuser.
        Br      30$                     ; Table full, too bad.
20$:    Mov     #Kbuf+1,R0              ; Point at line number.
        Call    $Cdtb                   ; Convert Decimal To Binary.
        Mov     R1,-(R4)                ; Put the number into the breakpoint
                                        ;   table.
30$:    Return                          ; Back to mainline code.

;
;       Clean up the input buffer.
;

Clean:  Mov     #Kbuf,R0                ; Address of input buffer.
        Mov     @#Xrb+Xrbc,R1           ; Lenght of text entered.
        Mov     #Kbuf,R2                ; Address of 'cleaned input.'
        Add     R0,R1                   ; End of input.
10$:    Cmp     R0,R1                   ; At end?
        Beq     40$                     ; Good, lets go home.
        Cmpb    (R0),#40                ; <SP>?
        Bgt     20$                     ; Ignore all trash.
        Inc     R0                      ; Bump pointer past trash.
        Br      10$                     ; Loop back for next character.
20$:    Cmpb    (R0),#'A+40             ; lower-case character.
        Blt     30$                     ; Ok.
        Bicb    #40,(R0)                ; Convert to UPPER-CASE.
30$:    Movb    (R0)+,(R2)+             ; Move a byte.
        Br      10$                     ; Loop until done.
40$:    Movb    #377,(R2)               ; Mark the end of the input.
        Mov     #Kbuf,R0                ; Point at the cleaned input.
        Return                          ; Back to mainline code.

;
;       Clear Xrb.
;

Clrxrb: Mov     #Xrb,R0                 ; Address of Xrb.
        Push    R0                      ; Save the address of Xrb.
        Push    R1                      ; Save the old R1.
        Mov     #<Xrbsiz/2>,R1          ; Size of Xrb in words.
10$:    Clr     (R0)+                   ; Zap a word.
        Sob     R1,10$                  ; Loop until done.
        Pop     R1                      ; Restore old R1
        Pop     R0                      ; Restore address of xrb.
        Return                          ; Back to mainline code.

;
;       Continue.
;
```

```
Con:    Bic     #S,Rsw              ; Clear 'STEP PENDING'
        Bis     #A,Rsw              ; Set advance.
        Return                      ; Back to mainline code.

;
;       Enable tracing
;

Ett:    Bis     #T,Rsw              ; Set the proper bit.
        Return                      ; Back to mainline code.

;
;       Exit program
;

Exi:    Exit$s                      ; Exit to default RTS.
        Halt                        ; We shouldn't get this far.

;
;       Goto linenumber.
;

Got:    Mov     #Kbuf+1,R0          ; Point at line number.
        Call    $Cdtb               ; Convert Decimal To Binary.
        Mov     #$CODD,R0           ; Point to start of $CODE.
        Mov     #$CODF,R3           ; Last word in $CODE.
        Mov     #Lin$,R2            ; Address of our LIN$ routine.
10$:    Cmp     R0,R3               ; End of $CODE?
        Bhi     30$                 ; Too bad.
        Cmp     (R0)+,R2            ; Pointing at LIN$?
        Bne     10$                 ; Loop until it is.
        Cmp     (R0),R1             ; Right line number?
        Bne     10$                 ; Too bad.
20$:    Tst     -(R0)               ; Decrement to point at LIN$.
        Mov     R0,2(SP)            ; Point to that statement.
        Bis     #A,Rsw              ; Set advance bit.
        Br      40$                 ; Exit routine.
30$:    Print   #Gtext,#Glen        ; Tell the poor abuser.
40$:    Return                      ; Back to mainline code.

;
;       Move a chunk of text.
;

Movtxt: Movb    (R2)+,(R0)+         ; Move a byte.
        Bne     Movtxt              ; Until we get a null.
        Dec     R0                  ; Backup pointer one char.
        Return                      ; Back to mainline code.

;
;       Execute BPT (enter ODT)
;

Odt:    Tst     #$$$ODU-$$$ODS      ; Is ODT here?
        Bne     10$                 ; Yes, do the trap.
        Print   #Otext,#Olen        ; No, tell the poor abuser.
        Br      20$                 ; And exit.
10$:    Bpt                         ; Enter ODT.
20$:    Return                      ; Back to mainline code.

;
;       Resume linenumber.
;

Rsu:    Getots  R0                  ; Point R0 at $Otsv.
        Bic     #Epend,Bitwrd(R0)   ; Clear 'error pending.'
        Br      GOT                 ; Same code as GOTO.

;
;       Set step mode.
;

Ste:    Bis     #A!S,Rsw            ; Set step mode and advance.
        Return                      ; Back to mainline code.

;
;       Print the tracing info.
;

Tra:
        Mov     #Kbuf,R0            ; Point at the output buffer.
        Mov     #Ttext,R2           ; Point at the text strings.
        Call    Movtxt              ; Move a chunk of text.
        Numl$   Curlin              ; Current line number.
        Call    Movtxt              ; Move a chunk of text.
        Mov     R0,R1               ; Save final address.
        Sub     #Kbuf,R1            ; How long?
        Print   #Kbuf,R1            ; Print out the tracing data.
        Return                      ; Back to mainline code.

;
;       Un-break.
;

Unb:    Mov     #Kbuf+1,R0          ; Point at line number.
        Call    $Cdtb               ; Convert Decimal To Binary.
        Mov     #Bkptbl,R2          ; Address of breakpoint table.
        Mov     #Bkplen,R3          ; Length of breakpoint table.
10$:    Cmp     R1,(R2)+            ; Found it?
        Beq     20$                 ; Good.
        Sob     R3,10$              ; Loop until done.
        Print   #Ntext,#Nlen        ; Tell the poor abuser.
        Br      30$                 ; Table full, too bad.
20$:    Clr     -(R2)               ; Clear out the entry.
30$:    Return                      ; Back to mainline code.

        .End
```

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.

From the editors . . .

. . . continued from page 4

Dave Mallery

simultaneous TECO users editing macro for micros. Try doing that with EDT.

All of the above sessions are great examples of how the user community can use DECUS to meet its real needs.

The gang from North County Computer Services delivered an excellent series on Basic + 2 while Software Techniques carried on with their speciality—disk optimization.

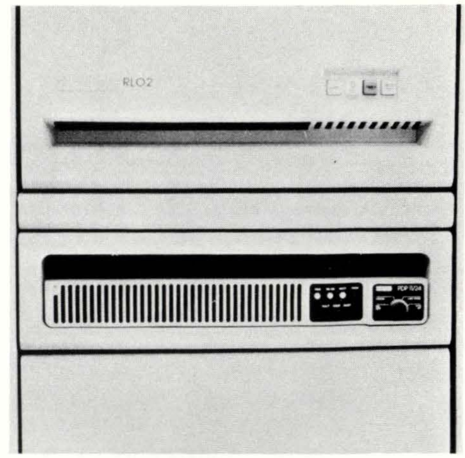
I noticed large numbers of users complaining about disk problems in the open sessions — "We get multiply allocated clusters all the time", "My system stops cold and locks up if I run out of room on my RM05" etc.,etc.

Clearly, there are lots of field service and software service types in need of some education. It amazes me to see that so many people could actually believe they have software problems when some controller was delivering data blocks into their satt buffers!

There was an excellent presentation by Glen D. Kinzey from Professional Datasystems on "Basic Communications". This is a clearly NON DEC area that should be developed in future symposia. So few users have any idea about modems and muxes. I recently heard about a site using 22 lines and 22 pairs of Bell rented modems on an application that could have used one or two lines and pairs of stat mux's.

I attended a "Commercial VAX" session. Unfortunately, the title of the session is still a contradiction. The session was like a late—5C RSTS wish list in 1975. I feel it will take two or three more years for VAX to recover from its initial RSX / scientific bias. Boy am I glad I don't have to waste six DECUS'S lobbying for what I already have. There were practically no user papers from the VAX folks—too bad.

New RSTS Product Manager J. Thomas McKinney Miami DECUS



The New 11/24! The RSTS Engine for the 80's. Miami DECUS

I think that the high point for me, was the opportunity to hear Gordon Bell reminisce on the invention of the UNIBUS. His talk consisted of a large number of "lessons" from the history of computers, illustrated profusely with slides from the Digital Computer Museum. DECUS has provided me with the ability to meet several of the heroes and heroines of our industry in person and I am very grateful. I was also delighted that Mr. Bell had indeed heard of our magazine.

The 11/24 was available in the exhibit hall for inspection. This is the RSTS engine for the next few years. It is too soon to really have any good benchmark info. on this machine, but by next DECUS, we should have some good numbers. I wonder (to myself) if it will get CPU bound before it can really use all that memory-time will tell. This is the machine that clearly shows the directions of RSTS, however, Gordon Bell made a remark that impressed me — he said: "always resist the impulse to build the last machine of a series." I guess that's why we have an 11/24 and not (yet) and 11/84. The Commercial OEM group stated in a product panel that they have chosen RSTS as the operating system of preference for PDP/11's in the 80's. The RSTS product line folks said they expect 20,000 RSTS licenses by 1985 (and agree with me that there are about 8000 known licenses now).

One notable comment — "Datatreive has their WOMBAT, RSTS should have a squirrel mascot because we're always caching."

The logistics—Miami is nice, especially after a bad winter. The hotels are beautiful but the prices are more suited to an OPEC meeting or a convention of Cocaine dealers. The "food" that was "served" at the convention center was a disgrace. We stopped eating there after the second lunch. I think DECUS should send out refunds and sue Miami. The campgrounds were so far away that one seriously considered camping there.

A few years ago, you bought DEC controllers or you took chances.

Time was you couldn't trust anyone. Price was often lower. Delivery was sometimes faster. But let's just say quality, reliability and service left lots to be desired (if the product even worked at all).

Times change. Enter Emulex.

First we decided that price shouldn't be the only reason to buy our controllers even though they cost less. The design goal of every tape, disk and communications controller we make is to go DEC one better. Emulex controllers actually optimize the DEC systems in which they're installed.

Emulex controllers would also be easier to install – plug 'em right into a backplane – hardware and software transparent.

We decided, also, to concentrate our expertise on DEC. We know DEC computers and designed products to improve them. Others didn't. You wouldn't remember their names.

In the last two years one Emulex success has led to another. Today, Emulex provides OEMs and end users with the industry's broadest line

of controllers that improve DEC performance across the board. If we did anything wrong, it was in setting a new standard. Competitors stopped imitating DEC and started trying to imitate us.

So even today, unless you're careful, you can still take chances. There's a big difference between an imitation DEC controller and a genuine Emulex.

Emulex. The genuine alternative.

EMULEX

EMULEX CORPORATION
2001 East Deere Avenue
Santa Ana, CA 92705
(714) 557-7580
TWX 510-595-2521

A WORD ABOUT THE AUTHOR . . .

Mike Draper is a principal in RAXCO, a computer software and services company specializing in DEC computer resource recovery systems. These systems include job accounting, performance analysis, resource accounting and auditing under VAX/VMS, RSTS, and RSX11m+.

Mr. Draper has more than 20 years of experience in the data processing field, having worked with IBM and Multiple Access Ltd. before joining RAXCO.

In addition to resource recovery systems, Mr. Draper has extensive experience in financial planning systems.

ANALYSING AND ALLOCATING DATA PROCESSING RESOURCES

By Mike Draper, RAXCO INC.

Despite the constantly decreasing hardware costs and the improvements in software tools, the cost of data processing activities remains a substantial expense for most companies, regardless of the type of system or service that they use. All too often, some, most, or all of this expense is allocated to the "Data Processing" or "Computer" department. Actually, however, most of the cost of running a computer center should relate to end-users of the system rather than to the group that maintains the system. Whether the users are paying real cash dollars for the computer services or simply shifting around budget allocations makes no difference. A prime concern is still to account fairly and accurately.

Two major problems that prevent many companies from using billing or chargeback as an effective management and control tool might be referred to as the "Shoemaker's Children" and the "Cart/Horse" syndromes. In the former, the DP personnel are so busy processing everyone else's data that they haven't the time to process their own. With the latter problem, when the question is asked "Why don't you do proper accounting?" The answer is all too often, "The machine doesn't provide the data." And all too often, upper management accepts these excuses thereby sacrificing not only accountability, but more importantly, a large measure of control over computer related facilities.

WHO NEEDS WHAT AND WHY!

A good place to start with this investigation, as with many other types of projects, is at the end. What kind of reports would the controller or VP finance like to see on his or her desk before making decisions about adding hardware? What type of reports would tell the manager of information systems if a new policy was having the desired effect? What information would be valuable to the system manager to evaluate the effect of a department's use or to anticipate the effect of adding a new application? Perhaps most important of all is what kind of feedback would help the end-user to use the DP resources more effectively?

There is another very important function performed by machine use accountability. Look at this graph:

```
                              ***
                          ************
                      ********************
                  ****************************
            ****************************************
Sun      Mon      Tue      Wed      Thur      Fri      Sat
```

This is a very frequent occurrence. The *'s represent the amount of machine resources used each day. The Wednesday amounts are limited by the resource available. The totals for the other days are limited by various other factors. The white space above the the *'s on all except Wednesdays represents idle machine time. There is a similar profile for the hours of the day except it is two humped because of lunch hour.

While this is not the case for all installations, it occurs more frequently than not. It is interesting to note that most DP managers and even fewer people in positions of accounting or responsibility are aware of the extent of the idle time on a "fully loaded system". I would not presume that some sort of resource accounting or auditing can cure this, or that a complete cure is possible or even desirable. Many financial people, however, feel twinges in their ulcers when they compare Mondays and Fridays with Wednesdays.

There are three areas where the information about who is using what, when is useful (or perhaps necessary).

1. Financial Management
2. Operational Management
3. End-user Management

Information, of course, is just the bread of the management sandwich. On top, it tells you where to start and on the bottom, it tells you if anything has changed. Effective resource accounting can also provide much of the meat of the sandwich as a tool for implementing policy.

IMPLEMENTING POLICY.

How can we get more people to use the computer on Mondays and Fridays? One way would be to just tell the worst offenders (if we can identify them). "You stop using the computer on Wednesdays, you turkeys!!" In addition to perhaps not being an effective management approach, it is also going after a very complicated problem with a very blunt instrument. A more delicate approach might be more effective.

Just the way every department head or project manager must estimate his or her manpower and other physical needs, why not have estimates of computer use? These, of course, might be based on accounting for previous projects or periods. Then, by giving discounts for off-prime time and perhaps even penalties for super-prime time, the users would have the option of trading off convenience and time effectiveness for cost effectiveness. There would be direct feedback every billing cycle and the ability to review patterns of changing use.

Where projects or departments have budgets, the periodic review of percent complete versus budget used would be useful not only in evaluating the current period, but more importantly, for planning the next cycle. I am certainly not recommending the chargeback of computer use as a threat of heads rolling. The state of the art of estimating computer use is still far too primitive to expect accuracy. It is only through constant feedback that the art or science of estimating will develop.

Realize the need for accountability is only the first step. Perhaps you have run through this scenario before.

You: Frank, I need some kind of chargeback, performance evaluation, departmental accounting system.

Frank: Well, you just tell me EXACTLY what you want and I'll get someone on it as soon as ...

You: I'm not exactly sure what I want, but it has to be useful and usable by various management levels.

Frank: [Lecture #324 "Computers Require Precision"]

WHAT YOU REALLY NEED!

You need a tool that will allow you to get answers to the type of questions that you know you have to ask, but that you can't ask until you get some other answers first.

You need to give Finnicky Frank enough detail to shut him up and Ball-park Bob a brief enough summary so he might read it.

You need simple pictures with simple numbers that the VP finance can stick on his wall and point to in meetings.

You need general information to spot long term trends and specific information to analyze trouble spots.

You need a standard set of reports that can be compared cycle to cycle, plus the adaptability to meet changing requirements and provide special one shot reports.

You need various levels of detail.

You need ALL expenses relating to the DP facility combined into one statement.

You will need various things that neither you nor I can anticipate until their time comes around.

HOW DO YOU GET WHAT YOU NEED?

You could go to your Systems Manager (Frank) and he will do one of two things. He might try to avoid the whole area as in the previous example, or he might say "Hey, that sounds like a terrific idea. I'll get someone on it right away." In this case you are really in trouble. "Someone" quickly becomes the permanent billing programmer/operator. You will not only not get the information that you need, but you will not get it weeks late and when you do get it, it will be wrong. Not getting what you want will also cost you more than you care to admit.

A better approach might be to look around to see if maybe someone else has attacked the same problem. The manufacturer of your equipment might have some useful software or even more useful ideas about "another 2 megs" or "distributed processing" (another computer). Hopefully the time will come when machine accountability is considered as important a part of an operating system as a good hex dump. You might also take advantage of the various user groups. It is difficult to evaluate software from glossy brochures and slick sales pitches.

How do you know when you do find a piece of software that may do the job? Particularly when you do not know what that job is?

EVALUATING A COMPUTER ACCOUNTABILITY SYSTEM.

The criteria for evaluating a computer auditing tool may be divided into several categories.

Flexibility

While flexibility is a virtue in almost any situation, it is more necessary in the audit and control area than in many other data processing activities. This is due firstly to the changing computer hardware and software environment and secondly to the need of an auditing tool to respond to specific questions that can not always be anticipated. This differs from the usual accounting software that, once installed, requires only minor alteration from time to time.

It is easy to look at a sample of reports and say, "Boy, could we use that kind of data!!", and forget to question if there is any more where that came from. Both the Data Processing and Accounting people will have various "suggestions" to improve the reports, and the system that can not easily respond to these "suggestions" will soon be relegated to a dusty backup tape of once bright ideas.

Completeness

In the area of cost allocation, it is easy to focus on the machine oriented costs and dismiss the so-called "minor" support costs. While it is true that the CPU/CONNECT time values give a good rule of thumb as to who is doing what, there are far more costs associated with a computer center than the computer itself.

Who is responsible for terminals, remote printers, tapes, manuals, courses, deliveries, programming, consulting and support? While sometimes some of these items must be included in "overhead", more often they can be allocated to specific users (if the system permits allocation of these costs). Many service bureaus have found that thousands of dollars have "slipped through the cracks" before they tightened up their charging system.

Another area of completeness relates to the so-called "special" types of services that each different company provides. Do you offer plotting or data conversion or other off-line facilities? Does it make sense to charge for some services on a per unit basis? Will the resource allocation system you are evaluating handle these types of accounting now and in the future?

Accuracy

Accuracy refers to more than just not making errors. It refers to the level of detail possible in both charging and reporting. An example might be 'Terminal Connect Charges'. You start out by saying "We will charge $10.00 per hour.", and then quickly realize that users with high speed equipment should be charged more than low speed equipment. Then someone asks about dedicated ports and someone else about the cost of using the package switching networks. Pretty soon the simple $10.00 per hour has mushroomed into a very complex algorithm.

While it may not be absolutely necessary to be quite so finicky about accuracy, the people who are being charged will be and accountants well know that the usefulness of any accounting relates directly to its accuracy. Even if you do not anticipate currently the need for great detail, it is definitely a plus in a computer auditing system.

Acceptability

No one likes getting bills. We all like it a lot less when we don't understand why we are being charged so much. Another thing that we don't like is surprises. If we were charged $3000 last month, how come we are being charged $7000 this month? Computer people have a long legacy of providing end-users with the right answers in the wrong format. This is often because the end-user thought that the wrong format was the right format, until he actually had to read the report. The success of any computer accountability system is dependent upon the acceptance of the accountable user.

This is often not an easy problem to solve. Some users may want to see all the bits and bytes while others just want to see summaries. The system manager may need data in one format, while the head accountant may require the same data in another format. These people all tend to ask the embarassing question, "If that computer is so darn smart why can't it provide information in a form we can understand?" If a computer accounting program is so smart, it should be able to.

Selectivity

There is a great deal of data available for both the billing and performance evaluation aspects of computer auditing. The usefulness of the auditing system depends upon being able to select the data relating to the specific problem or question and reject the rest.

Control

There is not automatic replacement for good management and corporate policy. The most you can expect from a computer auditing tool is a way to evaluate the need for policy changes and to evaluate the success of changes after they have been made. The only exception is where the policy change involves some type of discounting, surcharging or other billing technique to give end-users the choice of using prime or off-prime resources. This should certainly be provided by the system.

Vendor Support and Response

In areas where the needs are well defined, you can, by carefully choosing the product, be fairly confident that except for bugs and initial training, the product will do the job required. In the fairly new, and usually poorly defined field of computer auditing, after-sales support and response to problems and needs is vital. This goes beyond bugs and training.

It is in the area of strategy that the vendor can be particularly useful. The disadvantage of flexibility is that it is not always obvious how a tool can be used to solve a certain problem. When the software vendor has expertise in the area and is willing to share that expertise in a responsive manner, the tool can always be used most effectively.

Specific Features

Salesmen of all products love to extoll the various features of their products. What you, the buyer, must do is translate those features into benefits as they relate to your needs.

One such 'feature' might be the ability to rank the various users of your system by the number of CPU seconds that they have used. This may be interesting, but I am sure that the information could be compiled quickly from existing sources. How about ranking on something more exotic like I/Os or page faults? How about ranking on the ratio of page faults to CPU seconds? The question that you must ask is "How can I use this data to improve efficiency and/or make money?" If this feature looks like it has definite uses fine; if not, do not be misled by clever advertising.

There are three areas that require answers from a computer auditing system. They are WHO, WHAT and WHEN. There are various combinations and permutations of these that give the answer to each specific question.

For billing purposes we need to know WHO used WHAT WHEN.

For performance evaluation the sequence is WHAT was used WHEN by WHO.

By selecting and limiting the WHO, WHAT and WHEN and controlling the format of the answers all of the various 'features' are important. Do not limit your thinking to "Boy, that would be nice to have." Continue with "If I had that and also . . . I could . . ."

GETTING EVERYONE INVOLVED!

Often the resource allocation problem is left to the computer department ("They have the computer and the programmers, you know!"). While of course the computer department must be involved in the whole process and will get stuck with running the system, everyone from the financial people to the accounting department, to the various end-user groups must provide input. They will get stuck with the output.

Computer auditing is one of the applications where the dollar savings should be several times the cost of the system the first year and the savings on headaches and ulcers several times that. Experiences in installing systems in the last year have convinced me that savings can be great and the experience relatively painless (nothing involving computers is completely painless). ♥

LETTERS to the RSTS PRO . . .

. . . continued from page 6

Dear RSTS Pro...,

It was aggravating! I *saw* a TECO truck drive by the other day, but I didn't see any company name. However, the phonebook had one potential and here it is:

Trigon Engineering Corporation
doing maintenance on a Toll both set up.

Sorry that the guess was so long in coming, but the last time I entered one of your contests you said, "perhaps the folks at SPU Computer Services, Seattle Pacific University, can get back to work!"* and when *RSTS Professional* speaks . . .

Mark Emerson, Analyst/Programmer
SPU Computer Services, Seattle, WA

*Vol. 2, No. 2, p. 31, col. 3, . . .

Well Mark, we're pleased to hear from you again and we're also pleased to see that you're still working [sic] at SPU. Say "Hi" for us to Mark A. and Tim R. and tell them that, ***right or wrong,*** *we enjoy the time that the folks at SPU spend with us.*

Unfortunately, your TECO answer is not quite the one we're after . . . READ ON . . .

FOLLOWING ARE CORRECT ANSWERS:

Dear RSTS Professional:

I have learned that the answer to "TECO", is a company in Ft. Wayne, Indiana. They produce the lifts and buckets for all kinds of utility trucks. Upon further investigation, I learned that **"teco" does not stand for anything**. I was told that it was the name of the company, and that was all.

Sincerely, Daniel B. Wheeler
Digital, Indianapolis, IN

Leave it to someone from DEC to solve our TECO problem. Thank you, Daniel. It is interesting to note that after all the mystery and confusion, the answer is so simple. As with the editor — the word is fast, easy to use and extremely versatile.

Dave and Carol, [sic]

Since our company is an Authorized Distributor for DEC dealing with the Utility industry, we did not want an unfair advantage in your picture contest. However, enough is enough; it's time to put an end to this one and go on to something else.

Enclosed is an ad brochure for Teco, Inc. This company manufactures several types of Telescoping Buckets and such for the Utility industry. I hope this is sufficient info in your quest for a correct answer. A large Tee-shirt would be nice.

Sincerely, Thomas E. Nelson
Manager, Technical Services
Digital Systems, Inc.

See cover and Tom in his large Tee-shirt.

finis

Here, for the record, is a corrected version of both TTOPNF.MAC and TTDVR.TEC.

```
TITLE   TTOPNF,<CTRL/F OPEN FILES>,07,28-AUG-80,SPD

;       COPYRIGHT (C) 1980, 1981
;       BY SOFTWARE TECHNIQUES, INC., LOS ALAMITOS, CA 90720
;
; THIS SOFTWARE IS PROVIDED 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 EXCEPT FOR NON-COMMERCIAL USE AND
; TO ONE WHO AGREES TO THESE LICENSE TERMS. TITLE TO AND OWNERSHIP
; OF THE SOFTWARE SHALL AT ALL TIMES REMAIN IN SOFTWARE TECHNIQUES.
;
; THIS INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE WITHOUT
; NOTICE AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY SOFTWARE
; TECHNIQUES
;
; THIS SOFTWARE IS UN-RELEASED AND SOFTWARE TECHNIQUES HAS NO
; COMMITMENT TO SUPPORT IT AT THIS TIME, UNLESS STATED ELSEWHERE IN
; WRITING.

.SBTTL  SET UP WINDOW POINTERS
.ENABL  LC

.DSECT  ,NOCREF

F$LINK: .BLKW           ;POINTER TO NEXT FCB THIS UNIT
F$FID:  .BLKW           ;FILE ID (LINK TO NAME ENTRY)
F$PPN:  .BLKW           ;PPN OF FILE
F$NAM:  .BLKW           ;FILE NAME
        .BLKW           ;SECOND WORD FILE NAME
        .BLKW           ;EXTENSION
F$STAT: .BLKB           ;FILE STATUS BYTE
F$PROT: .BLKB           ;PROTECTION CODE
F$ACNT: .BLKB           ;OPEN COUNT
F$RCNT: .BLKB           ;READ REGARDLESS COUNT
F$WFND: .BLKW           ;FBB OF FIRST WINDOW
        .BLKW           ;RESERVED, RETRIEVAL ENTRY
F$UFND: .BLKW           ;CURRENT FBB
        .BLKW           ;OF NAME ENTRY
F$UNT:  .BLKB           ;FIP UNIT NUMBER
F$SIZM: .BLKB           ;FILE SIZE MSB
F$SIZL: .BLKW           ;FILE SIZE LSB
F$CLUS: .BLKW           ;FILE CLUSTER SIZE
F$WCB:  .BLKW           ;FIRST WCB FOR THIS FILE

.DSECT  ,NOCREF

W$IDX:  .BLKB           ;HANDLER INDEX (0=DISK)
W$STS:  .BLKB           ;STATUS FLAGS
W$JBNO: .BLKB           ;JOB # * 2
W$FLAG: .BLKB           ;FLAG BITS
W$PT:   .BLKB           ;PENDING TRANSFERS
```

. . . continued on page 67

DIRECT.TEC

By David Spencer

DIRECT.TEC was born when I began investigating the TECO operator "EN". I had never used it before and needed some application to help discover the kinks. What I finally arrived at was a simple directory program. I then put it away and went on to other things. With the arrival of version seven of RSTS, I returned to my little macro and expanded it to make use of the new FIP PPn wildcards.

Running DIRECT.TEC

The DIRECT.TEC program makes use ONLY of the "EN" verb. The directories contain less information than a program that utilizes "SYS( )" calls (or TECO :EG support). "EN" returns only file specification and protection code.

The information passed to DIRECT.TEC must be a single file specification without switches. Such specs as "SY:" or "[1,2].BAK" which are not completed with the asterisks are allowable, however. If no specification is supplied, then a directory of the current account on the system disk is performed.

The macro itself uses numeric registers zero through six. However, it pushes and pops them so that no data stored in those registers is lost. When running DIRECT.TEC while in TECO, it is VERY important to remember that the macro assumes the file spec to be the complete contents of the text buffer. The result is that the directory will be attempted to be taken on your text file if you haven't done an "EC". Also, after DIRECT.TEC has completed, the text buffer will be cleared. Beware!

There are three regular ways to invoke DIRECT.TEC: by CCL outside of TECO, by "EI" within TECO, and loading the macro into a register and executing it. The CCL for DIRECT.TEC is "FILES". (This of course can be changed.) If in TECO and you want to do a directory, then clear the text buffer area, insert your file specification, and "EI" DIRECT.TEC or load DIRECT.TEC into a register and execute that register name.

What you get.

The directory returned looks a lot like the directory returned by PIP or DIRECT.BAS. The difference is, of course, you get only the disk, account, file name and extension, and protection code. Unfortunately "EN" does not return file sizes, dates, or times. There is a nice summary at the bottom, and if there was a PPn wildcard, a grand total of all files and accounts listed.

Why bother?

Why even write a directory program in TECO? The reasons are not overwhelming. About the only time I ever wind up using it is to impress someone, or, when the system is slow, avoid having to exit TECO and then re-load everything.

There has been an interesting fact revealed from my exploration of the "EN" verb. I'll call this one "The Case of the Missed Edit". This anomaly was uncovered when I was "mass editing" a series of files using a macro that had a wildcard "EN" in it. I had opened my file with "EB" and allowed TECO to create a ".BAK" file for me. I would then make the changes (by macro) and get the next file. This would continue until I was out of files to edit.

The problem occurred that I wound up editing EVERY OTHER file, and some files got edited TWICE! The reason was this: "EN" returns the "nth" file number, where "n" is the number of times that "EN" is executed. The trouble was after the first "EN". This causes the next "EN" to get the second file in the directory matching the wildcard. But the first file was edited and the old version renamed to ".BAK". Thus the second file in the directory by that wildcard became the first to match. So therefore, every other file would be missed, and some would be mistakenly edited twice from the end of the directory because their name still matches the wildcard.

The moral of this story is to always output to a different extension when doing wildcard edits. I hope this little headache of mine won't be repeated by those who have read this and not yet explored the mysteries of "EN".

Closing Notes

As with all TECO code, DIRECT.TEC is just a base program. I hope that, as Carl Sagan would say, billions and billions of little macros evolve from this humble beginning. Enjoy!

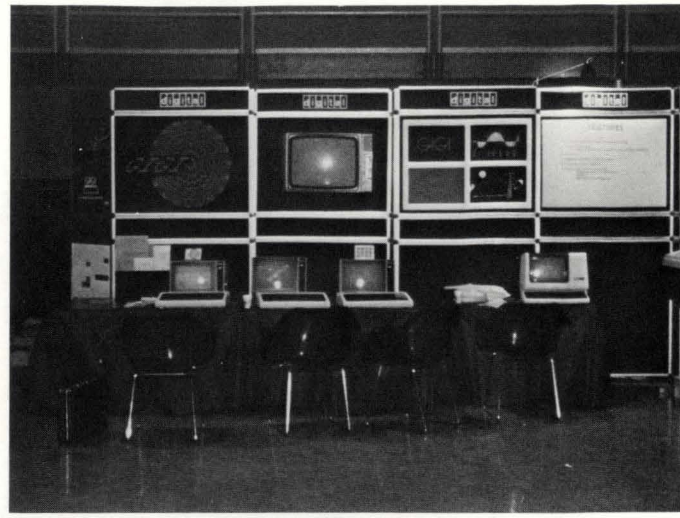
```
!---------------------------------------------!
!              D I R E C T . T E C            !
!                                             !
!          Simple directory program           !
!                                             !
!           Author: David Spencer             !
!                                             !
!---------------------------------------------!

!---------------------------------------------!
! This routine uses the following registers:  !
!                                             !
! 0s     Scratch, directory macro             !
! 0%     Scratch, file count                  !
! 1s     Scratch, complete wilcard spec       !
! 1%     "CCL entry" flag                     !
! 2s     Generate heading macro               !
! 2%     Old PPn in list                      !
! 3s     Print an account total               !
! 3%     New PPn in list                      !
! 4s     Last file listed                     !
! 4%     "Grand" total of files               !
! 5s     <cr> <lf> combination                !
! 5%     Number of accounts listed            !
! 6s     *NOT UTILIZED*                       !
! 6%     Total type flag                      !
!---------------------------------------------!

!---------------------------------------------!
! DIRECT.TEC is a TECO program written to take !
! directories of RSTS/E disks using only the  !
! normal "EN" support. It must be noted that  !
! the current version REQUIRES that the buffer !
! be clear of all data except the file spec   !
! to be used for the directory. Failure to    !
! clear the buffer before executing DIRECT will !
! certainly result in the loss of some data.  !
!                                             !
```

```
! Due only to the limitations of the data          !
! returned from the "EN" operator, DIRECT does     !
! a brief directory of only filename and           !
! protection code. Also, if the terminal is a      !
! scope, then DIRECT clears the screen before      !
! printing the directory.                          !
!                                                  !
! If DIRECT is running on version 7 of RSTS,       !
! you will find DIRECT will handle multiple        !
! accounts that occur with wildcard PPns.          !
!                                                  !
! DIRECT may be executed in one of three ways:     !
!                                                  !
! 1.    Via the CCL "FILES"                        !
!                                                  !
! 2.    By loading the file specification into     !
!       the text buffer and "EI DIRECT"            !
!                                                  !
! 3.    Loading the macro into a register and      !
!       executing it with an "Mreg-name".          !
!--------------------------------------------------!

! Set radix decimal, set counters                  !
©D [0 [1 [2 [3 [4 [5 [6 0U1 0U4 1U5 J 0X0

! Load Q5s with carriage-return, line-feed         !
@©U5%
%

! Load heading display macro                       !
@©U2%
@I/ Name .Ext  Protect    / G* @-S/]/ K G5 HT HK
%

! Load total display macro                         !
@©U3%
U6                          ! Get passed number    !
@I/
 Total of / Q0\ @I/ file/ Q0-1"N @I/s/ ' @I/ in /
Q6"= G1 | G4 @-S/]/ K ' G5 HT HK
%

! Remove possible CCL command                      !
::@S/FILES/"S ©SD -1U1 '

! Remove all spaces, tabs                          !
< @S/©ES/; ©SD >

! Insert file argument if none found               !
Z"E @I/*.*/ '

! Do some PPn work                                 !
J :@S/(/"S -D @I/[/ '
J :@S/)/"S -D @I/]/ '

! Strip off logical PPn's                          !
@©U1/$!%&@#/
:@S/©EG1/"S .-1,.X0 -D '

! Strip PPn so we can massage filspec              !
J :@S/[/"S R .U0 @S/]/ Q0,.X0 Q0-.D '

! Check for missing file specification             !
J :@S/:/"E @I/SY:/ ' ::@S/./ "S R @I/*/ ' .-Z "E @I/*/ '

! Check for missing extention                      !
J :@S/./"E ZL @I/.*/ | .-Z"E @I/*/ ' '

! Re-insert the PPn                                !
J @S/:/ G0

! Create EN command for directory search           !
J @I/EN/ ZJ @27I//

! Execute EN to expand filespec                    !
HX1 HK M1 G*

! Insert our PPn if one not found in spec          !
J :@S/[/"E :@S/:/"S @I/[/ 2EJ/256\ @I/,/ 2EJ&255\ @I/]/ ' '

! Fix PPn wildcards                                !
J :@S/[/"S \-255"= ©SD @I/*/ ' '
J :@S/,/"S \-255"= ©SD @I/*/ ' '

! Retreive the full filespec for display           !
HX1 HK

! Now create a macro to do the directory           !
@I#
:@EN//"= G5 @I/%No files matching specification / G1 G5 @O/END/
'
```

. . . continued on page 94

DECUS UK 1981 CONFERENCE

By Pauline Noakes, U.K. Correspondent



Peter Dick examines GIGI at Warwick

The DECUS UK Conference is an annual 4-day event held at a different University Campus each Spring. This year it was the turn of Warwick University in the Midlands to provide the venue, and 400-plus delegates poured in from all over Britain, together with the much-sought-after DEC speakers from the Digital development teams in the USA and a number of DEC personnel from the UK Education Services, Sales, Field Service, etc. divisions.

The site itself is a vast complex of lecture theatres, library, laboratories, residences, etc. which normally houses 3000 students. Therefore, accommodations here allowed all the attendees to stay on the site without the inconvenience and tedium of daily bus rides to and from the sessions.

Conference proceedings officially commenced at Monday lunchtime, enabling most of the delegates to travel from home on Monday morning. There were, however, two extra seminars, on real-time systems design and data communications, held on Monday morning for those who wished to spend a little extra time and money at Warwick, and consequently the influx of computer addicts started on Sunday.

After the traditional New Participants session and a DEC "New Products" session, SIG sessions began in earnest with all the roadmaps held on Monday evening.

For the RSTS/E group, the centre of interest was the sessions given by Jim Condict from the RSTS development team in Merrimac. Jim's first session outlined the development of the RSTS Monitor through the various releases and explained the way in which the current version works. He also spent some time in talking about the now famous "small buffer problem", why it exists and what work is being done to try to improve the situation. This session in particular was very well received as it cleared up some of the mysteries of life for the everyday RSTS user and provided food for thought for even the "RSTS hackers" in the audience!

The other two presentations by Jim Condict were (a) Terminal Service Internals and (b) Writing run-time systems and resident libraries for RSTS/E, both of which also attracted large audiences which packed the lecture theatre.

Contributions to the wish-list were discussed in a session which ran for an hour longer than scheduled and produced plenty of ideas for the RSTS developers to ponder over!

The exhibition hall was well attended at all times, with most interest being shown in the colour graphics on GIGI terminals and also in the VAX11/750 system. A competition was held to win a GIGI, the runner-up prize being the chance to buy a GIGI at half-price!! The accent on graphics was continued with the Conference Tutorial, an excellent presentation entitled "From today photography is dead", an entertaining demonstration of the impact of computing on graphics.

All in all, an excellent 4 days! Thanks to all those involved in organisation, to those who gave their time to prepare and present sessions and, of course, to DEC for all their co-operation in making the event a success. ♥

JBSTAT

By Francois Dubois, The City of Longueuil, Quebec, Canada

1. ABSTRACT

JBSTAT is a program which allows you to display the job status and open channel information for any job on a RSTS/E large file system. It is most useful to the system manager because it permits him to see what is going on with a job at any time. The programmers can use it for debugging purposes.

JBSTAT is written in Basic-Plus-2 and works on RSTS/E V7.0 "large files" system only.

2. OPERATION

Type in :
RUN (?,?)JBSTAT
or use the following CCL:
CCL J-BSTAT=(?,?)JBSTAT.TSK;PRIV 31500
Note: LOGIN can be modified to chain to line 32000.

JBSTAT then asks for the job number (or KB number) followed by one or many options.

Examples:

A) J/5-:R4:C4:C12

J	CCL FOR JBSTAT
/	"/" OR " " NEEDED IF OPTIONS SPECIFIED
5	INFORMATION ON JOB #5
-	DON'T DISPLAY THE STATUS OF JOB 5 "-" CAN APPEAR ANYWHERE AFTER "/"
:	USE TO SEPARATE SWITCHES
R4	REFRESH DISPLAY EVERY 4 SECONDS
C4	DISPLAY INFORMATION FOR CHANNEL #4
C12	DISPLAY INFORMATION FOR CHANNEL #12

B) J/12:S

S	DISPLAY ONLY THE STATUS FOR JOB #12

Refer to line 100 of the program to see other examples on how to use the switches.

3. MONITOR TABLES

Some monitor tables are described below to help you understand how JBSTAT gets to the desired information. Please refer to these while reading step #4.

The design of these tables come from the understanding of the SYSTAT program (which is well documented) and the reading of the TTDVR.LST file generated at SYSGEN time. That file contains a complete terminal DDB layout. Also, by studying the " ↑R and ↑ T handlers" code, one can understand how to get at the data stored in the keyboard's buffer and display it (JBSTAT displays the data in the KB's buffer and the typed ahead data, if any, for keyboards opened in mode 8).

4. HOW JBSTAT WORKS

After entering the job number (or KB #), JBSTAT first finds out if that job is active on the system. It does that by examining the job table. A 0 (zero) entry in the job table means an inactive job. A non-zero value is a pointer to the job data block for that job. -1 indicates the end of the job table.

If the job is found active, it will display the status like SYSTAT would do (except for the CPU time).

Then it peeks through the I/O block (pointed to be the first word of the JDB) for that job examining which channel is opened. The I/O block is 16 words long (one word per channel). Each word is a pointer to either a DDB (Device Data Block) or a WCB (Window Control Block). A zero value indicates that the channel is closed.

The first byte of a DDB or WCB is called the driver index and it's value determines which device is opened on that channel. A value of 0 indicates a disk device, 2 indicates a keyboard, etc... Also, if the driver index is zero, we are looking at a WCB, otherwise it is a DDB. The WCB is connected to the FCB by a pointer, stored in the fifth word of the WCB, which points to the file clustersize stored in the fifteenth word of the FCB (address of FCB + 34 octal).

Once JBSTAT reaches that point it is easy for it to extract from the DDB or the WCB and the FCB all the desired information.

NOTE: JBSTAT can easily be modified to display more information from the DDB, WCB or FCB.

5. DISPLAY

Refer to line 100 of the program for a complete description of the display. Refer to step #7 for examples.

6. NOTE

If you find any bugs in this program, please contact me. To receive this program on a 9-track (800 BPI or 1600 BPI) tape send a $30.00 U.S. Money Order payable to: The City of Longueuil.

Francois Dubois
The City of Longueuil
C.P. 5000
Longueuil, Quebec
Canada
J4K 4Y7
Tel: (514) 670-2220

7. EXAMPLES

```
--------------------------------------------------------------------------------
RSTS V7.0-07 LONGUEUIL JBSTAT 07-Apr-81 13:58

Job        Who      Where   What      Size     State     Rts       Prio/Brst

16       [  1,105]  KB19   DUMPTO    16/31K    DB        BASIC       -8/6

Chn Dev     PPn     File Name    Size Flags Prot Clu   Op/Rr      User     Block

 1. MM0:Density: 1600 BPI, Status: Write End-of-File
 2. DB0:[ 95,1   ]DSD14B.TSK       35  C   <124>  4    0/1       RdRr        35
 3. DB0:[  1,105]BAC953.LST         7      < 60>  4    1/0       RdWr         7
11. DB0:[  1,105]DUMP16.TMP         1  D   < 60>  4    1/0       RdWr         1
12. NL:
--------------------------------------------------------------------------------
RSTS V7.0-07 LONGUEUIL JBSTAT 07-Apr-81 14:01

Job        Who      Where   What      Size     State     Rts       Prio/Brst

14 +     [ 75,91 ]  KB17   IPA001    15/31K    RN        ...RSX      1/12

Chn Dev     PPn     File Name    Size Flags Prot Clu   Op/Rr      User     Block

 1. DB1:[ 75,2   ]FPAMAS.IND     3696  U   < 42> 128   1/0     RdWrCaUp     373
 2. KB17:Mode: CrLf Echo Field: Kpch 0 DH11[12]
         Type ahead: ¶<cr><cr>dubois francois<cr>¶
 4. DB1:[ 75,2   ]FPAREF.IND       82  U   < 42>  4    1/0     RdWrCaUp      13
15. DB0:[ 75,1   ]IPA001.TSK       58  C   <232>  4    0/1       RdRr        38

--------------------------------------------------------------------------------
RSTS V7.0-07 LONGUEUIL JBSTAT 07-Apr-81 14:18

Job        Who      Where   What      Size     State     Rts       Prio/Brst

 4       [  1,3  ]  Det    SPLRUN    16/31K    SL        BASIC       -8/6

Chn Dev     PPn     File Name    Size Flags Prot Clu   Op/Rr      User     Block

 1. DB0:[  1,3  ]SPL004.WRK         9      < 60>  4    1/0       RdWr         5
 2. LP0: Status: Hung
 3. NL:
 4. DB0:[  1,3  ]CHARS .QUE         2      < 60>  4    1/0        Rd          0
--------------------------------------------------------------------------------
RSTS V7.0-07 LONGUEUIL JBSTAT 07-Apr-81 14:25

Job        Who      Where   What      Size     State     Rts       Prio/Brst

18       [***,***]  KB11   LOGIN     14/31K    KB        ...RSX      0/6

Chn Dev     PPn     File Name    Size Flags Prot Clu   Op/Rr      User     Block

 1. KB11:DH11[6]Status: Noecho

--------------------------------------------------------------------------------
```

SOME MONITOR TABLES

```
     JOB TABLE               JOB DATA BLOCK             I/O BLOCK
:--------:--------:        :--------:--------:      :--------:--------:
:                0:    .-->: POINTER TO IOB 0:----->: CHAN. #0       0:----.
:--------:--------:        :--------:--------:      :--------:--------:    :
: POINTER TO JDB 2:---'    :      JDFLG     2:  .---: CHAN. #1       2:    POINTER
:--------:--------:        :--------:--------:  :   :--------:--------:    TO
:                4:        :                4:  :   : CHAN. #2       4:    USER'S
:--------:--------:        :--------:--------:  :   :--------:--------:    KB DDB
:                6:        :                6:  :   : CHAN. #3       6:    :
:--------:--------:        :--------:--------:  :   :--------:--------:    :
:               10:    .---:      JDJDB2   10:  :   : CHAN. #4      10:    :
:--------:--------:    :   :--------:--------:  :   :--------:--------:    :
\                 \    :   \                 \  :   \                 \    :
/                 /    :   /                 /  :   /                 /    :
:--------:--------:    :   :--------:--------:  :   :--------:--------:    :
:       -1        :    :   :                 :  :   : CHAN. #15     36:    :
:--------:--------:    :   :--------:--------:  :   :--------:--------:    :
                       :                        :                          :
 JOB DATA BLOCK 2      :          WCB           :          KB DDB          :
:--------:--------:    :   :--------:--------:  :   :--------:--------:    :
:                0:<--'    :STS FLG :  IDX   :<-'   : STATUS :  IDX  0:<---'
:--------:--------:        :--------:--------:      :--------:--------:
\                 \        :FLG BITS:JB # *2 :      :DV UNIT#:JB #*2 2:
/                 /        :--------:--------:      :--------:--------:
:--------:--------:        :        :        :      :                4:
:     J2NAME   14:         :--------:--------:      :--------:--------:
:--------:--------:        :                 :      :                6:
:               16:        :--------:--------:      :--------:--------:
:--------:--------:        : -> FCB @ CLUS   :--.   :       FLAGS   10:
\                 \        :--------:--------:  :   :--------:--------:
/                 /        :                 :  :   :OUT BUF. EP    12:
:--------:--------:        :--------:--------:  :   :--------:--------:
:      J2PPN   30:         :NXT WCB->FCB +FLG:  :   :OUT BUF. FP    14:
:--------:--------:        :--------:--------:  :   :--------:--------:
\                 \        \                 \  :   :OUT BUF. COUNT 16:
/                 /        /                 /  :   :--------:--------:
:--------:--------:        :--------:--------:  :   :        :      20:
                                                :   :--------:--------:
                                                :   :IN  BUF. EP    22:
                                    FCB         :   :--------:--------:
                           :--------:--------:  :   :IN  BUF. FP    24:
                           :                0:  :   :--------:--------:
                           :--------:--------:  :   :IN  BUF. COUNT 26:
                           \                 \  :   :--------:--------:
                           /                 /  :   :        :      30:
                           :--------:--------:  :   :--------:--------:
                           : CLUSTER SIZE  34:<-'   :        : MODE 32:
                           :--------:--------:      :--------:--------:
                           : FIRST  WCB    36:      :CHARACTERISTICS34:
                           :--------:--------:      :--------:--------:
                                                    :INTERFACE FLAG 36:
                                                    :--------:--------:
                                                    :ECHO EP (AHEAD)40:
                                                    :--------:--------:
                                                    :ECHO FP (AHEAD)42:
                                                    :--------:--------:
                                                    :ECHO BC (AHEAD)44:
                                                    :--------:--------:
                                                    :PAINT CH:FLD MD46:
                                                    :--------:--------:
```

```
               WCB
      WINDOW CONTROL BLOCK

:--------------:--------------:
:   W$STS    1:     W$IDX   0:
: STATUS FLAGS :  0 FOR DISK  :
:--------------:--------------:
:   W$FLAG   3:    W$JBNO   2:
:  FLAG BITS   :  JOB # * 2   :
:--------------:--------------:
:   W$NVBM   5:     W$PT    4:
:NEXT VBN (MSB):PENDING XFERS :
:--------------:--------------:
:            W$NVBL          6:
:        NEXT VBN (LSB)       :
:-----------------------------:
:            W$FCB          10:
:        -> FCB @ F$CLUS      :
:-----------------------------:
:            W$REN          12:
:    RETRIEVAL ENTRY NUMBER   :
:-----------------------------:
:            W$WCB          14:
: -> NEXT WCB THIS FCB + FLAGS:
:-----------------------------:
:            W$NXT          16:
:         FBB OF NEXT         :
:-----------------------------:
:           WINDOW            :
:                             :
:-----------------------------:
:            W$WND          22:
:                             :
:-----------------------------:
:                           24:
:              W              :
:-----------------------------:
:                           26:
:              I              :
:-----------------------------:
:                           30:
:              N              :
:-----------------------------:
:                           32:
:              D              :
:-----------------------------:
:                           34:
:              O              :
:-----------------------------:
:                           36:
:              W              :
:-----------------------------:
```

```
               FCB
       FILE CONTROL BLOCK

:------------------------------:
:            F$LINK           0:
:POINTER TO NEXT FCB THIS UNIT:
:------------------------------:
:            F$FID            2:
:FILE ID (LINK TO NAME ENTRY) :
:------------------------------:
:            F$PPN            4:
:         PPN OF FILE          :
:------------------------------:
:            F$NAM            6:
:                              :
:------------------------------:
:                            10:
:    FILE NAME & EXTENSION     :
:------------------------------:
:                            12:
:                              :
:--------------:---------------:
:   F$PROT  15:     F$STAT  14:
:  PROT. CODE  : STATUS BYTE   :
:--------------:---------------:
:   F$RCNT  17:     F$ACNT  16:
:R/R OPEN COUNT:  OPEN COUNT   :
:--------------:---------------:
:            F$WFND          20:
:         FBB OF FIRST         :
:------------------------------:
:       RETRIEVAL ENTRY      22:
:                              :
:------------------------------:
:            F$UFND          24:
:             FBB              :
:------------------------------:
:        OF NAME ENTRY       26:
:                              :
:--------------:---------------:
:   F$SIZM  31:     F$UNT   30:
:FILE SIZE MSB :  FIP UNIT #   :
:--------------:---------------:
:            F$SIZL          32:
:        FILE SIZE LSB         :
:------------------------------:
:            F$CLUS          34:
:         CLUSTER SIZE         :
:------------------------------:
:            F$WCB           36:
:  -> FIRST WCB FOR THIS FILE  :
:------------------------------:
```

BITS IN F$STAT OF FCB

```
    7      6      5      4      3      2      1      0
:------:------:------:------:------:------:------:------:
:US.DEL:US.UFD:US.NOK:US.NOX:US.UPD:US.WRT:US.PLC:US.OUT:
:MARKED:FILE  :DO NOT:CONTIG:OPEN  :WRITE :FILE  :OUT OF:
:DELETE:IS UFD:KILL  :FILE  :UPDATE:GIVEN :PLACED:SAT   :
:------:------:------:------:------:------:------:------:
```

BITS IN DDSTS OF WCB

```
   15      14      13      12      11      10       9       8
:------:------:------:------:------:------:------:------:
:WC$USE:WC$UFD:WC$LCK:WC$CTG:WC$UPD:DDWLO :DDRLO :DDNFS :
:WRITE :FILE  :LOCK  :CONTIG:UPDATE:WRITE :READ  :N.F.S.:
:PRIVS :IS UFD:IS ON :FILE  :MODE  :LOCK  :LOCK  :      :
:------:------:------:------:------:------:------:------:
```

BITS IN WSFLAG OF WCB

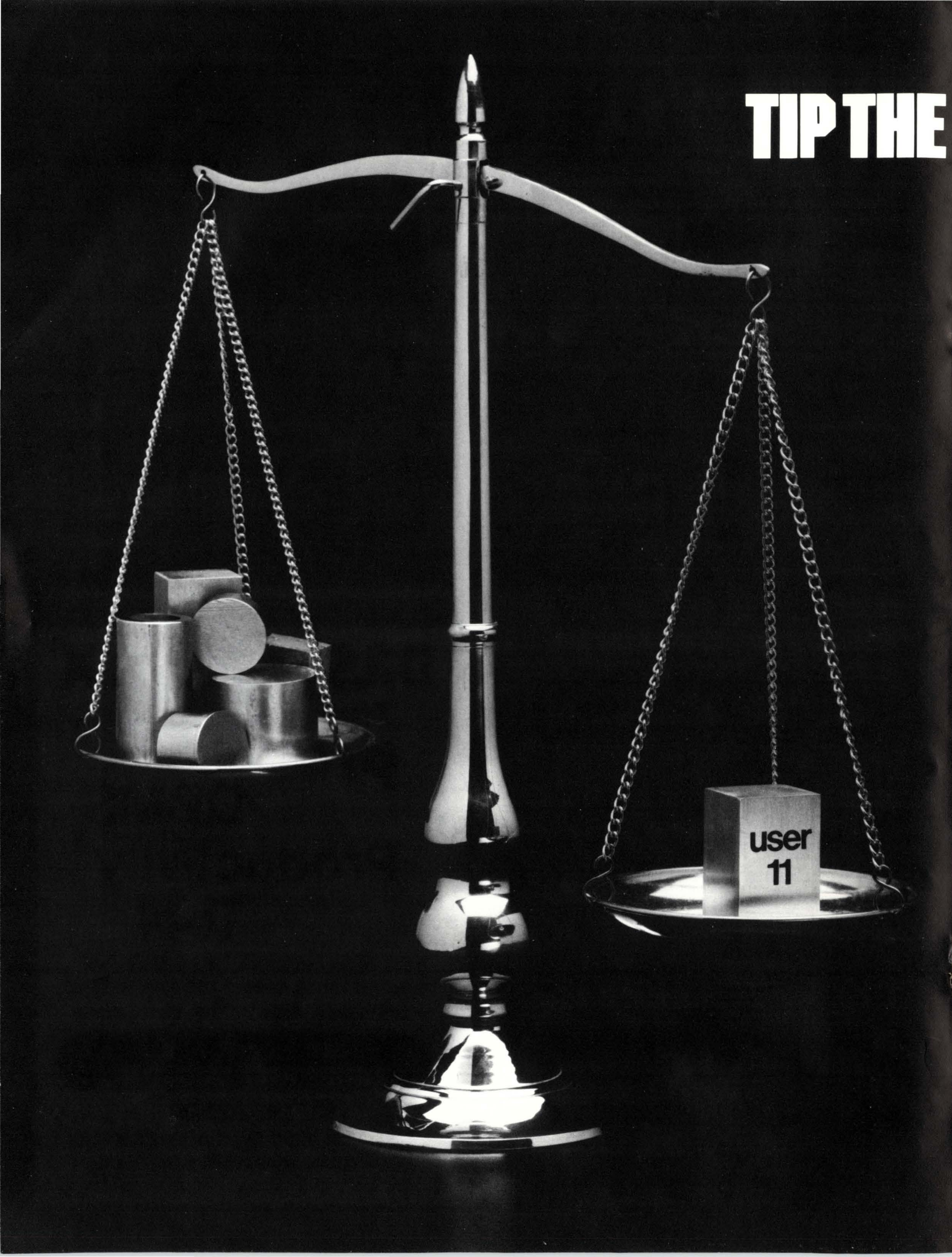
```
   15      14      13      12      11      10       9       8
:------:------:------:------:------:------:------:------:
:WC$NFC:WC$DLW:WC$EXT:              WC$LLK               :
:NFS BY:FIX SZ:EXTN'D:        LENGTH OF CURRENT          :
:CLUSTR:& DATE:WCB   :          IMPLICIT LOCK            :
:------:------:------:------:------:------:------:------:
```

BITS IN W$WCB OF WCB   BITS IN W$WCB OF WCB (CONTINUED)

```
  15     14     13     12     11     10      9      8      7      6      5      4      3      2      1      0
:------:------:------:------:------:------:------:------:------:------:------:------:------:------:------:------:
:                         ADDRESS OF NEXT WCB                                 :WC$CSQ:WC$CHE:WC$AEX:WC$SPU:WC$RR :
:                                                                             :SEQ'L :USER  :ALWAYS:SPEC'L:READ  :
:                                                                             :CACHE :CACHE :EXTEND:UPDATE:RDRGLS:
:------:------:------:------:------:------:------:------:------:------:------:------:------:------:------:------:
```

THE ADDRESSES GIVEN BELOW ARE OCTAL NUMBERS

```
JOB TABLE
---------

2 TO END          POINTER TO JOB DATA BLOCK FOR EACH JOB
      NOTE:       A POINTER VALUE OF -1 INDICATES THE END OF JOB TABLE

JOB DATA BLOCK
--------------

0                 POINTER TO I/O BLOCK
2 (JDFLG)         JOB STATUS FLAGS                                   FIG-1
10 (JDJDB2)       POINTER TO JOB DATA BLOCK EXTENSION

I/O BLOCK
---------

0                 POINTER TO USER'S KB DDB
2 TO 36           POINTER TO EITHER A DDB, SCB OR WCB

JOB DATA BLOCK 2
----------------

14 (J2NAME)       FIRST 3 CHARACTERS OF JOB NAME (RAD50)
16                LAST  3 CHARACTERS OF JOB NAME (RAD50)
30 (J2PPN)        ACCOUNT NUMBER

WCB
---

0                 DRIVER INDEX
1                 STATUS BITS FOR FILE
2                 JOB NUMBER * 2
10                POINTER TO FILE CLUSTERSIZE STORED IN FCB
14                POINTER TO THE NEXT WCB FOR THIS FILE AND
                  OTHER FLAGS

FCB
---

0                 POINTER TO THE NEXT FCB FOR THIS UNIT
4                 ACCOUNT NUMBER OF FILE
6                 FIRST 3 CHARACTERS OF FILE NAME (RAD50)
8                 LAST  3 CHARACTERS OF FILE NAME (RAD50)
10                EXTENSION (RAD50)
14                STATUS BYTE
15                PROTECTION CODE
16                NUMBER OF TIMES THIS FILE IS OPEN (NOT REGARDLESS)
17                NUMBER OF TIMES THIS FILE IS OPEN READ REGARDLESS
30                FIP UNIT NUMBER
31                FILE SIZE (MSB)
32                FILE SIZE (LSB)
34                FILE CLUSTERSIZE
36                POINTER TO FIRST WCB FOR THIS FILE
```


TIP THE
user
11

```
KB DDB
------

0     DRIVER INDEX
1     STATUS BYTE                                          FIG-2
2     OWNER JOB NUMBER *2
3     DEVICE UNIT NUMBER
10    FLAGS                                                FIG-3
12    EMPTY POINTER FOR OUTPUT
14    FILL  POINTER FOR OUTPUT
16    BUFFER COUNT  FOR OUTPUT
22    EMPTY POINTER FOR INPUT
24    FILL  POINTER FOR INPUT
26    BUFFER COUNT  FOR INPUT
32    OPEN MODE FOR KB                                     FIG-4
34    TERMINAL'S CHARACTERISTICS
36    INTERFACE FLAG                                       FIG-5
40    EMPTY POINTER FOR TYPE AHEAD (ECHO CONTROL)
42    FILL  POINTER FOR TYPE AHEAD (ECHO CONTROL)
44    BUFFER COUNT  FOR TYPE AHEAD (ECHO CONTROL)
46    ECHO CONTROL FIELD SIZE AND MODE                     FIG-6
47    PAINT CHARACTER AND FIELD ACTIVE FLAG                FIG-6
```

```
                2048        256
:---:---:---:---:---:---:---:---:---:---:---:---:---:---:---:---:
:                *           * :                                :
:---:---:---:---:---:---:---:---:---:---:---:---:---:---:---:---:
                 :           :
                 :           '---> TEMPORARY PRIVILEGES POSSIBLE
                 :
                 '--------------> TEMPORARY PRIVILEGES ACTIVE
```

FIG-1 (JOB STATUS FLAGS IN JDB)

```
32768
:---:---:---:---:---:---:---:---:---:---:---:---:---:---:---:---:
: *                              :                              :
:---:---:---:---:---:---:---:---:---:---:---:---:---:---:---:---:
  :
  :
  :
  '---------->CONTROL/O STATE (FOR KEYBOARD)
              PRINTER HUNG    (FOR PRINTER )
```

FIG-2 (STATUS BYTE IN KB DDB)

```
32768                                          32
:---:---:---:---:---:---:---:---:---:---:---:---:---:---:---:---:
: *                              :              *               :
:---:---:---:---:---:---:---:---:---:---:---:---:---:---:---:---:
  :                                             :
  :                                             :
  :                                             :
  '---------->CONTROL/S (STATE)                 '-----> NOT ECHOING
```

FIG-3 (FLAGS IN KB DDB)

```
                                   128  64  32  16   8   4   2   1
:---:---:---:---:---:---:---:---:---:---:---:---:---:---:---:---:
:                              : *   *   *   *   *   *   *   * :
:---:---:---:---:---:---:---:---:---:---:---:---:---:---:---:---:
                                 :   :   :   :   :   :   :   :
                                 :   :   :   :   :   :   :   :
   SPECIAL SCOPE RUBOUT       <---'   :   :   :   :   :   :   :
   RESERVED                   <-------'   :   :   :   :   :   :
   ENABLES INCOMING XON/XOFF  <-----------'   :   :   :   :   :
   GUARDED TERMINAL MODE      <---------------'   :   :   :   :
   ECHO CONTROL MODE          <-------------------'   :   :   :
   NO AUTO CR/LF MODE         <-----------------------'   :   :
   TECO MODE                  <---------------------------'   :
   BINARY MODE                <-------------------------------'
```

FIG-4 (OPEN MODE FOR KB IN KB DDB)

```
                 20  10
                  48  24 512 256 128  64  32  16   8   4   2   1
:---:---:---:---:---:---:---:---:---:---:---:---:---:---:---:---:
:                 *   *   *   * : *   *   *   *   *   *   *   * :
:---:---:---:---:---:---:---:---:---:---:---:---:---:---:---:---:
                :               :                               :
                '------. .------'------------. .----------------'
                       :                     :
                 SUB-LINE NUMBER     INTERFACE TYPE CODE:

                                          0 KL11
                                          2 DC11
                                          4 DL11
                                          6 DE11
                                          8 PK11
                                         10 DJ11
                                         12 DH11
                                         14 DZ11
```

FIG-5 (INTERFACE FLAG IN KB DDB)

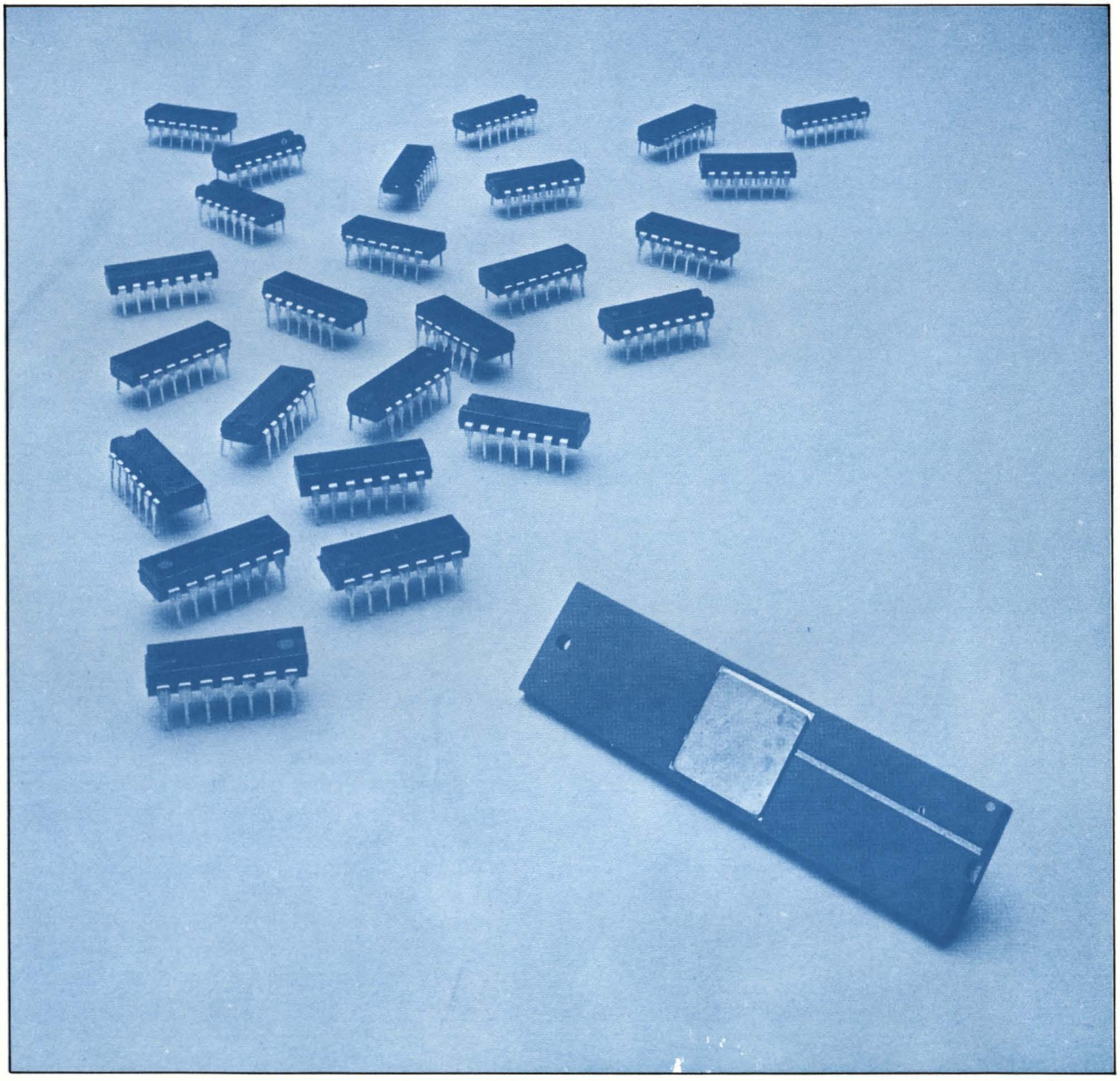
```
 32  16  81  40  20  10
768 384  92  96  48  24 512 256 128  64  32  16   8   4   2   1
:---:---:---:---:---:---:---:---:---:---:---:---:---:---:---:---:
: *   *   *   *   *   *   *   * : O   *   *   *   *   *   *   * :
:---:---:---:---:---:---:---:---:---:---:---:---:---:---:---:---:
  : :                           : : :                           :
  : '-----------. .-------------' : '-----------. .-------------'
  :             :                 :             :
  :       PAINT CHARACTER         :         FIELD SIZE
  :                               :
  '------> FIELD IS ACTIVE        '--------> KEYPUNCH MODE (OFF)
```

FIG-6 (ECHO CONTR. INFO. IN KB DDB)

. . . continued on page 58

The VAX-SCENE

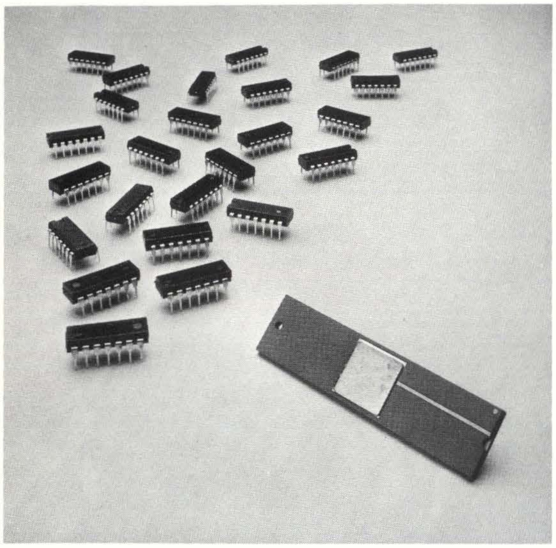
Number 3 (RSTS PROFESSIONAL, Vol. 3, No. 2) June 1981

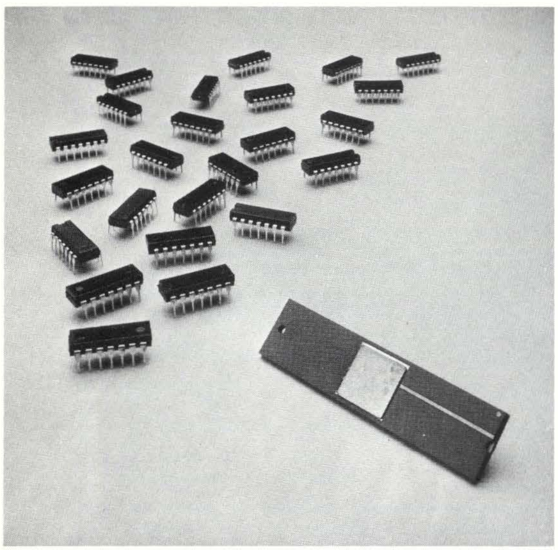


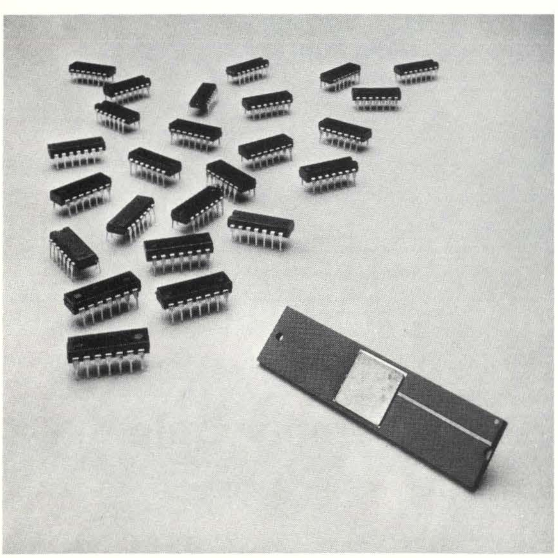
INSIDE:

☐ CONPAX: Conversion of PDP-11 Assembly Code to VAX-11 Native Mode

☐ VAX News







CONPAX: Conversion of PDP-11 Assembly Code to VAX-11 Native Mode

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

PSI is a software vendor specializing in software products for the DEC PDP-11 and VAX-11 computers. In late 1980, we found ourselves faced with the task of converting two of our products from the PDP-11 to VAX-11 native mode. The products were VIDIO/11, a CRT-independent video terminal I/O subroutine library, and SCRNIO/11, a subroutine library and screen form editor utility for CRT-independent screen form use and management, The two products together contained about 7,000 lines of Macro-11, which would have to be converted to Macro-32.

We considered the alternatives that presented themselves. We could recode the assembly code by hand; we could rewrite the assembly code sections in a higher language; or we could attempt some kind of automatic conversion. Our conversion schedule made the first two options unattractive, so we looked more closely at the possibility of automating such a conversion.

It was quickly apparent that a fully automatic conversion was not possible; there would be too many cases where human judgment would have to be injected into the process. However, the similarity of both the VAX-11 architecture and its assembler to the PDP-11 encouraged the development of a conversion tool to automate as much as possible. Our next question was whether it would be possible to design and implement such a tool to be of general use, beyond our immediate need. Our analysis was positive; the result was CONPAX (CONvert Pdp-11 to vAX).

Using a first version of CONPAX, the 7,000 lines of assembly code were converted and debugged in about 10 man-days. This is perhaps not broadly representative, since the conversion was done by two highly experienced systems programmers on code which was cleanly structured and thoroughly documented. However, subsequent experience on unfamiliar and less documented code has underscored the usefulness of the tool for conversion. We have been gratified to find that CONPAX not only performed very well on the specific conversion that prompted its development; it is proving to be highly successful on other conversions, and in fact is in use at several large DEC customer sites.

The reasons for such conversions are several. "Compatibility mode" on the VAX-11 is limited both in its performance (running at about half the speed of "native mode") and in its 64KB address space. In addition, all components of a task must be in compatibility mode, ruling out tne use of the VAX's native mode compilers and run-time services. CONPAX, by making possible a rapid, clean conversion to native mode, opens up the full power of the VAX-11 for existing PDP-11 applications containing assembly code.

In the conversion process, CONPAX uses addressing modes and other information from the PDP-11 assembler code to produce the VAX-11 conversion for each line, including multi-line alternatives when appropriate. Simple editing procedures and a post-processor allow the user to select the desired conversion and eliminate unwanted alternatives. During this refinement process, which can be iterative if necessary, VAX-11 assemblies are always possible.

CONPAX automatically handles details such as supplying octal radix indicators, translating ASCII literals to VAX-11 format, and substituting standard VAX-11 register names. The spacing of input lines is preserved or, optionally, aligned to the user's specification.

The CONPAX conversion process is controlled by tables which can be used to "fine tune" the conversion for particular programming environments. For instance, the conversion can be biased toward either 16 or 32 bit arithmetic. Special substitutions for names or argument strings can be accommodated. Substitution for any given symbol can be limited to a particular field of the input line — label, opcode, operand, or comment. Normally, conversion produces VAX-11 instructions, not macros. However, the control tables can be used to produce special conversions (including multi-line) of PDP-11 opcodes, macros, or operands.

CONPAX is available under permanent license or as a service from PSI. As a service, an initial $1,500 fee includes initial consultation, training, and conversion of 500 lines of source code. Additional conversion costs $.50 per line. A permanent license costs $5,000 for one user site (either PDP-11 or VAX-11 version), including installation, initial conversion consultation, and training. Additional sites can be licensed at substantial discounts. Consulting and conversion assistance are also available from PSI on a per diem basis.

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

CONPAX Conversion Example

This example of CONPAX conversion consists of a PDP-11 assembler subroutine and the resulting VAX-11 source code. This is immediately followed by notes which apply to specific lines of the source code.

Except as indicated, the conversion of each line from PDP-11 to VAX-11 code was automatic: the first (or only) alternative produced by CONPAX was the final output. No user action was required in these cases. Otherwise:

(C) marks lines where the user chose a secondary alternative. The choice was made by entering a one-character code.

(D) marks lines where the choice of a secondary alternative was deferred until after a trial assembly. The choice was then made by entering a one- or two-character code. Note that the conversion tables can be set up to eliminate the need for trail assembly of the converted VAX source; the trade-off is slightly less space-efficient VAX code.

(M) marks lines where manual conversion was necessary.

The CONPAX conversion process is controlled by tables. The particular tables used for the conversion shown here were designed to maintain the existing size (8 or 16 bits) of data in memory. However, data in registers and on the stack are handled as 32-bit quantities, since experience has shown that this is most generally effective. And, of course, addresses change from 16 bits on the PDP-11 to 32 bits on the VAX-11, whether they are stored in memory, in registers, or on the stack.

The notes given below further reflect the specific conversion tables used, as well as the general action of CONPAX.


```
                CONPAX Conversion Example:  Macro-11 Subroutine
Notes                              Macro-11 Source
-----                              ---------------
        ; ERMSG -- CONSTRUCT ERROR MESSAGE
        ; CALL:
        ;       CALL ERMSG(BUF,BUFLEN,ERRCOD,ERRTXT,ST)
        ; WHERE:
        ;       BUF     = BUFFER TO RECEIVE MESSAGE (ASCII ENDED WITH NULL)
        ;       BUFLEN  = LENGTH, IN BYTES, OF BUFFER
        ;       ERRCOD  = ENCODED ERROR NUMBER AND FLAGS
        ;       ERRTXT  = MESSAGE TEXT (ASCII ENDED WITH NULL)
        ;       ST      = SET TO STATUS:  0 = OK, -1 = BUF TOO SHORT
        ;
                .TITLE  ERMSG
        ;
   (N1)         .PSECT  INSTR,RO,I,LCL,REL,CON
        ;
   (N2)         .GLOBL  MOVST           ;"MOVE STRING" UTILITY ROUTINE
                .GLOBL  MOVCH,MOVCH2    ;"MOVE CHARACTER" UTILITY ROUTINES
                . . .
        ;
   (N3) ERMSG:: CLR     FLGS            ;INIT LOCAL FLAGS
   (N4)         TST     (R5)+           ;SKIP ARGUMENT COUNT
   (N5)         MOV     (R5)+,T0        ;BUFFER ADDRESS
   (N6)         MOV     @(R5)+,T1       ;BUFFER LENGTH
(C)(N7)         MOV     #TXTE,T2        ;ADDR OF INITIAL TEXT
   (N8)         JSR     PC,MOVST        ;MOVE TO BUFFER, UPDATE PTR + COUNT
(D)(N9)         BCS     XBUFLN          ;ERROR, BUFFER TOO SHORT
                . . .
   (NA)         MOVB    #'.,(T0)+       ;ADD PERIOD TO BUFFER
   (NB)         MOV     #"//,T2         ;SEPARATE NUMERIC CODES
                JSR     PC,MOVCH2
                . . .
(C)(NC)         MOVB    1(T2),T2        ;GET CONTROL COUNT
(C)(NC)         BIC     #↑C377,T2       ;CLEAN IT UP
                . . .
(M)(ND)         GTIM$   (T2)            ;GET DATE
                . . .
   (NE)         ASH     #5,T2           ;GET SECONDARY FLAGS
   (NF)         BIC     #↑C37,T2        ;CLEAN THEM UP
```

```
                . . .
                RTS     PC                      ;RETURN TO CALLER
                . . .
   (N1)         .PSECT  CONST,RO,D,LCL,REL,CON
        TXTE:   .ASCIZ  /? ERROR /
                . . .
                .END
```

CONPAX Conversion Example: Resulting Macro-32 Subroutine
VAX-11 Source

```
Notes
------
        ; ERMSG -- CONSTRUCT ERROR MESSAGE
        ; CALL:
        ;       CALL ERMSG(BUF,BUFLEN,ERRCOD,ERRTXT,ST)
        ; WHERE:
        ;       BUF     = BUFFER TO RECEIVE MESSAGE (ASCII ENDED WITH NULL)
        ;       BUFLEN  = LENGTH, IN BYTES, OF BUFFER
        ;       ERRCOD  = ENCODED ERROR NUMBER AND FLAGS
        ;       ERRTXT  = MESSAGE TEXT (ASCII ENDED WITH NULL)
        ;       ST      = SET TO STATUS: 0 = OK, -1 = BUF TOO SHORT
        ;
                .TITLE  ERMSG
        ;
   (N1)         .PSECT  INSTR,RD,NOWRT,EXE,LCL,REL,CON
        ;
   (N2)         .EXTRN  MOVST                   ;"MOVE STRING" UTILITY ROUTINE
                .EXTRN  MOVCH,MOVCH2            ;"MOVE CHARACTER" UTILITY ROUTINES
                . . .
        ;
   (N3)         ENTRY ERMSG,M½R2,R3,R4,R5,R6,R7,R8½
   (N3)         CLRW    FLGS                    ;INIT LOCAL FLAGS
   (N4)         TSTL    (AP)+                   ;SKIP ARGUMENT COUNT
   (N5)         MOVL    (AR)+,R2                ;BUFFER ADDRESS
   (N6)         CVTWL   @(AP)+,R3               ;BUFFER LENGTH
(C)(N7)         MOVL    #TXTE,R4                ;ADDR OF INITIAL TEXT
   (N8)         JSB     MOVST                   ;MOVE TO BUFFER, UPDATE PTR + COUNT
(D)(N9)         BCC     20001$
(D)             BRW     XBUFLN                  ;ERROR, BUFFER TOO SHORT
(D)     20001$:
                . . .
   (NA)         MOVB    #↑A/./.(R2)+            ;ADD PERIOD TO BUFFER
   (NB)         CVTWL   #↑A*//\,R4              ;SEPARATE NUMERIC CODES
                JSB     MOVCH2
                . . .
(C)(NC)         MOVZBL  1(R4),R4                ;GET CONTROL COUNT
                . . .
(M)(ND)         $GETTIM (R4)                    ;GET DATE
                . . .
   (NE)         ASHL    #5,R4,R4                ;GET SECONDARY FLAGS
   (NF)         BICL2   #↑C↑O37,R4              ;CLEAN THEM UP
                . . .
                RET                             ;RETURN TO CALLER
                . . .
   (N1)         .PSECT  CONST,RD,NOWRT,NOEXE,LCL,REL,CON
        TXTE:   .ASCIZ  /? ERROR /
                . . .
                .END
```

Notes

(N1) Appropriate substitutions were made for the .PSECT arguments which differ from PDP-11 to VAX-11. These arguments are changed only if they are .PSECT operands; for instance, if "I" appeared elsewhere, it would not be changed.
(N2) .GLOBL was changed to the more specific .EXTRN.
(N3) The double colon produced the "ENTRY", and the instruction moved down to a new line.
(N4) Note that R5 was converted to AP. The alternative with "R5" would have been chosen if the register were not being used as an argument list pointer. The conversion produces long-word instructions as the default for AP-relative values, since such values are usually addresses.
(N5) The VAX-11 assembler does not allow symbolic names for registers. CONPAX substituted explicit register names as specified by the user through the control tables.
(N6) The 16-bit value was extended to 32 bits since this is often useful, and rarely undesirable, for values in registers.
(N7) The MOVL alternative was chosen since the quantity moved is an address.
(N8) The "PC," was automatically deleted and an extra tab inserted to keep the comment properly aligned.
(N9) Conditional branches do not reach as far on the VAX-11 since destination offsets are byte-relative rather than word-relative. This alternative was chosen after preliminary assembly on the VAX showed that a simple branch (the default alternative) would not reach the destination. CONPAX generated the selected three statements as a single alternative, which was chosen with a single one-character command. CONPAX generates local labels as necessary, from a user-specified starting number.
(NA) "ADD" is not converted unless it appears in the instruction field.
(NB) The single-quote (one-character) and double-quote (two-character) ASCII literals allowed by the PDP-11 assembler are not permitted by the VAX assembler. CONPAX automatically converts these to the delimited "↑A" construction. Note that CONPAX picks a delimiter which isn't in the literal.
(NC) The "BIC" instruction was deleted because the "MOVZBL" on the VAX made it unnecessary. The deletion was done with a one-character command.
(ND) This required manual recoding, since the directive on the VAX is different from the one on the PDP-11.
(NE) CONPAX automatically doubled the destination operand, as required for the VAX "ASHL" instruction.
(NF) Since the VAX-11 doesn't allow setting octal as the default radix, CONPAX automatically applies "↑O" to octal numbers unless they're one digit.

VAX NEWS

VAX/VMS INCREMENTAL FILE BACKUP AND RESTORE SOFTWARE

RAXCO, Inc., West Palm Beach, Florida, announces the immediate availability of RABBIT-5, Incremental File Backup and Restore (IFBR), a new VAX/VMS software utility system that provides incremental backup and restore capability for data files. RABBIT-5 will save only the files selected from disk by writing them to a specified tape. It will then discretely restore specified files on request.

RABBIT-5 IFBR is designed to provide maximum efficiency for both the user and the computer system when saving and restoring data files. Files are saved and restored approaching tape passing speed.

The user may select file-names, extensions and versions to be saved and/or restored. "Wild card" operations are permissible. Creation date and update, with greater-than/less-than arguments may be used. Last file versions may be selected.

RABBIT-5 operates in a "novice" or "expert" mode. In the novice mode, a prompter requests "fill-in-the-blanks" information to save or retrieve files. In the expert mode, all system commands plus wild carding is permissible. However, the prompter is disenabled for efficiency.

While RABBIT-5 IFBR may be used by anyone, it is particularly useful to system management and operational personnel as it offers full and partial disk backup and restore capability. The system may be operated interactively or may "wake-up" and produce backup files automatically. Long jobs may utilize RABBIT-5 for checkpoint and restart procedures.

Some capabilities of RABBIT-5 include:

- ability to save/restore files by date.
- ability to save/restore file types with wild card notation.
- ability to save/restore files based on actual length in blocks to reduce data storage.
- ability to exclude/include files when saving or restoring.
- ability to produce a directory of file names.
- HELP command.

RABBIT-5 is written in FORTRAN-IV and macro assembler, and will operate under VMS version 2 and above. RABBIT-5 may be purchased for $3750 or rented for $149/month.

RABBIT-5 is the newest member of the VAX/VMS RABBIT Family which includes RABBIT-1 Resource Accounting and Billing; and RABBIT-2 System Performance Analysis.

RAXCO, Inc., also announces the immediate availability of Version 3.0 of RABBIT-1, a Resource Accounting and Billing System for VAX/VMS users.

Major improvements include a new menu selection approach, more efficient processing, plus several new optional features.

The RABBIT-1 menu allows the user to select from more than 10 categories of major transactions to be effected. Transactions include:

1. File Creation of Users
2. File Management
3. Name and Address Modifications
4. Report Specifications
5. Rates and Discounts
6. Disk File Management
7. Run Disk Job
8. Run Editor
9. Run Update
10. Run Reports

The RABBIT System may be run interactively, serially, step by step, or batch. Resource summaries, billing recaps, cross footing and balancing are automatic features.

RABBIT-1 options include:

- User Budget System
- Program Accounting System
- Communications Post Accounting System
- Project Code Accounting System
- Disk Accounting System
- Tape Storage System
- Report Generator

RABBIT-1 is written in FORTRAN IV and runs in VAX native mode. RABBIT-1 base price is $2495. Options are available for $250 each. Rentals are $99/month and $12.50/month respectively.

RAXCO develops and markets a full line of DEC software systems and a complete line of DEC VMS and RSTS/E operating systems including system support, data management and financial planning. RABBIT Systems are marketed and supported throughout the U.S.A., Canada, U.K., France and Germany.

RABBIT Systems are available for RSTS/E Version 7 plus users also.

For more information contact: Joseph Musler, RAXCO Inc., 3336 N. Flagler Drive, West Palm Beach, Florida 33407, U.S.A., Telephone: (305) 842-2115.

LD1: [1,3] AMORT.BAS

By R. Frazer, Applications Analyst for Nationwide Data Dialog, Inc.

```
1       EXTEND
10      !       AMORT.BAS       by R. Frazer, Applications Analyst      &
        !                       for Nationwide Data Dialog Inc.         &
        !                                                               &
        !                                                               &
        !       This program generates an amortization schedule with    &
        !       interest and principal components specified for each    &
        !       payment.  The final payment will be adjusted to include &
        !       all of the outstanding principal.                       &
        !                                                               &
        !       The payments are assumed to be monthly; the annual      &
        !       interest rate is divided by 12 and the printed date is  &
        !       incremented by one month per payment.                   &
        !                                                               &
        !       Note that you may specify a different monthly payment   &
        !       than the amount derived by the formula;                 &
        !       a smaller one results in a "balloon" final payment,     &
        !       since the principal does not approach zero correctly.   &
        !       However, if you specify a larger monthly payment and    &
        !       reduce the principal too quickly, the loan will expire  &
        !       prematurely with some final, partial payment.           &
        !
60        Z = .1E39                     ! DEMAND SCALE FACTOR ZERO
100       INPUT 'PRINCIPAL BALANCE ';O.B                                &
        \ INPUT 'NUMBER OF PAYMENTS';N%                                 &
        \ INPUT 'ANNUAL INTEREST % ';A.I                                &
        \ P$ = NUM1$(N%) + '-MONTH'                                     &
        \ P$ = NUM1$(N%/12%) + '-YEAR'                                  &
                IF (N%/12%) * 12. = 0. + N%     ! EVEN NO. OF YEARS     &
        !
200       B = O.B                                                       &
        \ I = A.I/1200.                 ! PERIOD INTEREST RATE          &
        \ X = 1. + I                                                    &
        \ X = (X^N%)/(X^N% - 1.)        ! FACTOR FOR FORMULA:           &
        !                                                               &
        !       M = B * I * (1 + I)^N / ((1 + I)^N - 1)                 &
        !                                                               &
        !       WHERE:  M = MONTHLY PAYMENT                             &
        !               B = PRINCIPAL BALANCE TO AMORTIZE               &
        !               I = PERIOD INTEREST RATE                        &
        !               N = TERM (NUMBER OF PERIODS)                    &
        !                                                               &
        !
300       Z = B * I * X                                                 &
        \ M = FNROUND.2(Z)                              ! MONTHLY PMT   &
        \ M = M + .01   IF M < Z                        ! ROUND UP      &
        \ PRINT 'MONTHLY PAYMENT <';                                    &
        \ PRINT USING '###,###.##', M;                                  &
        \ PRINT '> ';                                                   &
        \ INPUT X                                                       &
        \ M = X UNLESS X = 0.                                           &
        !
340       I1 = FNROUND.2(B*I)           ! 1ST INTEREST PORTION          &
        \ GOTO 500      UNLESS I1/M > 0.98                              &
        \ PRINT 'INTEREST/PAYMENT RATIO = ';                            &
        \ PRINT USING '###.#####', I1/M                                 &
        \ PRINT 'FOR THIS INTEREST RATE, YOU MAY NOT SCHEDULE';         &


                N%/12%;' YEARS OF PAYMENTS'                             &
        \ PRINT 'TRY AGAIN...'                                          &
        \ GOTO 100                                                      &
        !                                                               &
        !
500       PRINT 'OUTPUT DEVICE ';                                       &
        \ INPUT LINE O$                                                 &
        \ O$ = CVT$$(O$,-1%)                                            &
        \ O% = 0%                                                       &
        \ IF    LEN(O$)                                                 &
          THEN  O% = 6%                 ! WE HAVE SOME FOREIGN DEVICE   &
        \       OPEN O$ FOR OUTPUT AS FILE O%                           &
        \       PAGE.CT% = 0%                                           &
        !
1000      INPUT 'STARTING MM,DD,YY';MM%,DD%,YY%                         &
        \ P.DATE$ = SPACE$(8%)                                          &
        \ NO.DATE% = (MM% = 0%)                                         &
        \ MM% = MM% - 1%                                                &
        \ P.D$ = '00' + NUM1$(DD%)                                      &
        \ P.D$ = RIGHT(P.D$,LEN(P.D$)-1%)       ! IN CASE OF JUST 1 DIGIT &
        \ P.M$ = '..'                                                   &
        \ P.Y$ = 'YY'                           ! PREPARE FOR RSETS     &
        !
1100      GOSUB 8000                    ! COLUMN HEADINGS               &
        !
1200      PRINT #O%, TAB(57%);                                          &
        \ P.NB = B                                                      &
        \ P.N% = 0%                                                     &
        \ PRINT #O% USING '###,###.##', P.NB    ! INITIAL BALANCE       &
        \ LINE.CT% = LINE.CT% + 1%                                      &
        !
3000    !************ L O O P *****************                         &
        !                                                               &
          P.N% = P.N% + 1%                              ! PMT#          &
        \ GOTO 4000     IF P.N% > N%                                    &
        \ B = P.NB                                      ! BALANCE       &
        \ P.I = FNROUND.2(B*I)                          ! INTEREST      &
        \ P.P = M - P.I                                 ! PRINCIPAL     &
        \ P.NB = B - P.P                                ! NEW BALANCE   &
        \ GOTO 3040     IF P.N% < N%    ! ADJUST IF THIS IS LAST PMT#   &
                        IF P.NB > 0.    ! OR IT BOTTOMED OUT            &
        \ M = M + P.NB                  ! + OR - LAST PAYMENT           &
        \ P.P = P.P + P.NB              ! + OR - LAST PRINCIPAL         &
        \ P.NB = 0.                     ! FORCE CLEAN ENDING            &
        !
3040      MM% = MM% + 1%                                ! NEXT DATE     &
        \ IF    MM% > 12%                                               &
          THEN  MM% = 1%                                ! ROLL-OVER     &
        \       YY% = YY% + 1%                                          &
        !
3200      RSET P.M$ = '00' + NUM1$(MM%)                                 &
        \ RSET P.Y$ = '00' + NUM1$(YY%)                                 &
        \ LSET P.DATE$ = P.M$ + '/' + P.D$ + '/' + P.Y$                 &
                UNLESS NO.DATE%                                         &
        !
3600      GOSUB 8000    IF LINE.CT% > 59%       IF O% > 0%              &
        \ PRINT #O%, TAB(8%);                                           &
        \ PRINT #O% USING '####', P.N%;                                 &
        \ PRINT #O%, '  ';P.DATE$;'  ';                                 &
        \ PRINT #O% USING '##,###.##  ##,###.##  ##,###.##  ###,###.##', &
                M, P.P, P.I, P.NB                                       &
        \ LINE.CT% = LINE.CT% + 1%                                      &
        !
3900    GOTO 3000       UNLESS P.NB = 0.                                &
        !                                                               &
        !******************** L O O P ********************              &
        !
4000      PRINT #O%, CHR$(12%);                                         &
        \ CLOSE  O%                                                     &
        \ GOTO 100                                                      &
        !                                                               &
        !
8000      GOTO 8020     UNLESS O%                                       &
        \ PRINT #O%, CHR$(12%)                                          &
        \ PAGE.CT% = PAGE.CT% + 1%                                      &
        \ PRINT #O%, TAB(8%); P$; ' AMORTIZATION OF ';                  &
        \ PRINT #O% USING '###,###.##', O.B;                            &
        \ PRINT #O%, ' @ ';NUM1$(A.I);'% ANNUALLY';                     &
                TAB(65%);' PAGE';PAGE.CT%                               &
        \ PRINT #O%                                                     &
        !
8020      PRINT #O%, TAB(8%);                                           &
         ' PMT#   DATE       PAYMENT (PRINCIPAL + INTEREST) NEW BALANCE' &
        \ PRINT #O%                                                     &
        \ LINE.CT% = 5%                                                 &
        \ RETURN                                                        &
        !
20006   !>      FNROUND.2  ROUND FLOATING POINT NUMBERS TO 2 PLACES     &
        !                                                               &
          DEF FNROUND.2(Z) = FIX(100. * Z + .5) / 100.                  &
        !
32767     END
```

JBSTAT . . . continued from page 50

```
100     !-------------------------------------------------------------------!&
        !PROGRAM                AUTHOR                    DATE              !&
        !                                                                   !&
        !JBSTAT.B2S             FRANCOIS DUBOIS           1980-12-19        !&
        !                                                                   !&
        !VERSION 2                                                          !&
        !-------------------------------------------------------------------!&
        !DESCRIPTION:                                                       !&
        !       DISPLAY JOB STATUS AND OPEN CHANNEL INFORMATION             !&
        !                                                                   !&
        !       EXAMPLE:                                                    !&
        !               J/K5            FOR JOB OWNING KB5:                 !&
        !               J/19-           FOR JOB 19 (ONLY OPEN CHAN. INF)    !&
        !               J/8:R4:C4:C5    FOR JOB 8,DISPLAY EVERY 4 SEC.      !&
        !                               INFORMATION ABOUT CHANNELS 4,5      !&
        !               J/3:S:R3        FOR JOB 3,DISPLAY ONLY THE          !&
        !                               STATUS EVERY 3 SECONDS              !&
        !               J/LP0:=20       FOR JOB 20 (ON LP0:)                !&
        !               J/X.LOG=7       FOR JOB 7  (ON FILE X.LOG)          !&
        !                                                                   !&
        !       NOTE:                   SWITCH ":R??" WILL WORK FOR A       !&
        !                               MAXIMUM OF 60 SECONDS               !&
        !                                                                   !&
        !       DEFINITION OF "FLAGS":                                      !&
        !                                                                   !&
        !               P               FILE IS PLACED                      !&
        !               U               FILE IS OPEN IN UPDATE MODE         !&
        !               C               FILE IS CONTIGUOUS                  !&
        !               N               FILE CAN'T BE RENAMED OR DELETED    !&
        !               D               FILE IS MARKED FOR DELETION         !&
        !               T               FILE IS TENTATIVE                   !&
        !                                                                   !&
        !       DEFINITION OF "USER":                                       !&
        !                                                                   !&
        !               Rd              USER HAS READ  ACCESS               !&
        !               Wr              USER HAS WRITE ACCESS               !&
        !               Rr              OPEN READ REGARDLESS                !&
        !               Ca              OPEN FOR       USER DATA CACHING    !&
        !               Sq              OPEN FOR SEQ. USER DATA CACHING     !&
        !               Te              TENTATIVE FILE                      !&
        !               Up              OPEN FOR UPDATE                     !&
        !               Su              OPEN IN SPECIAL UPDATE MODE         !&
        !                                                                   !&
        !       KEYBOARD INFORMATION:                                       !&
        !                                                                   !&
        !               MODE    Bnry    BINARY                              !&
        !                       Teco    RESERVED FOR TECO                   !&
        !                       CrLf    SUPPRESS AUTOMATIC CR/LF            !&
        !                       Echo    ECHO CONTROL FOR BLOCK MODE         !&
        !                       Ctrc    GUARDS PROGRAM AGAINST CTRL/C       !&
        !                       XnXf    ENABLES INCOMING XON/XOFF PROC.     !&
        !                       ????    RESERVED                            !&
        !                       SRub    SPECIAL SCOPE RUBOUT                !&
        !                                                                   !&
        !               FIELD   (ONLY WITH MODE 8)                          !&
        !                       Act     FIELD IS ACTIVE                     !&
        !                       Kpch    KEYPUNCH MODE                       !&
        !                       NUMBER  NUMBER OF BYTES REMAINING TO        !&
        !                               FILL THE FIELD                      !&
        !                                                                   !&
        !               DZ11[LL]                                            !&
        !               DH11[LL]                                            !&
        !                       LL      LINE NUMBER FOR THAT     DH11 OR    !&
        !                                                        DZ11       !&
        !                                                                   !&
        !               STATUS  ©S      XOFF   WAS SENT                     !&
        !                       ©O      CTRL/O WAS SENT                     !&
        !                       NOECHO  NOT ECHOING                         !&
        !                                                                   !&
        !               BUFFER      !??! DATA IN BUFFER                     !&
        !                                                                   !&
        !               TYPE AHEAD !??! DATA TYPED AHEAD IN ECHO CONTROL    !&
        !                                                                   !&
        !       PRINTER INFORMATION                                         !&
        !                                                                   !&
        !               STATUS  HUNG    PRINTER NOT READY                   !&
        !                                                                   !&
        !       MAGTAPE INFORMATION                                         !&
        !                                                                   !&
        !               DENSITY         1600 BPI                            !&
        !                                800 BPI (EVEN PARITY)              !&
        !                                800 BPI (ODD  PATITY)              !&
        !                                                                   !&
        !               STATUS          CURRENT COMMAND                     !&
        !-------------------------------------------------------------------!&
        !LIBRARY ROUTINES:                                                  !&
        !-------------------------------------------------------------------!&
        !FUNCTIONS:                                                         !&
        !       FNR$(A$,B$)     REMOVE STRING X$ FROM STRING W$             !&
        !       FNN$(A%,B%)     TURN B% INTO A STRING AND LEFT PAD IT       !&
        !                       WITH SPACES                                 !&
        !       FNS$(A%,A$)     LEFT PAD STRING A$ WITH SPACES              !&
        !-------------------------------------------------------------------!&
        !CHANNELS:                                                          !&
        !       1       OUTPUT FILE                                         !&
        !-------------------------------------------------------------------!&
        !MODIFICATIONS:                                                     !&
        !-------------------------------------------------------------------!&
        !"BUGS":                                                            !&
        !-------------------------------------------------------------------!&
        !NOTE:                                                              !&
        !       THE INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE       !&
        !       WITHOUT NOTICE AND SHOULD NOT BE CONSTRUED AS A             !&
        !       COMMITMENT BY THE CITY OF LONGUEUIL.                        !&
        !                                                                   !&
        !       THIS SOFTWARE IS UN-RELEASED AND THE CITY OF LONGUEUIL      !&
        !       HAS NO COMMITMENT TO SUPPORT IT.                            !&
        !-------------------------------------------------------------------!&

200     !-------------------------------------------------------------------!&
        !DIM                                                                !&
        !-------------------------------------------------------------------!&
        DIM     M%(30%),                        !MONITOR TABLES PART I      !&
                M2%(30%)                        !MONITOR TABLES PART II     !&

300     ON ERROR GOTO 30000\                    !TRAP ERRORS                !&
        GOTO 1000                                                            &

310     !-------------------------------------------------------------------!&
        !DATA                                                               !&
        !-------------------------------------------------------------------!&
        DATA                                                                 &
                "Rewind then Off-Line"                  ,                    &
                "Read"                                  ,                    &
                "Write"                                 ,                    &
                "Write End-of-File"                     ,                    &
                "Rewind"                                ,                    &
                "Skip Forward"                          ,                    &
                "Backspace"                             ,                    &
                "Set Density/Parity"                    ,                    &
                "Return Unit Status"                    ,                    &
                "Return File Characteristics"           ,                    &
                "Rewind On Close"                                            &

400     !-------------------------------------------------------------------!&
        !FILE OPENINGS                                                      !&
        !-------------------------------------------------------------------!&

420     GOSUB 10400\                            !DROP TEMP. PRIVILEGES      !&
        OPEN OUT$ FOR OUTPUT AS FILE #1%\                                    &
        GOSUB 10440\                            !REGAIN PRIVILEGES          !&
        OPEN%=-1%\                              !SET FLAG OUTPUT OPENED     !&
        RETURN                                                               &

1000    !-------------------------------------------------------------------!&
        !MAIN PROGRAM                                                       !&
        !-------------------------------------------------------------------!&
        W$=SYS(CHR$(6%)+CHR$(-7%))\             !TRAP CTRL/C                !&
        TMD$="BnryTecoCrLfEchoCtrcXnXf????SRub"\                             &
        READ CMD$(W%) FOR W%=0% TO 10%\                                      &
        GOSUB 10020\                            !CHANGE PRIO-BURST          !&
        GOSUB 10300\                            !GET MONITOR TABLES         !&
        GOSUB 10340\                            !GET DEVICE NAME (DISKS)    !&
        GOSUB 10040 IF SCCL%\                   !GET CORE COMMON            !&
        GOSUB 10060 UNLESS LEN(CCS$)\           !TERMINAL INPUT             !&
        GOSUB 10100\                            !PARSE                      !&
        IF E%                                                                &
        THEN                                                                 &
                PRINT "ILLEGAL COMMAND - ";CCS$; "   (";W1$;")"\             &
                GOTO 32700                                                   &

1020    C%=C% OR ((C%=0%) AND (S.C%=0%))\                                    &
        TM=TIME(0)\                             !WALL TIME                  !&
        REFRESH%=-1%\                                                        &
        WHILE REFRESH%\                                                      &
                GOSUB 10600\                    !CHECK IF JOB IS ACTIVE     !&
                IF JOB%=-1%                     !ARE WE AT THE END OF       !&
                                                !THE JOB TABLE              !&
                THEN                                                         &
                        PRINT "JOB";JOB%;"NOT ACTIVE" IF JOB%\               &
                        PRINT "KB";NUM1$(KB%);" NOT IN USE" IF S.K%\         &
                        GOTO 32700                                           &

1040            GOSUB   420 UNLESS OPEN%\       !OPEN OUTPUT                !&
                GOSUB 10500\                    !SCAN DEVNAM TABLE          !&
                GOSUB 10700 UNLESS SC%\         !JOB STATUS                 !&
                GOSUB 11000 IF C%\              !CHANNELS INFORMATIONS      !&
                TM1=TIME(0)-TM\                 !TIME SINCE BEGINNING       !&
                REST=60-TM1\                    !REMAINING TIME             !&
                SLP%=SEC%\                      !SLEEPING TIME              !&
                SLP%=REST IF REST<SEC%\         !CORRECT SLEEPING TIME      !&
                REFRESH%=((SEC%<>0%) AND (TM1<60))\                          &
                                                !SET SWITCH REFRESH         !&
                SLEEP SLP% IF REFRESH%\         !WAIT                       !&
        NEXT\                                                                &
        GOTO 32700                                                           &

8000    !-------------------------------------------------------------------!&
        !FUNCTIONS                                                          !&
        !-------------------------------------------------------------------!&

8020    !-------------------------------------------------------------------!&
        !REMOVE STRING X$ FROM STRING W$ AND SET SC%                        !&
        !-------------------------------------------------------------------!&
        DEF FNR$(W$,X$)\                                                     &
        W%=INSTR(1%,W$,X$)\                                                  &
        SC%=(W%<>0%)\                                                        &
        WHILE W%\                                                            &
                W$=LEFT(W$,W%-1%)+RIGHT(W$,LEN(X$)+W%)\                      &
                W%=INSTR(1%,W$,X$)\                                          &
        NEXT\                                                                &
        FNR$=W$\                                                             &
        FNEND                                                                &

8040    !-------------------------------------------------------------------!&
        !TURN S% INTO A STRING$ AND LEFT PAD IT WITH SPACES                 !&
        !-------------------------------------------------------------------!&
        DEF FNN$(SP%,S%)\                                                    &
        W$=NUM1$((S% EQV 32767%)+32768.)\                                    &
        FNN$=FNS$(SP%,W$)\                                                   &
        FNEND                                                                &

8060    !-------------------------------------------------------------------!&
        !LEFT PAD A STRING                                                  !&
        !-------------------------------------------------------------------!&
        DEF FNS$(SP%,W$)=SPACE$(SP%-LEN(W$))+W$                              &

8080    !-------------------------------------------------------------------!&
        !CHANGE STRING X$ IN STRING W$  FOR STRING Y$                       !&
        !-------------------------------------------------------------------!&
        DEF FNCHANGE$(W$,X$,Y$)\                                             &
        W%=INSTR(1%,W$,X$)\                                                  &
        WHILE W%\                                                            &
                W$=LEFT(W$,W%-1%)+Y$+RIGHT(W$,LEN(X$)+W%)\                   &
                W%=INSTR(1%,W$,X$)\                                          &
        NEXT\                                                                &
        FNCHANGE$=W$\                                                        &
        FNEND                                                                &

10000   !-------------------------------------------------------------------!&
        !ROUTINES (GOSUB)                                                   !&
        !-------------------------------------------------------------------!&
```

```
10020   !------------------------------------------------------------------!&
        !STORE ORIGINAL PRIO-BURST AND CHANGE IT TO PRIO 120, BURST 36   !&
        !------------------------------------------------------------------!&
        BP%=PEEK(PEEK(520%)+28%)\                  !BURST AND PRIORITY   !&
        BR%=SWAP%(BP%) AND 255%\                   !BURST                !&
        PR%=BP%*256%/256%\                         !PRIORITY             !&
        W$= SYS(CHR$( 6%)+CHR$(-13%)+CHR$(255%)+!SYS CALL              !&
                CHR$(-1%)+CHR$(120%)+              !CHANGE PRIORITY TO 120 !&
                CHR$(-1%)+CHR$( 36%))\             !CHANGE BURST TO 36   !&
        RETURN                                                            &

10040   !------------------------------------------------------------------!&
        !GET CORE COMMON                                                  !&
        !------------------------------------------------------------------!&
        CCS$=SYS(CHR$(7%))\                        !GET CORE COMMON      !&
        CCS$=RIGHT(CCS$,I%+2%)                     !GET SWITCHES ONLY    !&
                IF LEFT(CCS$,I%)=LEFT("JBSTAT",I%)                        &
                        FOR I%=6% TO 1% STEP -1%\                         &
        CCS$=EDIT$(CCS$,32%+4%+2%+1%)\                                    &
        RETURN                                                            &

10060   !------------------------------------------------------------------!&
        !TERMINAL INPUT                                                   !&
        !------------------------------------------------------------------!&
        LINPUT CCS$\                                                      &
        CCS$=EDIT$(CCS$,32%+4%+2%+1%)                                     &

10080   RETURN                                                            &

10100   !------------------------------------------------------------------!&
        !PARSE                                                            !&
        !------------------------------------------------------------------!&
        W$=CCS$\                                                          &
        W$=FNR$(W$,"-")\                           !GET RID OF "-"       !&
        OUT$="KB:"\                                !DEFAULT OUTPUT IS KB: !&
        I%=INSTR(1%,W$,"=")\                                              &
        IF I%                                                             &
        THEN                                                              &
                OUT$=LEFT(W$,I%-1%)\               !OUTPUT FILE          !&
                OUT$="KB:" UNLESS P0%\             !FORCE OUTPUT TO KB: IF !&
                                                   !NOT LOGGED IN        !&
                W$=RIGHT(W$,I%+1%)                                        &

10120   W$=W$+":"\                                                        &
        I%=INSTR(1%,W$,":")\                                              &
        WHILE I% AND E%=0%\                                               &
                W1$=LEFT(W$,I%-1%)\                                       &
                W$=RIGHT(W$,I%+1%)\                                       &
                IF LEN(W1$)                                               &
                THEN                                                      &
                        W2$=LEFT(W1$,1%)\                                 &
                        IF W2$="S"              !STATUS                  !&
                        THEN                                              &
                                S.C%=-1%                                  &
                        ELSE                                              &
                        W2%=VAL(RIGHT(W1$,2%))\                           &
                        IF W2$="C"              !CHANNEL                 !&
                        THEN                                              &
                                SWC%=-1%\                                 &
                                C%=C% OR 2%**W2%!SELECTED CHANNELS       !&
                        ELSE                                              &
                        IF W2$="K"              !KEYBOARD                !&
                        THEN                                              &
                                S.K%=-1%\                                 &
                                KB%=W2%         !KEYBOARD #              !&
                        ELSE                                              &
                        IF W2$="R"              !REFRESH                 !&
                        THEN                                              &
                                SEC%=W2%        !SLEEPING TIME           !&
                        ELSE                                              &
                                JOB%=VAL(W1$)   !JOB #                   !&

10140           I%=INSTR(1%,W$,":")\                                      &
        NEXT\                                                             &
        IF E%=0% AND    ((S.K% AND JOB%)         OR !KB # AND JOB #       !&
                         (S.K%=0% AND JOB%=0%)   OR !NO KB # AND NO JOB #!&
                         (S.C% AND C%)           OR !STATUS AND CHANNELS !&
                         (S.C% AND SC%)          OR !STATUS AND NO STATUS!&
                         (SWC% AND (C%=0% OR C%=1%)))!BAD VALUE FOR      !&
                                                    !CHANNEL OR CHANNEL 0 !&
                                                    !WAS SPECIFIED       !&
        THEN                                                              &
                E%=-1%\                                                   &
                W1$=CCS$                                                  &

10160   RETURN                                                            &

10300   !------------------------------------------------------------------!&
        !GET MONITOR TABLES PART I - II, GET OWN PPN                      !&
        !------------------------------------------------------------------!&
        CHANGE SYS(CHR$(6%)+CHR$(-3%)) TO M%\                             &
        M%(I%)=M%(I%)+SWAP%(M%(I%+1%)) FOR I%=5% TO 29% STEP 2%\          &
        CHANGE SYS(CHR$(6%)+CHR$(-12%)) TO M2%\                           &
        M2%(I%)=M2%(I%)+SWAP%(M2%(I%+1%)) FOR I%=3% TO 29% STEP 2%\       &
        P0%=PEEK(PEEK(PEEK(520%)+8%)+24%)\         !OWN PPN              !&
        P%=(P0% AND -256%)=256%\                   !P%=-1% (PRIVILEGED)  !&
        RETURN                                                            &

10340   !------------------------------------------------------------------!&
        !GET DEVICES NAME FOR THE DISKS                                   !&
        !------------------------------------------------------------------!&
        DKNM$=DKNM$+CVT%$(SWAP%(PEEK(M2%(5%)+W1%)))+CHR$(48%+W%)          &
                FOR W%=0% TO PEEK(M%(5%)+W1%)                             &
                FOR W1%=0% TO M2%(9%)-2% STEP 2%\                         &
        RETURN                                                            &

10400   !------------------------------------------------------------------!&
        !DROP AND REGAIN PRIVILIGES                                       !&
        !------------------------------------------------------------------!&
        W$=SYS(CHR$(6%)+CHR$(-21%)+CHR$(-1%))\                            &
        RETURN                                                            &

10440   W$=SYS(CHR$(6%)+CHR$(-21%)+CHR$(0%))\                             &
        RETURN                                                            &
```

```
10500   !------------------------------------------------------------------!&
        !SCAN THE DEVNAM TABLE TO SEE IF PK'S ARE LEGAL                   !&
        !------------------------------------------------------------------!&
        P2%,P3%=0%\                                                       &
        I%=M2%(5%)+M2%(9%)\                                               &
        WHILE (P2%=0%) AND (I% < M2%(23%))\                               &
                IF CVT%$(SWAP%(PEEK(I%)))="PK"                            &
                THEN                                                      &
                        PCNT%=PEEK(M%(5%)+I%-M2%(5%))\  !NUMBER OF PK'S !&
                        P2%=PEEK(M%(7%)+I%-M2%(5%))\    !DEVTBL POINTER !&
                        P3%=PEEK(PEEK(P2%)+8%)/2%       !KB # FOR PK0   !&

10520           I%=I%+2%\                                                 &
        NEXT\                                                             &
        RETURN                                                            &

10600   !------------------------------------------------------------------!&
        !FIND OUT IF SPECIFIED JOB (OR KB#) IS ACTIVE ON THE SYSTEM       !&
        !------------------------------------------------------------------!&
        J%,JDB%=0%\                                !RESET                !&
        WHILE NOT(((S.K% AND (KB%=K%) AND NOT(DET%))                      &
                        OR (JOB%=J%))                                     &
                        AND JDB%<>0%)                                     &
                        AND JDB%<>-1%\                                    &
                DET%=0%\                           !PRETEND ATTACHED     !&
                J%=J%+1%\                          !JOB #                !&
                JDB%=PEEK(M%(11%)+(J%*2%))\        !JOB DATA BLOCK       !&
                IF JDB%<>-1% AND JDB%<>0%          !NOT END OF TABLE AND !&
                                                   !JOB IS ACTIVE        !&
                THEN                                                      &
                        JDFLG%=PEEK(JDB%+2%)\   !STATUS FLAGS            !&
                        JDB2%=PEEK(JDB%+8%)\    !JOB DATA BLOCK 2        !&
                        PPN%=PEEK(JDB2%+24%)\   !PPN                     !&
                        IOB%=PEEK(JDB%)\        !I/O BLOCK               !&
                        DDB%=PEEK(IOB%)\        !DEVICE DATA BLOCK       !&
                        INTFAC%=PEEK(DDB%+30%) AND 255%\!INTERFACE FLAG !&
                        K%=SWAP%(PEEK(DDB%+2%)) AND 255%\!KEYBOARD #     !&
                        DET%=-1% UNLESS (PEEK(DDB%+2%) AND 255%)=J%*2%   &
                                AND (PEEK(DDB%+6%) AND 8192%)<>0%        &
                                                   !SET FLAG DETATCHED   !&

10620   NEXT\                                                             &
        RETURN                                                            &

10700   !------------------------------------------------------------------!&
        !PRINT JOB STATUS                                                 !&
        !------------------------------------------------------------------!&
        W$=EDIT$(RIGHT(SYS(CHR$(6%)+CHR$(9%)+CHR$(0%)),3%),4%)\           &
        PRINT #1%\                                                        &
        PRINT #1%,W$;" JBSTAT ";DATE$(0%);" ";TIME$(0%);CHR$(10%)\        &
        PRINT #1%,      "Job"           ;                                 &
                TAB(11%);"Who"          ;                                 &
                TAB(19%);"Where"        ;                                 &
                TAB(27%);"What"         ;                                 &
                TAB(36%);"Size"         ;                                 &
                TAB(44%);"State"        ;                                 &
                TAB(54%);"Rts"          ;                                 &
                TAB(62%);"Prio/Brst"    ;                                 &
                CHR$(10%)                                                 &

10720   J$=FNS$(2%,NUM1$(J%))\                     !JOB #                !&
        W$=""\                                                            &
        W$="-" IF JDFLG% AND 256%\                 !TEMP. PRIVILEGES DROPED!&
        W$="+" IF JDFLG% AND 2048%\                !TEMP. PRIVILEGES ACTIVE!&
        J$=J$+" "+W$\                                                     &
                                                                          &
        W$=FNS$(3%,NUM1$(SWAP%(PPN%) AND 255%))+!PROJECT #               !&
                ","+NUM1$(PPN% AND 255%)\          !PROGRAMMER #         !&
        W$=W$+SPACE$(7%-LEN(W$))\                                         &
        W$="***,***" UNLESS PPN%\                  !NOT LOGGED IN        !&
        WHO$="["+W$+"]"\                                                  &
                                                                          &
        W$="KB"\                                                          &
        IF INTFAC%=8% AND NOT(DET%)                                       &
        THEN                                                              &
                W$="PK"\                                                  &
                K%=K%-P3%\                                                &
                W%=(PEEK(PEEK(P2%+K%*2%)+2%) AND 255%)/2%\                &
                IF W%                                                     &
                THEN                                                      &
                        W$="P"+NUM1$(K%)+"J"\                             &
                        K%=W%                                             &

10740   WHERE$=W$+NUM1$(K%)\                       !KB # OR PK#          !&
        WHERE$="Det" IF DET%\                      !DETATCHED            !&
                                                                          &
        WHAT$=RAD$(PEEK(JDB2%+12%))+RAD$(PEEK(JDB2%+14%))\!JOB NAME      !&
                                                                          &
        SIZE$=  FNN$(2%,(PEEK(JDB%+22%) AND 255%))+"/"+!ACTUEL SIZE      !&
                FNN$(2%,(PEEK(JDB%+30%) AND 255%))+"K"\!MAXIMUM SIZE     !&
                                                                          &
        W%=PEEK(M%(15%)+(J%*2%))\                                         &
        IF (W% AND PEEK(M%(13%)+(J%*2%)))                                 &
        THEN                                                              &
                W$="RN"                            !RUNNING              !&
        ELSE                                                              &
        IF PEEK(JDB%+24%)=0% AND                                          &
          (PEEK(JDB%+22%) AND -256%)=(NOT 15359%)                         &
        THEN                                                              &
                W$="RS"                            !RESIDENCY WAIT       !&
        ELSE                                                              &
        W%=W% AND (NOT 16384%) IF (W% AND (NOT 16384%))\                  &
        IF W% AND 16384%                                                  &
        THEN                                                              &
                W$="BF"                            !BUFFER WAIT          !&
        ELSE                                                              &
        IF W% AND 8192%                                                   &
        THEN                                                              &
                W$="SL"\                           !SLEEP WAIT           !&
                W$="SR" IF (PEEK(JDB%+4%) AND 255%)=5%!RECEIVER WAIT     !&
        ELSE                                                              &
        IF W% AND 4096%                                                   &
        THEN                                                              &
                W$="FP"                            !FILE PROCESSING WAIT !&
        ELSE                                                              &
        IF W% AND 2048%                                                   &
        THEN                                                              &
                W$="TT"                            !TERMINAL OUTPUT WAIT !&
```

```
        ELSE                                                          &
        IF W%=0%                                                      &
        THEN                                                          &
                W$="HB"                         !HIBERNATING          !&
        ELSE                                                          &
                CHN%=PEEK(PEEK(JDB%+6%)+6%) AND 255%\!CHANNEL #       !&
                SCB.WCB%=PEEK(IOB%+CHN%)\       !POINTER TO WCB       !&
                DIDX%=PEEK(SCB.WCB%) AND 255%\  !DRIVER INDEX         !&
                IF DIDX%=0%                     !0 = DISK             !&
                THEN                                                  &
                        F.U%=PEEK(PEEK(SCB.WCB%+8%)-4%) AND 255%\     &
                                                !FIP UNIT NUMBER      !&
                        W$=CVT%$(SWAP%(PEEK(M2%(21%)+(F.U%*10%)+4%)))\ &
                                                !DISK NAME            !&
                        W$="??" UNLESS LEN(CVT$$(W$,-1%))             &
                                                !CANNOT BE DETERMINED !&
                ELSE                                                  &
                        W$=CVT%$(SWAP%(PEEK(M2%(5%)+M2%(9%)+DIDX%-2%)))\ &
                                                !DEVICE NAME          !&
                        W$="^C" IF PEEK(PEEK(JDB%+6%)+10%)<0% IF DIDX%=2%&
                                                !^C ON KB INPUT WAIT  !&

10760   STATE$=W$+" "\                                                &
        W%=SWAP%(PEEK(JDB%+30%)) AND 255%\                            &
        W1%=PEEK(JDB%+22%)\                                           &
        IF W1%<0%                                                     &
        THEN                                                          &
                W%=64%\                         !LOCKED               !&
                IF W1% AND 1536%                                      &
                THEN                                                  &
                        W%=W%+64%\              !SWAPPING IN          !&
                        IF W1% AND 512%                               &
                        THEN                                          &
                                W%=W%+64%       !SWAPPING OUT         !&

10780   IF W% AND 63%                                                 &
        THEN                                                          &
                W$=CHR$(65%+(W% AND 192%)/64%)+ !SWAPPING FILE        !&
                   RIGHT(NUM$(100%+(W% AND 63%)),3%)!SLOT NUMBER      !&
        ELSE                                                          &
                W$=MID("    Lck Swi Swo ",(W% AND 192%)/16%+1%,4%)\   &
                W$="NSW " IF (W% AND 192%)=0% AND (JDFLG% AND 16384%)<>0%&
                                                !NOT TO BE SWAPPED    !&

10800   STATE$=STATE$+W$\                                             &
                                                                      &
        W%=PEEK(JDB%+12%)\                                            &
        RTS$=RAD$(PEEK(W%+2%))+RAD$(PEEK(W%+4%))\!RTS NAME            !&
                                                                      &
        W$=FNS$(4%,NUM1$(PEEK(JDB%+28%)*256%/256%))\    !PRIORITY     !&
        PB$=W$+"/"+NUM1$(SWAP%(PEEK(JDB%+28%)) AND 255%)\!RUN BURST   !&
                                                                      &
        PRINT #1%,              J$      ;                             &
                TAB(08%);       WHO$    ;                             &
                TAB(19%);       WHERE$  ;                             &
                TAB(26%);       WHAT$   ;                             &
                TAB(35%);       SIZE$   ;                             &
                TAB(44%);       STATE$  ;                             &
                TAB(53%);       RTS$    ;                             &
                TAB(62%);       PB$     ;                             &
                CHR$(10%)\                                            &
        RETURN                                                        &

11000   !-------------------------------------------------------------!&
        !CHANNEL INFORMATION                                          !&
        !-------------------------------------------------------------!&
        PRINT #1% IF SC%\                                             &
        IF SC%=0% OR C%=-1%                                           &
        THEN                                                          &
                PRINT #1%,              "Chn"           ;             &
                        TAB(04%);       "Dev"           ;             &
                        TAB(11%);       "PPn"           ;             &
                        TAB(18%);       "File Name"     ;             &
                        TAB(31%);       "Size"          ;             &
                        TAB(36%);       "Flags"         ;             &
                        TAB(42%);       "Prot"          ;             &
                        TAB(47%);       "Clu"           ;             &
                        TAB(53%);       "Op/Rr"         ;             &
                        TAB(63%);       "User"          ;             &
                        TAB(72%);       "Block"         ;             &
                        CHR$(10%)                                     &

11020   GOSUB 11100 IF (C% AND 2%**CH%) FOR CH%=1% TO 15%\!INFORMAT. FOR!&
                                                !SELECTED CHANNELS ONLY !&
        RETURN                                                        &

11100   WCB%=PEEK(IOB%+CH%*2%)\                 !POINTER TO WCB       !&
        RETURN UNLESS WCB%\                     !IS CHANNEL OPEN      !&
        STA%=PEEK(WCB%)\                                              &
        DIDX%=STA% AND 255%\                    !DRIVER INDEX         !&
        IF DIDX%                                                      &
        THEN                                                          &
                GOSUB 11200\                    !OTHER THAN DISK      !&
                GOSUB 12000 IF DIDX%=2%\        !IF WE HAVE A KEYBOARD !&
                GOSUB 13000 IF LEFT(DEV$,2%)="MM" OR LEFT(DEV$,2%)="MT"\ &
                                                !IF WE HAVE A TAPE    !&
                GOSUB 14000 IF LEFT(DEV$,2%)="LP"\!IF WE HAVE A PRINTER !&
                PRINT #1% IF CCPOS(1%)                                &
        ELSE                                                          &
                GOSUB 11400                     !WE HAVE A DISK       !&

11120   RETURN                                                        &

11200   DEV$=CVT%$(SWAP%(PEEK(M2%(5%)+M2%(9%)+DIDX%-2%)))\!DEVICE NAME !&
        UN%=SWAP%(PEEK(WCB%+2%)) AND 255%\      !UNIT NUMBER          !&
        W$=NUM1$(UN%)\                                                &
        DEV$=DEV$+W$ UNLESS DEV$="NL"\                                &
        DEV$=DEV$+":"\                                                &
        PRINT #1%,      FNS$(2%,NUM1$(CH%));".";                      &
                TAB(4%);DEV$;\                                        &
        RETURN                                                        &

11400   FCB%=PEEK(WCB%+8%)-28%\                 !POINTER TO FCB       !&
        DEV$=MID(DKNM$,3%*(PEEK(FCB%+24%) AND 255%)+1%,3%)+":"\       &
                                                !DISK NAME            !&
        W=65536.*(SWAP%(PEEK(FCB%+24%)) AND 255%)+32768.+             &
                (PEEK(FCB%+26%) EQV 32767%)\    !FILE SIZE            !&
        SIZE$=FNS$(6%,NUM1$(W))\                                      &
        CLU$=FNN$(3%,PEEK(FCB%+28%))\           !FILE CLUSTERSIZE     !&
        W=65536.*(SWAP%(PEEK(WCB%+4%)) AND 255%)+32768.+              &
                (PEEK(WCB%+6%) EQV 32767%)\     !CURRENT BLOCK        !&
        BL$=FNS$(6%,NUM1$(W))\                                        &
        GOSUB 11500 IF STA% AND 256%\           !IF NON-FILE-STRUCTURED !&
        GOSUB 11600 IF (STA% AND 256%)=0%\      !IF NOT  N-F-STRUCTURED !&
                                                                      &
        PRINT #1%,              FNS$(2%,NUM1$(CH%));".";              &
                TAB(04%);       DEV$    ;                             &
                TAB(29%);       SIZE$   ;                             &
                TAB(35%);       FL$     ;                             &
                TAB(41%);       PROT$   ;                             &
                TAB(47%);       CLU$    ;                             &
                TAB(52%);       OPN$    ;                             &
                TAB(60%);       USER$   ;                             &
                TAB(71%);       BL$     ;                             &
                TAB(77%);       LK$\                                  &
        RETURN                                                        &

11500   DEV$=DEV$+"NON-FILE-STRUCTURED"\                              &
        FL$,PROT$,OPN$,USER$,LK$=""\                                  &
        RETURN                                                        &

11600   DEV$=DEV$+"["+FNN$(3%,SWAP%(PEEK(FCB%+4%)) AND 255%)\         &
                                                !PROJECT NUMBER       !&
        W$=SPACE$(3%)\                                                &
        LSET W$=NUM1$(PEEK(FCB%+4%) AND 255%)\  !PROGRAMMER NUMBER    !&
        DEV$=DEV$+","+W$+"]"\                                         &
        W$=RAD$(PEEK(FCB%+6%))+RAD$(PEEK(FCB%+8%))+"."+!FILE NAME     !&
                RAD$(PEEK(FCB%+10%))\           !EXTENSION            !&
        W$=" -- UFD --" IF STA% AND 16384%\     !IF IT IS A UFD       !&
        DEV$=DEV$+W$\                                                 &
                                                                      &
        S%=PEEK(FCB%+12%)\                                            &
        W$=""\                                                        &
        W$=   "P"       IF S% AND 2%\           !PLACED FILE          !&
        W$=W$+"U"       IF S% AND 8%\           !UPDATE MODE          !&
        W$=W$+"C"       IF S% AND 16%\          !CONTIGUOUS FILE      !&
        W$=W$+"N"       IF S% AND 32%\          !CAN'T BE RENAMED OR  !&
                                                !DELETED              !&
        FL$=W$\                                                       &
        W$=STRING$(((S% AND 128%)<>0%)*-1%,     !MARKED FOR DELETION OR !&
                68%-16%*(((PEEK(WCB%) AND 2048%)=0%) AND !TENTATIVE   !&
                        ((PEEK(WCB%+12%) AND 2%)<>0%)))\              &
        FL$=FL$+W$\                                                   &
        FL$=SPACE$((6%-LEN(FL$))/2%)+FL$\                             &
                                                                      &
        PROT$="<"+FNS$(3%,NUM1$(SWAP%(PEEK(FCB%+12%)) AND 255%))+">"\ &
                                                !PROTECTION CODE      !&
                                                                      &
        W%=PEEK(FCB%+14%)\                                            &
        OPN$=FNN$(3%,W% AND 255%)+"/"+NUM1$(SWAP%(W%) AND 255%)\      &
                                                !OPEN COUNT           !&
                                                                      &
        W$=""\                                                        &
        W$=W$+"Rd" IF (STA% AND  512%)=0%\      !READ  ACCESS         !&
        W$=W$+"Wr" IF (STA% AND 1024%)=0%\      !WRITE ACCESS         !&
        W.WCB%=PEEK(WCB%+12%)\                                        &
        W$=W$+"Rr" IF W.WCB% AND 1%\            !READ REGARDLESS      !&
        W$=W$+"Ca" IF W.WCB% AND 8%\            !DATA CACHING         !&
        W$=W$+"Sq" IF W.WCB% AND 16%\           !SEQUENTIAL DATA CACHING!&
        USER$=W$\                                                     &
        W$=""\                                                        &
        W$="Te" IF (STA% AND 2048%)= 0% AND (W.WCB% AND 2%)<>0%\      &
                                                !TENTATIVE FILE       !&
        W$="Up" IF (STA% AND 2048%)<>0% AND (W.WCB% AND 2%)= 0%\      &
                                                !UPDATE MODE          !&
        W$="Su" IF (STA% AND 2048%)<>0% AND (W.WCB% AND 2%)<>0%\      &
                                                !SPECIAL UPDATE MODE  !&
        USER$=USER$+W$\                                               &
        USER$=SPACE$((10%-LEN(USER$))/2%)+USER$\                      &
                                                                      &
        LK$=""\                                                       &
        LK$="Lk" IF STA% AND 8192%\             !RECORD IS LOCKED     !&
                                                                      &
        RETURN                                                        &

12000   !-------------------------------------------------------------!&
        !KEYBOARD INFORMATION                                         !&
        !-------------------------------------------------------------!&
        W%=PEEK(WCB%+26%) AND 255%\                                   &
        W$="Mode: "\                                                  &
        W$=W$+MID(TMD$,(X%*4%)+1%,4%)+" "       !OPEN MODES FOR KEYBOARD!&
                IF W% AND 2%**X% FOR X%=0% TO 7%\                     &
        MODE$=""\                                                     &
        MODE$=W$ IF W%\                                               &
        FLD$=""\                                                      &
        M8%=0%\                                 !SET FLAG MODE 8 "OFF" !&
        IF W% AND 8%                            !IF ECHO CONTROL MODE !&
        THEN                                                          &
                M8%=-1%\                        !SET FLAG MODE 8  "ON" !&
                W%=PEEK(WCB%+38%)\                                    &
                W$="Field: "\                                         &
                W$=W$+"Act " IF W%<0%\          !FIELD IS ACTIVE      !&
                W$=W$+"Kpch " IF (W% AND 128%)=0%\!KEYPUNCH MODE      !&
                FLD$=W$+NUM1$(W% AND 127%)+" "  !NUMBER OF BYTES LEFT !&

12020   W%=PEEK(WCB%+30%)\                                            &
        INTERFACE$=MID("KLDCDLDEPKDJDHDZ",(W% AND 255%)+1%,2%)\       &
        IF LEN(INTERFACE$)                                            &
        THEN                                                          &
                INTERFACE$=INTERFACE$+"11["+NUM1$(SWAP%(W%) AND 15%)+"]" &
                                                !SUB-LINE NUMBER      !&

12040   STATUS$,W$=""\                                                &
        W$="^O" IF STA%<0%\                     !^O (JUNK OUTPUT)     !&
        DDFLAG%=PEEK(WCB%+8%)\                                        &
        W$=W$+"^S" IF DDFLAG%<0%\               !^S (XOFF)            !&
        W$=W$+" Noecho" IF DDFLAG% AND 32%\     !NO ECHO              !&
        STATUS$="Status:"+W$ IF LEN(W$)\                              &
        PRINT #1%,MODE$;FLD$;INTERFACE$;STATUS$\                      &
                                                                      &
```

```
        RETURN IF DDFLAG% AND 32%\                  !DON'T SHOW BUFFER IF NO!&
                                                    !ECHO BIT IS ON         !&
                                                                             &
        W$=""\                                                               &
        EFL%=0%\                                    !SET FLAG "ODD ADDR" OFF!&
        EP%=PEEK(WCB%+18%)\                         !EMPTY POINTER          !&
        FP%=PEEK(WCB%+20%)\                         !FILL  POINTER          !&
        BU$="Buffer: "\                                                      &
        GOSUB 12060\                                                         &
        RETURN UNLESS M8%\                          !DISPLAY TYPE AHEAD IF  !&
                                                    !MODE 8 (ECHO CONTROL)  !&
        W$=""\                                                               &
        EP%=PEEK(WCB%+32%)\                         !EMPTY POINTER          !&
        FP%=PEEK(WCB%+34%)\                         !FILL  POINTER          !&
        BU$="Type ahead: "\                                                  &
        EFL%=((EP% AND 1%)<>0%)\                    !SET FLAG "ODD ADDR"    !&
        EP%=EP%-1% IF EFL%\                         !IF WE HAVE AN ODD ADDR.!&
                                                    !THEN MAKE IT EVEN      !&
                                                    !BECAUSE WE CAN'T PEEK  !&
                                                    !AT AN ODD ADDRESS      !&
        GOSUB 12060\                                                         &
        RETURN                                                               &

12060   WHILE (EP%<>FP%) AND (EP%-1%<>FP%)\                                  &
                IF (EP% AND 31%)<>0%                !IF END OF SMALL BUFFER !&
                THEN                                                         &
                        W$=W$+CVT%$(SWAP%(PEEK(EP%)))\                       &
                                                    !GET 2 CHARACTERS FROM  !&
                                                    !SMALL BUFFER           !&
                        EP%=EP%+2%                  !INCREMENT ADDRESS BY 2 !&
                ELSE                                                         &
                        EP%=PEEK(EP%-32%)           !POINTER TO DATA IN THE !&
                                                    !NEXT SMALL BUFFER      !&

12080   NEXT\                                                                &
        W$=RIGHT(W$,2%) IF EFL%\                    !DROP THE FIRST BYTE OF !&
                                                    !CHAIN IF WE STARTED ON !&
                                                    !AN ODD ADDRESS         !&
        W$=LEFT(W$,LEN(W$)-1%) IF EP%>FP%\          !DROP THE LAST BYTE OF  !&
                                                    !CHAIN IF THE FILL PTR  !&
                                                    !HAS AN ODD ADDRESS     !&
        IF LEN(W$)                                                           &
        THEN                                                                 &
                W$=FNCHANGE$(W$,CHR$(09%),"<tab>")\                          &
                W$=FNCHANGE$(W$,CHR$(10%),"<lf>")\                           &
                W$=FNCHANGE$(W$,CHR$(12%),"<ff>")\                           &
                W$=FNCHANGE$(W$,CHR$(13%),"<cr>")\                           &
                W$=FNCHANGE$(W$,CHR$(W1%),"^"+CHR$(W1%+64%))                 &
                        FOR W1%=1 TO 26%\                                    &
                W$=FNCHANGE$(W$,CHR$(W1%),"<"+NUM1$(W1%)+">")                &
                        FOR W1%=27% TO 31%\                                  &
                BUFFER$=BU$+"¶"+W$+"¶"\                                      &
                PRINT #1%,TAB(9%);BUFFER$                                    &

12100   RETURN                                                               &

13000   !-----------------------------------------------------------------!&
        !MAGTAPE INFORMATION                                              !&
        !-----------------------------------------------------------------!&
        W%=PEEK(WCB%+22%) AND 127%\                                          &
        BPI$=""\                                                             &
        BPI$="1600 BPI" IF (W% AND 104%)=0%\     !1600 BPI                 !&
        BPI$="800 BPI (Even)" IF (W% AND 104%)=104%\!800 BPI EVEN PARITY!&
        BPI$="800 BPI (Odd)" IF (W% AND 104%)=96%\  !800 BPI ODD  PARITY!&
        BPI$="Density: "+BPI$+"," IF LEN(BPI$)\                              &
                                                                             &
        W%=PEEK(WCB%+26%) AND 255%\                                          &
        CMD$=""\                                                             &
        CMD$=" Status: "+CMD$(W%) IF W%<11%\    !CURRENT COMMAND           !&
        PRINT #1%,BPI$;CMD$\                                                 &
        RETURN                                                               &

14000   !-----------------------------------------------------------------!&
        !PRINTER INFORMATION                                              !&
        !-----------------------------------------------------------------!&
        STATUS$=""\                                                          &
        STATUS$=" Status: Hung" IF STA%<0%\                                  &
        PRINT #1%,STATUS$ IF LEN(STATUS$)\                                   &
        RETURN                                                               &

30000   !-----------------------------------------------------------------!&
        !ERROR ROUTINE                                                    !&
        !-----------------------------------------------------------------!&
        RESUME 30020                                                         &

30020   W$=SYS(CHR$(6%)+CHR$(-7%))\                 !TRAP CTRL/C           !&
        GOSUB 10440\                                !REGAIN TEMP. PRIVILEGES!&
        GOTO 32700 IF ERR=28%\                      !CTRL/C                !&
        GOTO 32700 IF ERR=11% AND ERL=10060         !CTRL/Z                !&

30040   IF ERL=10120 AND (ERR>49% AND ERR<53%)                               &
        THEN                                                                 &
                E%=-1%\                                                      &
                GOTO 10140                                                   &

30060   IF ERR=10% AND ERL=420                                               &
        THEN                                                                 &
                PRINT "ILLEGAL OUTPUT: ";OUT$\                               &
                GOTO 32700                                                   &

30080   GOTO 12100 IF ERR=33% AND ERL=12060                                  &

30100   PRINT ERR,ERL\GOTO 32700
31500   !-----------------------------------------------------------------!&
        !ENTERED VIA CCL                                                  !&
        !-----------------------------------------------------------------!&
        SCCL%=-1%\                                  !SET FLAG CCL "ON"     !&
        GOTO 100                                                             &

32000   !-----------------------------------------------------------------!&
        !ENTERED VIA LOGIN                                                !&
        !-----------------------------------------------------------------!&
        SCCL%=-1%\                                                           &
        W$=SYS(CHR$(7%))\                           !GET CORE COMMON       !&
        W$="JBSTAT"+W$\                                                      &
        W$=SYS(CHR$(8%)+W$)                         !PUT CORE COMMON       !&
        GOTO 100                                                             &

32700   !-----------------------------------------------------------------!&
        !END OF PROGRAM                                                   !&
        !-----------------------------------------------------------------!&
        CLOSE #1%\                                                           &
        W$= SYS(CHR$( 6%)+CHR$(-13%)+CHR$(255%)+!SYS CALL               !&
                CHR$(-1%)+CHR$(PR%)+                !ORIGINAL PRIORITY     !&
                CHR$(-1%)+CHR$(BR%))                !ORIGINAL BURST        !&

32767   END
```

RSTS DISK DIRECTORIES, Part 5

By Scott Banks, Systems Design

Introduction

I have planned this fifth episode, a general overview of FIP's role in maintaining disk directories, not to require the previous articles. We'll deal with conceptual OPENs, CLOSEs, and such, concerning ourselves little with all the blasted details. It would only be fair to admit that I currently have Volume 2, Number 1 of the RSTS Pro in front of me right now. So, if you want 'em, the UFD layouts begin on page 45. Hmmm... the 'marked for deletion' bit...

FIP and SATT.SYS

The File Processor, FIP, is a major component of the RSTS monitor. FIP is the code responsible for overall file management. All operations that affect the directory structure occur via FIP. This set of chores is more comprehesive than it appears at first. For instance, the RSTS monitor creates a new job, but it is FIP that (after running LOGIN) actually 'logs' the user into the system. Ditto for error messages (the text is located in [0,1] ERR.SYS) and run-time system changes (true, the linked list is in memory, but remember all those files with .RTS extensions). FIP even worries about magtape and other devices. For the moment, we'll restrict ourselves to those FIP operations that deal with disk directory maintainance and are readily enjoyable by the Basic-Plus programmer. In the interest of clarity, I will ignore the effects of any caching or large-files optimization.

Every RSTS disk must have a file in [0,1] called SATT.SYS, the Storage Allocation Truth Table. Each PDP-11 word has 16 bits, and 256 of these fit into a disk block. Therefore, each block of SATT contains 4096 bit flags. There is one bit flag for each device cluster available to the system. The filesize of SATT depends on the device clustersize and the total number of clusters. SATT can never be greater than 16 blocks (65536 clusters). Remember that we must always be able to describe uniquely any DCN via a 16-bit unsigned integer.

The SATT must be a contiguous file. When a disk is mounted, the location of its SATT is noted. FIP operations that affect the allocation of disk clusters to files reference and update the SATT. Any cluster that is currently not allocated to any file will have its correspondiing bit clear. That bit is set to 1 when the cluster is allocated, and reset to 0 when it is again free.

Your Basic File Search

For one reason or another, FIP may search for a specific file. There are three essential pieces of information required to do this. Two of these, the disk device (e.g. DB0:) and the account or PPN (e.g. [2,3]) are often allowed their default values. When supplied, they may be specified explicitly or by logical names or special characters (such as $). In any event, the device and PPN are definitely known with the effect that the search is narrowed down to exactly one UFD. The initial DCN (device cluster number) of this UFD may already be available. The UFD for SY:[1,2] and that of each job currently logged-in (corresponding to its PPN on SY:) appears in a monitor table. For random account selections, the MFD of the appropiate disk must be searched for the UFD entry. Obviously, if this is a fruitless effort, the file does not exist.

The UFD search is a serial scan of the linked list of Name blockettes. If one is stumbled upon that has a match for our desired filename, the search ends successfully. The search terminates in failure when the end of the UFD is reached. If the status byte of a name blockette has either its 'MFD' or 'marked for deletion' bits (6 and 7) set, that entry is ignored.

OPENing an Existing File

The Basic-Plus OPEN FOR INPUT statement, for example, attempts to open an existing file. FIP performs a search for the specific file, terminating with an error 5 if the file cannot be located. But even if it is there, the work is only half complete. The status and protection bytes in the Name blockette (word 4) determine whether or not the file may be accessed. The rules for file protection are applied to decide read vs. write access privileges. If no valid access is possible, the file is not opened and an error is returned.

The open file count, the lower byte of word 5 of the NB, will be zero if we are opening this file at a time when no other job already has it open. The upper byte (the read-only regardless open count) is of no significance in this respect. Assuming we are first in line, we have our choice of update mode or normal mode. These two may never be mixed, so we use bit 4 of the status byte to determine (from now until no channel has this file open) whether this is an update mode or a normal mode file. If this is an update OPEN, bit 4 will be set, else it will be cleared. Something must also be done about bit 2. For files opened mode 0 and write-access available, bit 2 will be set to indicate that write access has been given. If write access is not granted (either by restriction or mode request) bit 2 will be cleared.

If the file is already open by one or more jobs, special restrictions apply. If bit 3 of the status is set, this would mandate that the current OPEN itself be seeking update mode. On the other hand, if the bit is clear, this OPEN must not be attempting update mode. For normal mode OPENs, write access will not be granted if bit 2 is already set. If write access is given for this OPEN, then the bit will be set at this time.

For all files that are truely opened (no errors occur), the open file count is incremented. After some internal bookkeeping, the OPEN is logically complete. The job that requested the OPEN is now allowed to continue. All read and write transfers need only refer to the channel number under which the OPEN occured.

In both large and small-files systems, a certain amount of information is kept in memory. Each open channel has a file control block that retains data about the file opened. Large-file systems minimize disk transfers by using the directories only to locate files. All the status bytes and access counts are kept in memory via window control blocks. Furthermore, directory caching eliminates physical disk reads by keeping current copies of UFD blocks in memory.

KILLing a File

The first requirement for removing a file from the system is that it must exist. Assuming the file search has indeed located the doomed denizen of disk, the protection code and status bytes of the file must be checked. KILL demands that the protection code allow write access to the file (from whatever account and privilege status you happen to be under). Additionally, the delete/rename protection (bit 5 of the status byte) must not be selected. Even with permanent privilege, you cannot delete a file (such as SATT) when it has delete/rename protection (look for the P after the filesize).

If a KILL has made it to this point, the file will eventually be deleted from the system. Normally, it will be removed immediately. The file's Name blockette will be unlinked from the remaining NB's in the UFD. The device clusters allocated to the file will be returned to system by clearing one or more bits in the SATT for the proper disk. Then each blockette consumed by this directory entry will be zeroed, thus forming Hole blockettes available for future reuse. But I did say eventually, didn't I? If the file to be killed happens to be currently open, we just can't delete it. This could possibly allow two files to share the same space on disk, in a totally random and changing fashion. It's easy enough to check if any channel has this file open. If so, FIP simply marks the file for deletion by setting bit 7 of the status byte. In the section on CLOSE, we'll see how this action affects the file.

For non-executable files, the privilege bit of the protection code may be set to zero all blocks of the data file. This happens during the kill phase. For files of significant size, say 10000 blocks, this can take several minutes. Since FIP is single-threaded, it must complete this operation before going on to another job's request. Such an operation will effectively stop the computer, as few jobs go very long without a FIP request. This is an extreme example, but serves to illustrate the point that FIP optimization is an important part of getting the most out of RSTS. File and directory entry placement, minimizing OPENs and KILLs, FIP buffering, and FIP code residency are investments with a good return.

CLOSEing an Open Channel

As expected, CLOSE reverses the OPEN process. Most important is the fact that the open file counter is decremented. RSTS insists that no file or device may be opened on a channel that is already in use. BASIC-PLUS and other high-level environments buffer us from this decree by automatically issuing a CLOSE request before the OPEN is attempted.

If a file is marked for deletion, CLOSE depends on the KILL code to now do away with the file. Of course, if multiple channels have had the file open, this will happen when the access count finally drops to zero. Until such time, the file will not be accessible to OPENs, KILLs, and other directory operations. Aside from its appearance on DIRECTORY/S listings, the file is gone.

Another detail about CLOSE is that it has a counterpart called RESET. RESET is accessible in BASIC-PLUS by means of a CLOSE referencing a negative channel number. The RESET operation bypasses certain device related functions (such as final magtape EOF marks). For disk, this leads us to tentative files. When a file is created tentatively using MODE 32%, it is also marked for deletion. Performing a normal CLOSE on the channel will clear status bit 7, thus allowing the file to become permanent. This new file will supersede an existing file of the same name, working much like OPEN FOR OUTPUT. If you CLOSE using the negative of the channel number, the resulting RESET will kill the tentative file.

Creating a New File

The OPEN FOR OUTPUT statement will create a new data file. If a file of the same name already exists, that file will first be deleted. Therefore, as far as directory maintainence is concerned, the first step is to search for the file. If the file is not found, the creation simply continues. If it does exist, progress then becomes dependent upon whether it is allowable to delete the file. In such cases, the KILL routine is called.

FIP locates a hole (unused blockette) by scanning the UFD until it encounters a blockette with zero in word 0 and 1. This is to be the Accounting blockette and is loaded with two dates (today for both), a time (now), an RTS name (whatever run-time system happens to be running the current job), and the file's clustersize (which must remain constant for the life of the file). The filesize is set to 0 for now. The attribute link will be null.

A second hole is singled out for the distinction of becoming the Name blockette. The filename and extension, protection code (default unless specified), status byte, open counts, and the link to the AB (which is now known) are all intialized. The link to the next NB is made null to indicate that this is to be the last file in the UFD. The link to the Retrieval blockette is made null for now, consistent with a filesize of 0.

The new NB must be linked into the list of existing NB's. In order to create this file, the directory has been searched and found not to contain a duplicate, an entire scan of the NB list has been made. The last NB was noted, and will be the one whose link changes from null to point to the new guy. If we are now creating the first file in a UFD, the Label blockette will effectively play this role. In the event that new-files-first is invoked, the scene is only slightly different. The link from the Label blockette is copied to the new Name blockette and the LB is then set to point to the new NB.

If the creation did not require that a file of some definite size be created, we are done. If so, the procedure is essentially an extend operation. Let's see...

Extending a File

The length of a file may be increased by attempting to PUT a record that is beyond its last block. Files are always extended in increments of their own private clustersize. When the file's clustersize exceeds that of the device, two or more adjacent device clusters must be available and properly aligned to form the required extension cluster. The SATT is scanned to locate the correct pattern of 0 bits, indicating the cluster is available. As most files are created (and therefore extended) by the default clustersize, this really turns into a search for the first 0.

As clusters are allocated to the file, they are noted in the Retrieval blockettes of the directory entry. New RB's are taken from Holes as needed. The two advantages of using large clustersizes are fewer window turns and less directory space. Even for contiguous files, which completely eliminate window turns, choose a sensible clustersize in order keep the directory trim.

A file is extended until its filesize is long enough. If this cannot be done the operation ends in error. Usually the clusters can be allocated and the desired filesize is achieved. This doesn't mean that the last cluster allocated is completely used. For a new, zero length file having a clustersize of 256, PUTting record 6 will cause the filesize to become 6. Although FIP has actually allocated 256 blocks to the file, you cannot GET block 7. If the file is extended further, such partially used clusters are exploited completely before new ones are allocated.

Renaming a File

PIP is commonly used to rename files. The name, extension, and protection code, and run-time system are easily changed. If the filename and extension are not changed, there is no great impact upon the remainder of the UFD. Any valid protection code, for example, may be rewritten into the Name blockette without concern. Changing the filename/extension, on the other hand, implies a search of the UFD to ensure that the new filename will still be unique. As far as PIP goes, it will leave the file unchanged if the new name exists.

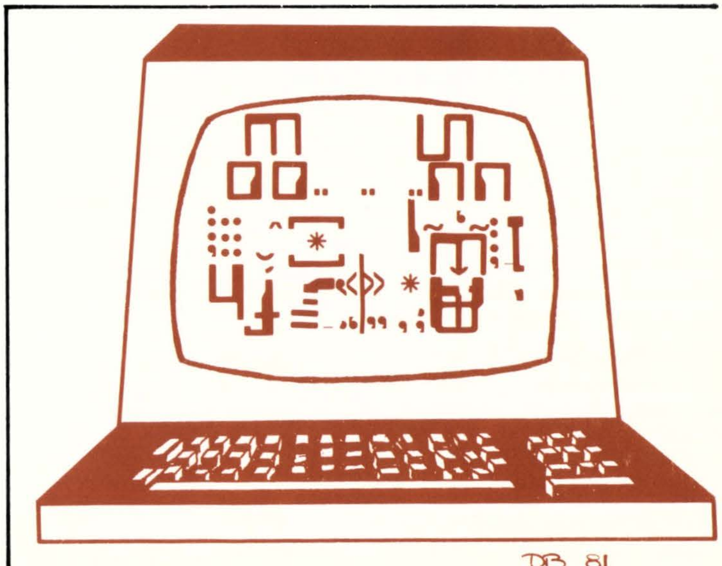
There is another mode in which rename may operate, in which a renamed file will supersede an existing one of the same name. TECO and other editors use this feature when properly exiting an edit session. Assume you have been editting PROG.BAS and have done so before so that PROG.BAK also exists. First, TECO will rename (with supersede) PROG.BAS to PROG.BAK (forcing the old PROG.BAK to be killed). Then the temporary work file TECOnn.TMP will be turned into the new PROG.BAS, again using the rename feature.

Wildcard Directory Lookups

FIP has a mechanism which allows system and application programs to lookup directory entries. To view the entries in a given UFD, the device, PPN, and filespec must be specified, typically through SYS() call. The filespec may optionally contain * or ? characters. In addition, the caller must supply an index value greater than or equal to 0. FIP will return information on the first file in the UFD that matches the filespec if an index value of 0 is given, the second match for index = 1, and so on. The SYS() call is repeatedly issued with ever increasing index values. When an index value greater than the number of matching files is attempted an error is returned, indicating the end of the directory.

The procedure followed is essentially the basic directory search, but allows wildcard matches and demands that a counter equal the index value. PIP issues this FIP call process for all its operations. The technique is inherently inefficient because, as the index value increases, the entire UFD Name blockette list up to that point must be traversed. The DIRECT program opens each UFD as a virtual array (as demonstrated in earlier articles of this series), but suffers from other high-level language drawbacks. I know of at least two macro directory listing programs which operate by opening a disk in nonfile-structured mode and use large buffersizes to optimize the whole affair.

Wildcard PPN lookups are accomplished similarly by suppling a device and one or two *'s in the PPN. Here the index value is again used with repeated SYS() calls to get each desired PPN. PIP combines both the wildcard PPN and filespec techniques to effect global listings and transfers.



TECHNICAL NOTES

By The RSTS "Pro" Staff

You will recall that in the last article we reviewed methods of doing input and output to simple and more complicated devices attached to the PDP 11. The data transfers were accomplished simply by loading the appropriate registers with information and then checking other status registers to determine what had happened to that information. The simplest example was a terminal which has a control and status register and a data register. I/O can be accomplished simply by putting data into the register then checking the control status register to make sure that the data has either been read or written.

It is possible to transfer data using interrupt structure available on the PDP 11. Using the control and status register, it is possible to set the interrupt enable bit which then causes a processor interrupt when data is either read or written. This allows the processor to go about other business without having to be constantly looking or polling the terminals to determine if there is input or output to be processed.

The next level of input/output also utilizes the interrupt facility, but in this case, more than one byte or word of data can be transferred at a time without involving the processor. This was the so-called cycle stealing mode. In this mode the processor steals a cycle or cycles from whatever programs happen to be operating and loads into the various registers, information necessary to transfer one or more bytes of data; then returns to the program. When the data has either been successfully transferred or an error condition has arisen, an interrupt occurs from the device. The processor can then either steal more cycles in order to reload the registers to continue the transfer or go off to an error processing routine to process the error condition.

By way of an example, let us examine in more detail how a multiple data transfer might be accomplished from a specific location in memory onto magnetic tape utilizing the TU 10 magnetic tape drive. The PDP 11 Peripherals Handbook describes the operation beginning on page 4-501 with a description of the TM 11 controller and the TU 10 drive. The first register in the TM 11 is the status register. Each of the 16 bits in this register indicate the status of the particular drive in question. These include a bit for illegal command, end of file, cyclical redundancy check, parity error, bus grant late error, end of tape, record length error, bad tape error, non-existent memory error, select tape, beginning of tape, whether it's a 7 or 9 channel tape, whether the tape is slowing down, whether the tape is write-locked, the current rewind status of the drive, whether the tape unit is ready. In addition to the status register, there is a command register. This 16 bit register is used to set information necessary for the data transfer. It is possible to set the density of the tape, to clear the tape drive, set the parity, select one of 8 units, enable the interrupt, set a particular function; that is, read or write, space forward or space backward. The final bit is a go bit telling the drive controller to begin the operation. A third register in the TM 11 is a byte record counter. This is simply a 16 bit binary counter which is used to count bytes in a read/write operation. This register should initially be set by the program to the twos compliment of the number of bytes to be written on the tape. When this register becomes zero after the last byte of the record has been read or written, a control signal (interrupt) is sent indicating that there are no more data characters to be read or written from the tape. In addition to the number of bytes necessary to be transferred, a fourth register, the current memory address register, contains the memory address at which the transfer operation is to begin. Although this is a 16 bit register, it is possible to address 18 bits of memory. That is because in the command register bits 5 and 4 contain the two high order bits necessary for addressing the 18 bit address space. There are an additional two registers, a data buffer and read lines register, also in the TM 11.

In general, the following steps would be carried out in order to transfer data from memory to the TU 10 mag tape. First, using the command register, we would select the unit that we are planning to use. Selection is simply made by loading bits 10, 9 and 8 with the drive number we plan to use. After the unit has been selected, the status register is checked to find out the current status of the tape, and back once again to the command register to enable the interrupt. Since we want to write the data on the beginning of the tape, we should at this point rewind it. Using the command register, bits 3, 2 and 1 are function bits. Setting these bits all

DEAR RSTS MAN

. . . continued from page 16

In the last issue *Peter Dick, Silver Programs, London, slipped a question into the "Letters to the RSTS Pro" column. RSTS MAN caught it and now answers.*

The question . . . How does one reset the statistics without the switch register - i.e., after RDC have nicked it?

The answer . . . *The switch register is accessible through console emulator commands. Specifically, R will read the contents of the switch register while nnnnnnW will write n8 into the switch register. Resetting statistics requires switches 15 and 0 to be set to 0. Therefore, in console mode 0W will write all zeros into the switch register, resetting statistics (and turning them off). 777777W will put all ones into the switch register, starting statistics gathering again.*

Send questions to: DEAR RSTS MAN, P.O. Box 361, Fort Washington, PA 19034.

to 1 indicates that the function we wish the tape to execute is the rewind function. Finally, by setting bit 0 in the command register to 1, the tape will begin to rewind. It is not necessary for the CPU to wait while the tape rewinds since when the tape is rewound (we have enabled the interrupt), the TM 11 will interrupt the CPU and the status register will indicate that the tape has been rewound. Now, seeing that the tape is ready by looking in the status register, we would set the density and parity to the desired numbers, load the number of bytes that we wish to transfer into the byte record counter, set the memory address at which we wish this transfer to begin in the memory address register, including the two bits that are contained elsewhere, if necessary, set the function bits to 0, 1 and 0 for bits 3, 2 and 1, respectively, indicating a write operation. Then, finally, set the go bit to 1 and the function will begin. Once again the processor goes off to do other functions. When the TM 11 is finished, it will interrupt the processor. If the data we wish to transfer is not in contiguous memory locations, when the current transfer is finished, the TM 11 interrupts the processor which then steals a cycle to load the next memory address into the memory address register, the number of bytes to be transferred into the byte record counter, sets the go bit and then returns from the interrupt. Thus, many data transfers can be made from different memory locations by the TM 11 utilizing only a few cycles of the CPU to effect these transfers.

What happens internally during this transfer of data from memory to the TU 10? Over what paths does the data flow? Over what paths do the control signals flow? How does the interrupt get back to the processor? These paths can vary depending upon the peripheral device and the PDP 11 model in question. Continuing to utilize our TU 10 example, the TU is connected to the unibus of the PDP 11. Also connected to the unibus are the processor, memory, terminals, disks and many other types of input and output equipment. All data that flows from memory to the TU 10 flows via the unibus. This includes control signals and interrupt signals. Communication between the two devices on the unibus can be controlled by those two devices themselves. When the TM 11 is controlling transfer between the TU 10 and memory, it is in control of the unibus. Since the unibus is used by the processor in many I/O devices, a priority structure is used to determine which device can obtain control of the bus at any one time. By now we should know that all devices attached to the unibus have an address, (maybe more than one address), an interrupt vector, as described

TTOPNF.MAC & TTDVR.TEC

. . . continued from page 35

```
W$NVBM: .BLKB               ;NEXT VIRTUAL BLOCK MSB
W$NVBL: .BLKW               ;NEXT VIRTUAL BLOCK LSB
W$FCB:  .BLKW               ;FCB @ F$CLUS
W$REN:  .BLKW               ;RETREIVAL ENTRY NUMBER
W$WCB:  .BLKW               ;NEXT WCB THIS FCB + FLAGS
W$NXT:  .BLKW               ;FBB OF NEXT RETREIVAL WINDOW
        .BLKW               ; (FROM ABOVE)
W$WND:  .BLKW               ;CURRENT WINDOW
        .BLKW               ;WORD TWO OF WINDOW
        .BLKW               ;WORD THREE OF WINDOW
        .BLKW               ;WORD FOUR OF WINDOW
        .BLKW               ;WORD FIVE OF WINDOW
        .BLKW               ;WORD SIX OF WINDOW
        .BLKW               ;WORD SEVEN OF WINDOW

.BSECT                      ;BITS FOR FILE STATUS

US.OUT: .BLKB   .           ;FILE IS PHYSICALLY ON ANOTHER DISK
US.PLC: .BLKB   .           ;FILE HAS BEEN PLACED
US.WRT: .BLKB   .           ;FILE IS OPEN FOR WRITE ACCESS
US.UPD: .BLKB   .           ;FILE IS OPEN FOR UPDATE
US.NOX: .BLKB   .           ;FILE IS CONTIGUOUS
US.NOK: .BLKB   .           ;FILE CANNOT BE KILLED DURING TIMESHARING
US.UFD: .BLKB   .           ;NAME ENTRY IS A UFD
US.DEL: .BLKB   .           ;FILE IS MARKED FOR DELETION

.BSECT                      ;BITS FOR W$STS OR DDSTS

        .BLKB   .           ;OPEN NON-FILE STRUCTURED
        .BLKB   .           ;READ-LOCK ON FILE
        .BLKB   .           ;WRITE-LOCK ON FILE
WC$UPD: .BLKB   .           ;UPDATE MODE ON FILE
WC$CTG: .BLKB   .           ;FILE IS NO EXTEND (CONTIGOUS)
WC$LCK: .BLKB   .           ;LOCK IS ON IN FILE
WC$UFD: .BLKB   .           ;FILE IS A UFD
WC$USE: .BLKB   .           ;WRITE PRIVS GRANTED

.SBTTL  TTOPNF - PRINT OPEN FILES FOR USER
;+
; TTOPNF - PRINT OPEN FILES FOR USER
;
; CALL: R1 -> TO CONSOLE DDB
;
; BACK: C-BIT SET IF CAN'T DO
;-

        DEFORG  TTOPNF

TTOPNF::MOVB    DDJBNO(R1),R3   ;;GET OWNING JOB # *2
        BIC     #^C<63.*2>,R3   ;;AND ENSURE A VALID JOB # *2
        BEQ     10$             ;;NO JOB, NO OPEN FILES
        MOV     JOBTBL(R3),R3   ;;GET JOB DATA BLOCK POINTER
        BEQ     10$             ;;NO JDB, NO OPEN FILES
        BIT     #JFPRIV,JDFLG(R3) ;;ARE WE PRIVILIGED?
..PRVF  ==      .               ;;** PATCH ** TO 'NOP' SO NON-PRIV CAN DO
        BEQ     10$             ;;NO, DON'T SHOW OPEN FILES
        BIT     #DDCONS,DDCNT(R1) ;;REALLY CONSOLE DEVICE FOR JOB?
        BNE     DOOPNF          ;;YES, GO SHOW OPEN FILES
10$:    SEC                     ;;MARK AS NO GOOD
        RETURN                  ;; AND RETURN

GLOBAL  <JOBTBL>

.ENABL  LSB

UFDMSG::.ASCIZ  %USER FILE DIRECTORY%<11>
NFSMSG::.ASCIZ  <11><11>%NON-FILE STRUCTURED%<11>
.EVEN   ;THIS COULD BE ANNOYING

DOOPNF: MOV     R5,-(SP)        ;;SAVE AN IMPORTANT REGISTER
        CMPB    DDHORZ(R1),DDHORC(R1) ;;IS CARRIAGE CURRENTLY RESTORED?
        BEQ     10$             ;;YES
        MOV     R3,-(SP)        ;;SAVE THIS POINTER
        CALLX   ASCOUT,R5,CRLF.0 ;;NO, SO RESTORE THE CARRIAGE
        MOV     (SP)+,R3        ;;GET IT BACK
10$:    MOV     JDIOB(R3),R3    ;;POINT TO THEIR I/O BLOCK
        TST     (R3)+           ;;SKIP THEIR CONSOLE TERMINAL
        CLR     R5              ;;LET R5 BE THE CHANNEL COUNTER
20$:    CALL    30$             ;;OUTPUT NEXT CHANNEL
        CMP     #15.,R5         ;;PAST LAST CHANNEL
        BGT     20$             ;;NOT YET
        CALLX   ASCOUT,R5,CRLF.0 ;;TO MAKE IT LOOK GOOD
        MOV     (SP)+,R5        ;;RESTORE THE IMPORTANT REGISTER
        CLC                     ;;SAY DONE
        RETURN                  ;; AND EXIT

30$:    INC     R5              ;;SET NEXT CHANNEL NUMBER
        MOV     (R3)+,R4        ;;GET POINTER TO WCB
        BEQ     40$             ;;NOTHING THERE, GO FOR NEXT
        BIT     #1,R4           ;;ODD ADDRESS?
        BNE     40$             ;;THAT'S ODD, QUIT
        BR      50$             ;;CONTINUE
40$:    RETURN                  ;;PRE-MATURE RETURN
50$:    MOV     R3,-(SP)        ;;SAVE THIS POINTER
```

in the first Newsletter article, and a unibus priority which is known as bus request level.

The maximum transfer rate on the unibus using optimum device design would be one 16-bit word every 400 nanoseconds or 2.5 million 16-bit words per second. Although at first glance this seems very rapid, remember that the PDP 11 processors are capable of executing instructions in close to 1 microsecond. Assuming that each instruction that the processor executes has to be gotten from memory, as would be the case in a program, each of these instructions took 1 microsecond to execute. One out of every three words transferred via the unibus would consist of the processor getting its instructions from memory, leaving only 66% of the unibus available for transfer. The word "only" here means approximately 1.6 million words per second are transferrable via the unibus. Of course, the device is never optimum and considering that one RP04 is capable of transferring close to 400,000 words per second, it is possible to see how the bus could become the limiting factor in data transfers on a heavily I/O bound PDP 11. Referring now to the PDP 11/70 Processor Handbook, figure 1.1 on page 1-2 indicates a slightly different architecture available for this model of the PDP 11. In addition to the regular unibus peripherals, which in this case could include terminals and TU tape drives, there are four high-speed controllers which interface themselves to the unibus. However, the only signals that pass over the unibus are the control signals and status and interrupt signals as described earlier. The actual transfer of data occurs via special high-speed I/O bus to the mass storage peripheral, such as in RP04 or a TU16 tape and via another high-speed bus directly linked to a special memory bus through the cache memory of the PDP 1170. This relieves the unibus of the high-speed data transfers between memory and peripheral devices. In this case at least, data can be transferred simultaneously via the unibus and the high-speed controller bus, sometimes called a mass bus, to the high-speed peripherals.

Since we have reviewed earlier input and output operations via TU 10 and TM 11 controller, it is interesting to compare the similarities and differences between this tape drive and a TJU 16 controller and TU 16 tape which operate utilizing a high-speed controller attached to the separate I/O bus and high-speed bus directly to the memory.In the first control and status register, bit 13 is set by a parity error on the control bus although the control bus is not the high-speed bus going to the high-speed device itself. Parity errors occurring on that bus are indicated in another register. The second register is a word count register

```
        MOV     R5,-(SP)            ;;SAVE THIS COUNTER
        MOV     R4,-(SP)            ;;SAVE POINTER TO WCB
        MOV     R5,-(SP)            ;;PUT CHANNEL NUMBER ON STACK
        CALL    OUTSIZ              ;;OUTPUT THE CHANNEL NUMBER
        CALL    OUTCHR,R5,<11>      ;;NOW OUTPUT A <TAB>
        MOV     (SP),R4             ;;GET WCB POINTER BACK
        MOVB    W$IDX(R4),R2        ;;IS THIS A DISK?
        BEQ     60$                 ;;YEP
        BIT     R2,#1               ;;BOGUS INDEX?
        BNE     120$                ;;QUIT
        MOV     R2,R5               ;;COPY DEVICE INDEX *2
        ADD     #DEVNKB-2,R5        ;;POINT TO NAME IN "DEVNAM"
        MOVB    DDUNT(R4),-(SP)     ;;SAVE THE UNIT NUMBER
        CALL    OUTPNT              ;;OUTPUT DEVICE NAME
        CALL    OUTSIZ              ;;AND UNIT NUMBER
        BR      110$                ;;AND CONTINUE
60$:    MOV     W$FCB(R4),R4        ;;POINT TO FCB FOR THIS FILE
        MOVB    F$UNT-F$CLUS(R4),R4 ;;GET THE FIP UNIT FOR THIS FILE
        MUL     #10.,R4             ;;COMPUTE OFFSET INTO LOGICAL NAME TABLE
        ADD     #LOGNAM+6,R5        ;;POINT TO UNIT NUMBER
        MOVB    (R5),-(SP)          ;;SAVE THE UNIT NUMBER
        TST     -(R5)               ;;POINT BACK ONE WORD
        CALL    OUTPNT              ;;PRINT OUT DISK NAME
        CALL    OUTSIZ              ;;PRINT OUT THE UNIT NUMBER
        MOV     (SP),R4             ;;GET BACK WCB POINTER
        MOV     W$FCB(R4),R4        ;;POINT TO FCB AGAIN
        SUB     #F$CLUS,R4          ;;POINT TO TOP OF WINDOW
        MOV     F$STAT(R4),-(SP)    ;;SAVE FILE STATUS
        MOV     F$CLUS(R4),-(SP)    ;;SAVE CLUSTERSIZE
        CLR     -(SP)               ;;FOR MSB AS CLUSTERSIZE
        MOV     F$SIZL(R4),-(SP)    ;;SAVE FILE SIZE LSB
        MOVB    F$SIZM(R4),-(SP)    ;;SAVE SIZE MSB
        CMP     -(SP),-(SP)         ;;SAVE TWO SPOTS FOR VITUAL BLOCK NUMBER
        BIT     #DDNFS,@16(SP)      ;;OPEN NON-FILE STRUCTURED?
        BEQ     70$                 ;;GO ABOUT OUR BUSINESS IF NOT
        CALLX   ASCOUT,R5,NFSMSG    ;;PRINT OUT THE MESSAGE
        BR      100$                ;;AND GO ON OUR MERRY WAY
70$:    MOVB    F$PROT(R4),-(SP)    ;;GET PROTECTION CODE
        ADD     #F$NAM,R4           ;;POINT TC FILENAME
        MOV     R4,-(SP)            ;;SAVE THE POINTER
        TST     -(R4)               ;;SKIP THE WORD
        MOV     (R4),-(SP)          ;;SAVE THE PPN
        MOV     (SP),-(SP)          ;;COPY IT TWICE
        SWAB    (SP)                ;;SWAP FOR PROJECT NUMBER
        CALL    OUTCHR,R5,<'[*400+' > ;;NOW OUTPUT A <SP>"[
        CALL    OUTSIZ              ;;OUTPUT THE PROJECT NUMBER
        CALL    OUTCHR,R5,<',>      ;;AND THEN A COMMA
        CALL    OUTSIZ              ;;THEN PROGRAMMER NUMBER
        CALL    OUTCHR,R5,<11*400+']> ;;THEN OUTPUT ]<TAB>
        MOV     (SP)+,R5            ;;POINT TC FILE NAME
        BIT     #<WC$UFD*400>,@20(SP) ;;WAS IT JUST A UFD?
        BEQ     80$                 ;;NO, TAKE NO SPECIAL ACTION
        TST     (SP)+               ;;SKIP THE PROTECTION CODE
        CALLX   ASCOUT,R5,UFDMSG    ;;AND SAY IT'S A UFD
        BR      100$                ;;AND CONTINUE
80$:    CALL    FILNAM              ;;GO PRINT OUT FILENAME
        BIT     #US.DEL,16(SP)      ;;ARE WE DELETED?
        BEQ     90$                 ;;NOPE
        CALL    OUTCHR,R5,<'*>      ;;OUTPUT A STAR
90$:    CALL    OUTCHR,R5,<60.*400+11> ;;NOW OUTPUT <TAB><
        CALL    OUTSIZ              ;;PRINT PROTECTION CODE
        CALL    OUTCHR,R5,<11*400+62.> ;;NOW OUTPUT ><TAB>
100$:   MOV     16(SP),R4           ;;RESTORE WCB POINTER
        MOV     W$NVBL(R4),2(SP)    ;;SAVE VIRTUAL BLOCK LSB IN STACK
        MOVB    W$NVBM(R4),(SP)     ;;SAVE VIRTUAL BLOCK MSB IN STACK
        CALL    DBLNUM              ;;OUTPUT NEXT BLOCK
        CALL    OUTCHR,R5,<11>      ;;NOW OUTPUT A <TAB>
        CALL    DBLNUM              ;;OUTPUT FILESIZE
        CALL    OUTCHR,R5,<11>      ;;NOW ANOTHER <TAB>
        CALL    DBLNUM              ;;OUTPUT THE CLUSTER SIZE
        CALL    FILSTA              ;;DO FILE STATUS
110$:   CALLX   ASCOUT,R5,CRLF.0    ;;FINALLY RESTORE CARRIAGE FOR THIS TIME
120$:   TST     (SP)+               ;;SKIP WCB POINTER
        MOV     (SP)+,R5            ;;RESTORE CHANNEL POINTER
        MOV     (SP)+,R3            ;;RESTORE IOB POINTER
        RETURN                      ;;EXIT SUBROUTINE

GLOBAL  <CRLF.0,LOGNAM,DEVNKB>

.DSABL  LSB

.SBTTL  PRINT OUT FILE STATUS FLAGS
;+
; CALL FILSTA
;
; SP -> FILE STATUS BITS
;
; R4 GETS CLOBBERED
;-
.ENABL  LSB

FILSTA: MOV     (SP),R4             ;;SAVE RETURN ADDRESS
        MOV     2(SP),(SP)          ;;RESET AS FILE STATUS
        MOV     R4,2(SP)            ;;AND RESET RETURN ADDRESS
        CMP     #US.WRT,(SP)        ;;IS THERE STATUS?
        BEQ     50$                 ;;NOPE, SKIP IT
        CALL    OUTCHR,R5,<11>      ;;AND YET ANOTHER <TAB>
```

DATA BOSS/2 USERS:

LET DBQ break the *CHAINing* Habit
and release the full power of your DBMS

THE DBQ extensions to DB2 let you

ENTER
REMOVE
REPCHN

records ***DIRECTLY*** from your Basic Program

and **ALSO** provide:

Logical record locking
Journaling
Corruption proof
& and , (comma) permitted in key fields

The DBQ module RTV (an NRTRV1 replacement) allows user programs to enter or REMOVE records and change key fields (REPCHN) WITHOUT writing A.S.C.I.I. files and chaining to a DB2 utility.
The RTV module is compatible with NRTRV1 and olds (Basic +) at just 6K, and expands to only 8K with 5 open databases (maximum 9K).

Disk journal and audit trail can be selected for any database.

Databases updated with RTV cannot be corrupted under normal system conditions including powerfails.

Sort keys (except DRK) can have & and , (comma) characters

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)634-8114

(305)447-2695

which operates exactly like the matching register in the TU 10. Also, the unibus address register, which really is a memory address register, operates in a similar function to the one in the TU 10, even to the degree that the high order two bits are contained in the control and status register; one in order to give 18 bit addressing capability. While spacing forward records or characters was controlled in the TU 10 in a different fashion, the TU 16 has a different frame count register which contains a twos compliment number of records to be spaced over, characters to be written or characters that have been read. There is a second control and status register. This register contains some additional information including bit 8 which is the mass I/O bus data bus parity error. This is the error that gets set on a parity error indication on the high-speed bus. The drive status register contains similar information, some additional peripheral information. One of the additional capabilities that the TU 16 has on an optional basis is to write in phase encoded 1600 bits per inch. There is an error register in which each of the 16 bits contains a certain particular error. An attention summary register in the controller enables the programmer to determine immediately which of the 8 drives has interrupted rather than looking at the status of all 8 drives of the sequence. A character check register contains the cyclic redundancy check character that is used on the TU 16. There is the same data buffer register as well as a drive type register which allows the program to find out what type of drive this is. There are also serial number registers containing the serial number of the drive and a tape control register to control the density and format of the data being written. So, although the TU 16 has a few more registers than the TU 10, most of the additional data available is better error checking, increased error diagnostic aids and little, if any of it, has to do with the difference in architecture between the way the data is transferred via a high-speed controller. Thus, the fact that the TU 16 can operate via a mass bus and a high-speed controller is transparent to the programmer who has to program applications for it. ♥


```
        BIT     #US.NOK,(SP)     ;;PERMANENT FILE?
        BEQ     10$              ;;NOPE
        CALLX   OUTCHR,R5,<'P>   ;;SAY SO
10$:    BIT     #US.NOX,(SP)     ;;CONTIGUOUS?
        BEQ     20$              ;;NOPE
        CALLX   OUTCHR,R5,<'C>   ;;SAY SO
20$:    BIT     #US.UPD,(SP)     ;;OPEN FOR UPDATE?
        BEQ     30$              ;;NOPE
        CALLX   OUTCHR,R5,<'U>   ;;SAY SO
30$:    BIT     #US.UPD!US.WRT,(SP) ;;WE HAVE WRITE PRIVELEGES?
        BNE     40$              ;;YES, SO IGNORE
        CALLX   OUTCHR,R5,<'R>   ;;IF NOT, SAY READ ONLY
40$:    BIT     #US.PLC,(SP)     ;;ARE WE A PLACED FILE
        BEQ     50$              ;;NOPE
        CALLX   OUTCHR,R5,<'L>   ;;SAY SO
50$:    TST     (SP)+            ;;GET RID OF STATUS
        RETURN                   ;;AND GO BACK

.DSABL  LSB

.SBTTL  PRINT OUT SINGLE OR DOUBLE INTEGER
;+
;DBLNUM - PRINT OUT DOUBLE PRECISION INTEGER
;       WITH OR WITHOUT LEFT JUSTIFICATION
;
;       SP -> LSB OF INTEGER, MSB OF INTEGER
;
;       CALL    DBLNUM
;
;       R2 - R5 RANDOM
;-
.ENABL  LSB

DBLNUM: MOV     (SP)+,R5          ;;SNAG RETURN ADDRESS
        CLR     R2                ;;CLEAR A HIGH ORDER
        BISB    (SP)+,R2          ;;GET THE HIGH ORDER OF THE INTEGER
        MOV     (SP)+,R3          ;;GET THE LOW ORDER
        MOV     R5,-(SP)          ;;REPLACE THE RETURN ADDRESS
        DIV     #10000.,R2        ;;SPLIT THE NUMBER
        MOV     R3,-(SP)          ;;SAVE LOW ORDER
        TST     R2                ;;IS THERE A HIGH ORDER?
        BEQ     10$               ;;NO HIGH ORDER
        MOV     R2,R3             ;;SET TO OUTPUT IT
        CALL    20$               ;;DO IT
        SEC                       ;;LOW ORDER SHOULDN'T SUPRRESS LEADING ZEROES
10$:    MOV     (SP)+,R3          ;;DO LOW ORDER NOW
        CALL    20$               ;;OUTPUT THE LOW PART
        RETURN                    ;;AND GO BACK

20$:    CALL    DODIVS,R5,30$     ;;GO DO THE DIVISIONS, PRINTING ASCII DIGITS
         .WORD  1000.,100.,10.
        RETURN                    ;;EXIT

30$:    MOVB    50$(R2),R2        ;;GET THE NUMERIC
40$:    CALLRX  CHOUTE            ;;OUTPUT THE CHARACTER

        .ASCII  "$.?"
50$:    .ASCII  "0123456789"      ;;NUMBERS

        .EVEN

;+
; FILNAM - PRINT OUT A FILENAME
;
;       R5 -> TO THREE FILENAME WORDS
;
;       CALL FILNAM,R5
;
;-

FILNAM: MOV     #70$,-(SP)        ;;PUSH FOR EXTENSION
        MOV     PC,-(SP)          ;;BACK HERE ONCE
60$:    MOV     (R5)+,R3          ;;GET THE RAD50 WORD
        SEC                       ;;DON'T SUPRESS LEADING ZEROES
        CALL    DODIVS,R5,80$     ;;GO DO THE DIVIDES
          .WORD 50*50,50
        RETURN                    ;;EXIT

70$:    MOV     #'.,R2            ;;SET TO OUTPUT A "."
        CALLX   CHOUTE            ;;AND CALL ROUTINE TO DO IT
        BR      60$               ;;AND CONTINUE

80$:    TST     R2                ;;CHECK FOR A SPACE
        BEQ     90$               ;;OUTPUT A SPACE
        ADD     #'A-1,R2          ;;ADJUST FOR ALPHABETICS
        CMP     R2,#33+<'A-1>     ;;IS IT IN FACT ALPHABETIC?
        BLO     40$               ;;YES, GO USE IT
        SUB     #36+<'A-1>,R2     ;;ADJUST FOR NUMERIC (ZERO = 0)
        BR      30$               ;;GO DO IT

90$:    MOV     #' ,R2            ;;SET SPACE AS CHARACTER
        BR      40$               ;;AND OUTPUT IT

;+
; OUTSIZ - PRINT A NUMBER 0-255 WITH LEADING ZERO SUPPRESSION.
;
;        SP -> NUMBER (HIGH ORDER BYTE IGNORED), ...
```

The RDC Revisited

By Carl Marbach

Some months ago I wrote an editorial about some bad experiences I had with Digital's Remote Diagnosis Center. For the uninitiated, the DDC (Digital Diagnosis Center) is a facility of DEC that is set up to do Remote Diagnosis on 11/44's, 11/70's, and VAX's. They accomplish this feat by dialing up your computer's console from Colorado Springs, Colorado. Once connected they do strange and unintelligible (until now) things to your computer. Rumor has it that they then call your branch field service people and send the engineer with the right parts to fix your problem. Sounds good? Works even better provided you follow some ground rules, and are willing to help out.

I was invited to visit the DDC by Al Toussach, who is in charge of Field Service Marketing in Maynard. Now if you're like me, you didn't know that DEC marketed Field service. Al helps to sell DECsystems by selling Field service. No one ever sold me field service; in fact it is the other way around — I sell them. You see I like my branch, and the branch I had at another site; they are responsive, helpful, kind, and hard working. I am always proud to point to the 15 years I have worked with DEC computers, and I can say that I have had a perfect record with field service (I do seem gruff at times, and I hate it when my machine is down). If I have had a problem, they tried to solve it. A case in point is the RDC (now DDC) problem I had. We solved it. Me, my branch, and the DDC. I would like to tell you now what I learned so your problem can be solved before it happens.

First, the DDC is not located near SAC, nor is it buried deep inside a mountain. It is just off the interstate on a beautiful site of many, many, many acres that goes right up to the Rocky mountains themselves. It is

```
;
;       CALL    OUTSIZ
;
;       R2-R4 = RANDOM
;       SP -> ...
;-

OUTSIZ: CLR     R2                      ;;CLEAR THE BUCKET
        BISB    2(SP),R2                ;; AND GET THE K SIZE
        MOV     (SP)+,(SP)              ;;MOVE THE RETURN ADDRESS DOWN
        ;CLC                            ;;C-BIT=0 FROM 'CLR' ABOVE
100$:   MOV     R2,R3                   ;;COPY THE NUMBER
        CALL    DODIVS,R5,30$           ;;GO DO THE DIVISIONS, PRINTING ASCII DIGITS
          .WORD 100.,10.
110$:   RETURN                          ;;EXIT


;+
; OUTCHR - PRINT ONE OR TWO CHARACTERS.
;
;       CALL    OUTCHR,R5
;        .BYTE  BYTE1,BYTE2     (BYTE2 NOT PRINTED IF 0)
;
;       R2-R5 = RANDOM
;-

OUTCHR: MOV     R5,(SP)                 ;;CLOBBER SAVED R5 WITH RETURN ADDRESS -2
        ADD     #2,(SP)                 ;; THEN UPDATE TO REAL RETURN ADDRESS

;+
; OUTPNT - PRINT ONE OR TWO CHARACTERS (VARIABLE).
;
;       R5 -> 2 BYTES TO PRINT
;
;       CALL    OUTPNT
;
;       R2-R4 = RANDOM
;       R5 -> PREVIOUS + 2
;-

OUTPNT: MOVB    (R5)+,R2                ;;GET FIRST CHARACTER
        CALLX   CHOUTE                  ;; AND OUTPUT IT
        MOVB    (R5)+,R2                ;;GET SECOND CHARACTER (IF ANY)
        BIC     #^C<177>,R2             ;; AND TRIM THE SIGN BIT
        BNE     40$                     ;;OUTPUT SECOND CHARACTER IF ONE EXISTS
        RETURN                          ;;ELSE JUST EXIT

.DSABL  LSB

;+
; DODIVS - DO A SERIES OF DIVISIONS AND EMIT THE QUOTIENTS.
;
;       R3 = NUMBER TO DIVIDE
;       C=1 IF LEADING ZEROS ARE TO BE PRINTED
;
;       CALL    DODIVS,R5
;        .WORD  ROUTINE TO CALL TO EMIT A QUOTIENT
;               (QUOTIENT IS IN R2, MAY NOT ALTER R0, R1, OR R5)
;        .WORD  DIVISOR 1, DIVISOR 2, ...
;       RETURN
;
;       R2-R4 = RANDOM
;
;       NOTE: (DIVISOR N & 200) MUST BE 0 FOR N >= 2
;-
.ENABL  LSB

DODIVS: MOV     (R5)+,-(SP)             ;;SAVE THE OUTPUT ROUTINE ADDRESS
        BIC     -(SP),(SP)              ;;SAY NOT INTO SIGNIFICANCE YET
        ADC     (SP)                    ;;FORCE SIGNIFICANCE IF NEEDED
10$:    CLR     R2                      ;;CLEAR A HIGH ORDER
        DIV     (R5)+,R2                ;; AND DIVIDE
        BNE     20$                     ;;RESULT <> 0, SAY SIGNIFICANCE STARTS
        TST     (SP)                    ;;RESULT = 0, IS IT SIGNIFICANT?
        BEQ     30$                     ;;NOT YET...
20$:    INC     (SP)                    ;;INDICATE RESULTS ALL NOW SIGNIFICANT
        MOV     R3,-(SP)                ;;SAVE REMAINDER
        CALL    @4(SP)                  ;;CALL THE OUTPUT ROUTINE
        MOV     (SP)+,R3                ;;RESTORE REMAINDER
30$:    TSTB    (R5)                    ;;MORE?
        BPL     10$                     ;;YES, LOOP
        MOV     R3,R2                   ;;NO MORE DIVISIONS, SET FINAL RESULT
        TST     (SP)+                   ;;DUMP THE SIGNIFICANCE INDICATOR
        CALL    @(SP)+                  ;;GO OUTPUT FINAL RESULT
        RETURN  R5                      ;; AND EXIT

.DSABL  LSB
.END

@:EB"TTDVR.MAC" "U @^A"%CAN'T FIND TTDVR.MAC - ABORTING" 13^T 10^T EX'
ET&512 "N 155^T @^A"[?2L" 155^T 72^T 155^T 74^T'
@^A"FOUND TTDVR.MAC - WORKING" 13^T 10^T
@N"ORG  TTSYST" -1W L @I"
        ORG     TTOPNF                  ;OPEN FILES CODE
" -1W
@N"TI$CRR::" @-S"TI$CTT" -1W L @I"; TI$CFF - ^F HANDLER
" -1W
```

nice. Why here? Well, telephone charges are minimized because it is very close to the center of the U.S. But they admit that it is easier recruiting people to the quality of life available in the Rocky Mountain foothills. There is also unlimited expansion available on the large site they occupy. The DDC shares the facility with the telephone support group and a disk manufacturing facility. Growth has been so rapid that there are signs of people being moved all over; new desks, new terminals (some in boxes), new walls, and new people.

```
@S"TI$CTT::" L -1W
@I"TI$CFF::                               ;;HANDLE CONTROL/F (OPEN FILES)
" -1W
@S"BNE  40$" @S";;" -1W K
@I"NOPE, IT IS CONTROL/T OR CONTROL/F
" -1W
@S"40$:" -1W @I"        CMP     R2,#'F-100      ;;CONTROL/F MAYBE?
        BEQ     60$                     ;;YEP, HANDLE IT
" L @I"50$:" 3L @I"
60$:    CALL    MAPPED,R5,TTOPNF ;;GO TRY FOR OPEN FILES
        BR      50$                     ;;DO SAME AS CONTROL/T
" -1W @S"TTSYST:" 3L -1W @I"    TMPORG  TTOPNF
TTOPNF: SEC                             ;;SAY ILLEGAL UNLESS OVERLAID
        RETURN                          ;; AND EXIT
        UNORG

" -1W @N".WORD  'G-100" OL -1W
@I"     .WORD   'F-100                  ;CONTROL/F (OPEN FILES)
" -1W @N".WORD  TI$CCG"  OL -1W @I"     .WORD   TI$CFF          ;CONTROL/F
" -1W
ET&512 "N 155^T 72^T 155^T 74^T'
@^A"CLOSING FILE OUT ..." 13^T 10^T EC
ET&512 "N 155^T 72^T 155^T 74^T'
@^A"FINISHED..." 13^T 10^T EX
```

I met Al in Denver and we drove the one and one half hours to 'the springs'. The DDC itself is spectacular, it sits alone on what appears to be miles of land mostly covered by low vegetation. At the end of the land are the foothills of the Rockies and the Rockies themselves with Pikes Peak standing out like a sore thumb that isn't sore; just pretty to look at. I suppose they get used to it, but guys, it sure is nice looking.

Once inside I met Bob Ross, fugitive from Detroit who is PDP-11 family manager (see corporate chart). Bob suggested that before we went into detail about the DDC itself that I might like a tour of the building and the disk manufacturing facility it contains. Warren Shubert was our guide through a plant that manufactures RK07's, pseudo-manufactures RM03's and RM05's (really CDC) and RP06's (memorex). They are also making the disk packs for the RK07's in a 'clean room'. In another 'clean' room they are making winchester media, this room is 1000 times cleaner than the RK07 room! The RP07 winchester (available for VAX) looks like a state of the art drive, with a monthly service cost less than the RP06 which holds less. The manufacturing system is RSTS of course (we're moving to VAX. . .) and tracks the process from receiving to final production and testing. Testing is done en masse with many smaller 11's, and hundreds of drives seeking forever (on a clear disk you can seek forever). Warren showed us the new packaging and shipping area that had recently been overhauled. They had done a study and found that this area was the bottleneck in production. A new system of wrapping, stacking, and moving these large packages around was installed and . . . now there is another bottleneck somewhere else. Slick this is, a small person (female even) can move huge amounts of disk packs around in a flash. All this confirms how little I know about manufacturing, and at the same time how similar manufacturing is to a complicated program. Maybe if I figured out where the bottleneck was in my trial balance I could. . .well, thanks to Warren and we went back upstairs.

Bob and I sat down and went over some of the goals of the DDC. The main objective of the DDC was to:

- Reach a diagnostic conclusion with a recommen--dation within 1 hour
- Achieve this in 90% of the cases
- Continue to work the problem after 1 hour upon branch request

After these the DDC also:

- Can perform/help in PM
- Installation check-outs
- Branch Demo's (on request)

According to Bob, they are getting close to achieving the 90% effectiveness that they are shooting for. To you and me this means that they will make the correct recommendation to the branch for repair within 1 hour of your call about 90% of the time. Great.

In addition there is a 'hot call' list. 'Hot calls' are for VIP's, loud yellers, intermittent problems, and continuing problems. You can be put on this list by your branch, who will then agree on a plan of action with the DDC. You should be an integral part of this plan; if they don't ask you, then ask them. Remember, its your machine. Once you are flagged on their data base as a 'hot call', the DDC will continue to follow up on the problem, i.e. if they don't hear from the branch or you for a week they will call to find out what's happening.

Why does the DDC work? Mostly because of the people who actually do the diagnosis (they get help from the computer of course). Jim Porter gave us a demonstration of exactly how the diagnosis is done. The computer is an 11/70 (they have four of them) running RSTS (we're moving to VAX). Jim sat at a VT100 and paged through their data base showing me various installations including mine. All necessary data is stored including your configuration, telephone number, contact, contact number, and DDC history. The engineer can look at all this to determine how to proceed. After looking up your configuration, Jim asks the computer (DDC) to connect to the remote location; connection and dial-out (Vadic auto-dialers) is automatic. Once connected he can instruct the computer to run through several 'scripts' of diagnostic sessions. There is a general check-out script as well as many specialized ones. Jim and his fellow engineers know these systems inside and out, you are guaranteed to get an expert. Nice is a keyword here, but then when they go to lunch they look out at Pikes Peak and that is bound to help even my disposition.

The calls are initially taken by the phone answering group, who are trained to route the calls to the proper engineers, and to make you feel good about talking to the DDC.

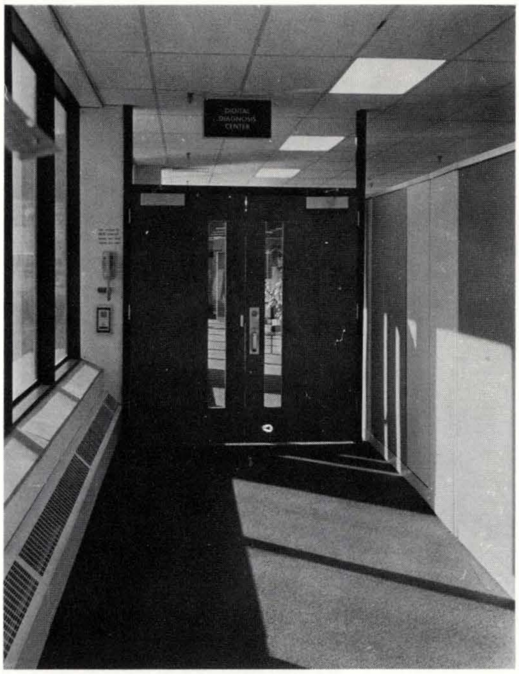
Located in the middle of this floor is a library of documentation that has a microfiche directory. All (or most) of DEC's documentation is here and available to the people when they need it.

Remote Diagnosis involves several groups within DEC. They include Field Engineering, The DDC, Remote Diagnosis Engineering, Maintainability Engineering, Diagnostic Engineering and Central Engineering. All these groups together make up the RD concept and bring it to fruition. What are they working on?

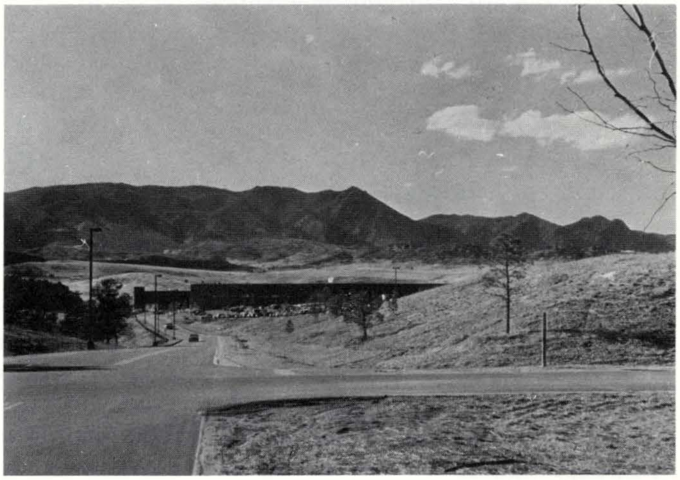
- Improve call handling
- Make the DDC look like a branch
- DDC movie
- DDC Notebook (hooray)
- Host software
- Diagnostic enhancements
- Review process for repeat problems, etc.
- U.S. Technical library
- System Products telephone support
- Area support
- DDC account Rep program

They are busy trying to improve what is already a good deal.

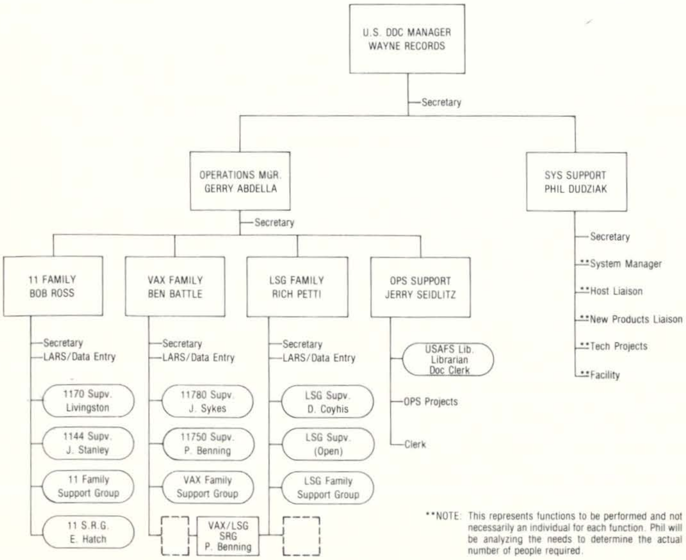
We ended what had been a most informative day. No longer is the DDC a phenomena in the mountains, they are made up of some real fine people out to do the best job they know how. We can help them do the job by doing a few simple things, and knowing a few things about them and our machines. Next issue I'll discuss how we can help them help us. I also had a chance to visit with the telephone software support people and I'll tell you what I saw there.

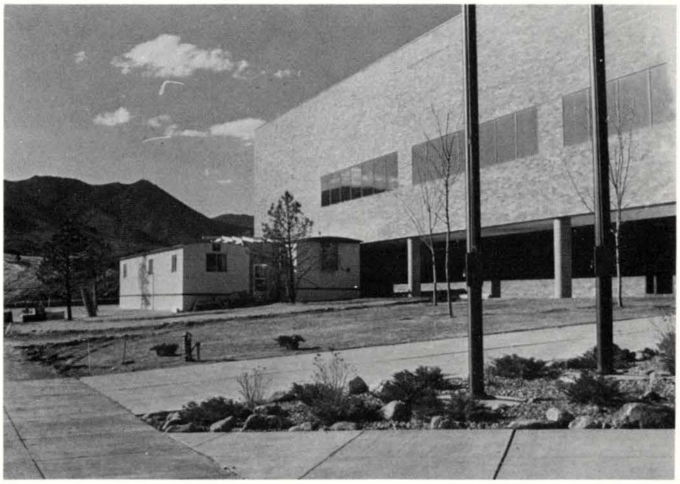


"THROUGH THESE DOORS . . . "



THE DDC WITH THE ROCKIES IN THE BACKGROUND





THESE TRAILERS ARE THE TEMPORARY QUARTERS FOR THE RSTS TELEPHONE RESPONSE GROUP

ODT.DOC

By Bob "MACRO MAN" Meyer

I'm Back! Sorry for missing last month's RSTS Professional; I would have been here much sooner, but I had to wait for a BP2 Task-build. . . (snicker!)

Anyway. . . We now present your prevously promised article on:

O D T ! !

"What's he getting us into this time", your're probably asking. Program bugs have been and will been in existance for about as long as programmers, as we all know. And a major aspect of programming is, of course, debugging. However, when you get yourself a bug in assembly language, ya' got yerself a BUGG!

We can all deal with messages like:

'Can't find file or account'

or:

'I/O channel not open'

and so on; but the real killers are the ones that say:

'Memory Manglement Violation'

or:

'Reserved instruction Trap'

or maybe:

'Odd Address Trap'

Now under normal circumstances, we should never have the honor of seeing one of these beauties on our screen (unless of course we're using BP2 . . .); but in Macro Land, these can become quite a common occurance, as some of you have already found out.

So long ago and far away, one (or more) of my Great Ancestors (Forefathers?) in Dec Land devised the infamous 'Octal Debugging Tool' (or was it 'Online Debugging Tecnique'. . .). Well, whatever it's really called, ODT has saved MY life more times than I can count in 16 bits or less, so I feel it may be of great assistance to some of you out there. (Probably only those of you that make mistakes.)

Due to time & space limitations, I'm only going to present some of ODT's basic commands, and a few simple examples. I strongly recommend reading the ODT Reference Manual as a supplement to this article. It comes shipped with the RSTS and RSX-11 manual sets.

Other than taking a 'PM Dump', this is about the only way of debugging pure assembly language programs. ODT can also be helpful if you're calling Macro subroutines from your favorite high-level language, as many run-time/object-time systems aren't expecting 'Odd address traps' from the code generated by the respective compiler.

Some of the things we can do with ODT are:

Examine/alter the contents of any memory location available to our program.

Examine/alter our own General Registers.

Set 'breakpoints' within the program. (much like the Basic 'Stop' statement)

Single-step through the program.

Remember, ODT's functionality goes a good bit beyond this list, so leaf through the manual when you feel your getting the hang of what we've covered here.

I usually find it easier to explain things with an example, so I'll wait while you key in the small program in Figure 1 . . .

Good! Your typing's getting a little better.

Here's how to assemble the Demo program:

MAC DEMO,DEMO = DEMO

and to link the task and ODT together:

TKB DEMO/DA,DEMO = DEMO

The '/DA' switch on the task image instructs TKB to link in the Debugging Aid. This is ODT.OBJ, and should exist in LB:SYSLIB.OLB. If it dosen't, the task builder will let you know.

Before we go on, you should PIP out the .LST & .MAP files you just created. These will be useful, and may even start to make sense after a while.

You can now

RUN DEMO

and on your screen you should see:

ODT:DEMO

_

That's ODT letting us know he's linked in (and has control of your task), as well as the name of the task being debugged, and finally, his prompt.

The only other thing you'll need to know is that ODT talks with your terminal in Binary or single character input mode; (also known as 'ODT submode' or 'DDT Submode' to us old DEC-10'ers). This means that he looks at each character as you type it in; the Return key is not needed to send commands to ODT, and in fact has the function of change the contents of memory (only in your own workspace, of course), so be careful not to hit it out of habit.

Also note that ODT, being designed for Macro minded folks like us, displays and accepts ALL numbers in OCTAL. This will take a bit of getting used to, but when dealing with a 16 bit machine, makes good sense. One of those newfangled Octal/Decimal & Hex calculators would be helpful, or TECO has the ability to convert between octal & decimal very nicely.

If you've typed in the Demo program, you can try some of the following commands.

First, let's look around in memory a bit. To examine a single memory location, we use the '/' character as follows:

_2000/ 000007

(remember, no return after the slash)

That command shows us the contents of memory location 2000 (octal) relative to the start of our task image. If you typed in the program as printed, you should see a seven, which corresponds to the following line in the source:

ONE:: .WORD 7.

To look at any location in our workspace, type that memory location followed by the slash. If you try to examine something out of the range of your task, the results will look like this:

_150000/ ?

indicating that the address is out of range. No harm is done.

. . . continued on page 77

ODT.DOC

By Bob "MACRO MAN" Meyer

Insert for page 74, RSTS PROFESSIONAL, June 1981.

Dear Readers:

We're pleased to present [HOT OFF THE PRESS], Figure 1.

```
.title      demo
one::       .word       7
two::       .word       6
demo::      mov         one,r0
            add         two,r0
            bpt
            .end        demo
```

FIGURE 1.

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.*

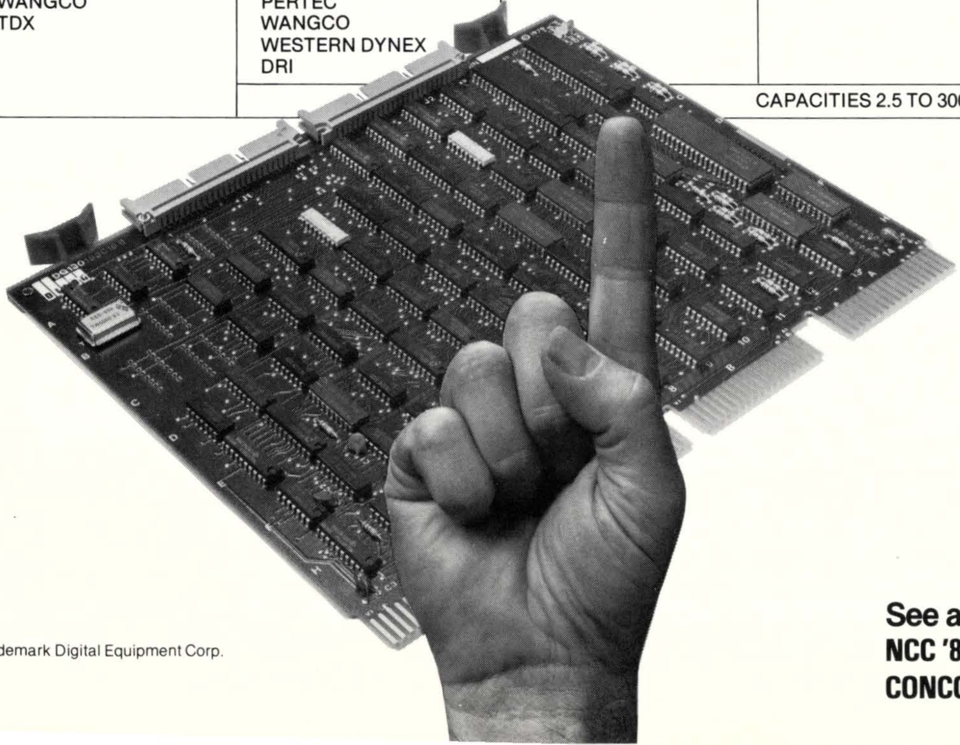
These high performance data storage interface products also feature • minimum bus/space requirements • up to 60% less power • 10 to 50% lower cost • automatic self-test... and numerous other features for easy system integration.

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"**
AMPEX CIPHER CONTROL DATA DIGI-DATA KENNEDY MICRODATA PERTEC TANDBERG DATA (IDT) WANGCO TDX	AMPEX CAELUS CENTURY DATA CONTROL DATA DEC DIABLO IOMEC MICRODATA PERTEC WANGCO WESTERN DYNEX DRI	AMPEX CONTROL DATA	AMPEX CENTURY DATA CONTROL DATA BALL COMPUTER MITSUBISHI	BASF CONTROL DATA FUJITSU KENNEDY MEMOREX PRIAM SHUGART SEAGATE QUANTUM IMI
	CAPACITIES 2.5 TO 300 MB			



DISTRIBUTED LOGIC CORP.
DILOG

See all the new DILOG products
NCC '81 BOOTH 5007 - 5009
CONCOURSE LEVEL • McCORMICK PLACE

*Trademark Digital Equipment Corp.

THE RSTS/E BENCHMARKS, Part I

. . . continued from page 12

you can reproduce on your own system. This will allow you to compare your configuration with those I use. This is one of the important techniques of benchmarking creating 'easily reproducable tests'. One of my favorites is the building of the RSTS/E cusps, a benchmark I've run on a variety of configurations.

Having written several times on performance evaluation and non-hardware methods to improve performance (file characteristics, caching, etc.) I will be taking some of those theories out of the classroom and into the laboratory to show the results of efforts I and others are promoting (such as well-structured disks).

I am also going to try and test as much new hardware as I can get access to — the 11/44 and 11/24 will be primary targets as will some non-digital hardware such as solid-state swapping disks and replacement terminal multiplexors. I hope the trip down benchmark lane will be enlightening for all of us. ♥

CASE –1578 C.M.

By Joel Schwartz, M.D.

One cold February night in 1979 I received a call from a man asking for help. His voice was filled with anxiety. "Doctor, I'm at a WITTS end." "I can't take it anymore." "Tell me about it." I responded in my best psychiatric tone. "First I couldn't get the bird in the cage, then I couldn't get the clam through the door, and now I'm at WITTS END." he replied. There was no question about it, this man needed help. "I'll come right over." I said, slamming down the phone. "Where are you going?" asked my wife. "I have an emergency, something about a bird and a clam. I'll be back within the hour." When I arrived, it was almost too late. The man was pushing up <CR>, down <CR>, North <CR>, South <CR>, East <CR>, West <CR>, in a perseverative fashion. "Can you help him?" begged his wife. "Please save our daddy." Sobbed the children. I told everybody to leave and then I locked the door. This case would take all the skill I possessed.

"How did this begin?" I said, sitting down beside the man. He looked up at me, his glassy eyes showing pain and frustration. "Somewhere nearby is a colossal cave" he muttered softly, "where others have found fortune and treasure in gold, but some who have entered have never been seen again." "Very interesting" I said, nodding my head and stroking my beard. He continued, "You start in a small brick building which is a well house for a large spring. You must try to find your way into an underground cave." "Hmm", I hmmd. "And where is this cave?" His face brightened up and his arms flew wildly into action as he began to push all sorts of keys. "Look" he said, pointing to the screen. Instructions and the beginning of this so called ADVENTURE game appeared before me. "Sit and play." he said emphatically. Not wanting to upset him more, I sat down and began.

YOU ARE IN FRONT OF A BUILDING, AN OLD WELL HOUSE, A STREAM FLOWS NORTH. "Punch in" he said, and I did. INSIDE THERE IS A LAMP, SOME KEYS, A BOTTLE, SOME WATER, AND SOME FOOD. "Take it all." he said. I'll just play along. Take all. ALL HAVE BEEN TAKEN. "Go out." I typed in "Out". "YOU ARE NOW OUTSIDE." "Go downstream." I was becoming annoyed, "Let me do this myself!" I said. Downstream. YOU ARE IN A VALLEY WITH STEEP ROCKS. Up <CR> . YOU ARE AT A STEEL GRATE. IT IS LOCKED. "Hey Doc, I feel much better now." Open grate. THE GRATE IS OPENED. "Doc, you can go now, I'm fine." "Thanks." In. YOU ARE IN A NARROW EASTWEST PASSAGE. "Hey Doc, I'm OK. Look, Doc." WEST <CR> . YOU ARE IN A CAVERN WITH ROCKS, ROCKY WALLS. ON THE FLOOR IS A CAGE. Take cage. TAKEN. "Doc, speak to me. Doc, speak to me." West. East. Up . . .

HELP! The doctor needs help with $\boxed{\text{DUNGEON}}$ **.**
Please write care of: RSTS Professional, Box 361, Ft. Washington, PA 19034.

insert photo X 50%

JOEL SCHWARTZ, M.D. ♥

ODT.DOC

. . . continued from page 74

Let's look at our program in memory. The task builder map tells us that the program occupies memory starting at location 2000. So one way to look at it would be to type:

```
__2000/000007
__2002/000006
```

and so on. An easier way to look at successive memory locations is to use the Line Feed key. Line Feed means 'close the current location, and open the next sequential location'. So we could also do it this way:

```
__2000/000007  < LF >
__2002/000006  < LF >
__2004/016700
```

and continue to the end of the program.

If we want to look at a large chunk of memory, we can use the 'L' command:

```
2000;2010L
002000 /000007 000006 016700 177770 066700
```

Although that's not a REAL large chunk, you get the general idea. (and I don't have to type all those numbers in...)

We can also use the slash to look at our registers. This is done by typing a '$' followed by the register we wish to examine (0-7):

```
__$0/000000
```

Lord only knows what'll be in them, until you put something there.

Since the BPT (breakpoint) instruction in our program will be intercepted by ODT, we can issue the 'Go' command, and watch our demo fly!

```
__G
BE:002014
```

'BE' means ODT picked up a breakpoint. He prints the location of the breakpoint, and returns control to us. At this point, we can check to see if our addition worked by looking at the contents of register 0:

```
__$0/000015
```

and as we all know, octal 15 is decimal 13, so it seems that the hardware really CAN add.

If you'd like to see that in slow motion, we can single-step through the program. First let's exit and start over:

```
__X
Ready (or '>' or '.' or whatever...)
RUN DEMO
ODT:DEMO
__
```

to single-step, simply type the letter 'S' for each instruction you wish to step through. ODT will allow your program to execute one complete instruction, then return control to you:

```
__S
8B:002010
```

'8B' is basically the 'eighth' breakpoint, which ODT uses for single step mode. The number printed is the address of the NEXT instruction to be executed.

Now let's ALTER the program a bit with ODT. Exit (X) and re-run the program. This time we'll modify one of the constants, proceed with program execution, and check the results:


```
ODT:DEMO
__2002/000006 3 < CR >
```

we got the address of the symbol 'TWO' from the task builder map (near the bottom, under 'GLOBAL SYMBOLS:') when ODT shows us the value contained there, we can modify it by typing the OCTAL number to put there, followed by CR . Now let's see the results:

```
__G            ;the GO command
BE:002014      ;the BREAKPOINT instruction
__$0/000012
```

examining register zero reveals octal 12, which is decimal 10.

In a real debugging situation, we may wish to alter instructions as well as data, without re-assembling & task building the program; this can be done as well. Now we'll change the ADD instruction into a subtract we know the start or TRANSFER address of the program from the task builder map, so let's first find the ADD instruction:

```
ODT:DEMO
__2004/016700  LF                 ;keep looking till we find it.
__2006/177770  LF
__2010/066700 166700  CR          ;that's the add.
```

Once the ADD instruction is found, we replace it with a SUB (166700 (I did mention having a Processor Handbook handy, didn't I?)) followed by a return. Now if we Proceed and look at the results, we should see:

```
__G
BE:002014
__$0/000001
```

The correct answer, 1.

Last of all, I'll give a simple example of how to set breakpoints in your program. Breakpoints give us the ability to monitor and/or control the execution of the program, by halting the task at a location which we specify. Let's first make a small modification to our demo program remove the BPT instruction, which will cause the program to loop indefinitely. Now rebuild the task as before, and fire it up:

RUN DEMO
ODT:DEMO

_

Looking at the task builder map (under 'MEMORY ALLOCATION SYNOPSIS:'), we can find that our program ('DEMO' if you used the '.TITLE' directive in the begining of the source, else '.MAIN.') begins at memory location 2000. Now refering to the Macro listing, a nice place to stop the task would be just before the instruction 'BR START'. This instruction (according to the listing) starts at 14 (octal) from the top of the task. Since 2000 + 14 = 2014, we could set the breakpoint as follows:

__2014;B

_

To start the program on it's way, we GO as before:

__G
0B:002014

'0B' means we've stopped at breakpoint 0, (we have from 0-7 available) and the address of the next instruction to be executed is printed. As before, we can check the results:

__$0/000015

_

Change values (or instructions):

__2000/000007 2

And PROCEED (always proceed after breakpoints, not GO):

__P
__0B:002014

And remember, the ONLY way out of ODT is X:

__X
Ready

That sums up what I wanted to cover here. Again, there's a LOT more to ODT than what you've seen here, so if you intend to use it to it's fullest extent, I suggest reading the manual.

GOOD LUCK

WITH YOUR

BUGGS!

♥

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

ADDRESS FOUR MEGABYTES WITH AN 11/34

May, 1981

ABLE 34 MAGNUM™ AND 44 MAGNUM™ BRING COMPATIBLE ALTERNATIVES TO DEC SYSTEMS MARKETPLACE

Irvine, California — ABLE Computer has entered the computer systems market with the introduction of 34 MAGNUM and 44 MAGNUM. Both are high performance and high reliability alternatives to existing Digital Equipment Corporation computer models while maintaining hardware/software compatibility. The 34 MAGNUM provides significantly enhanced PDP-11/34 class performance while the 44 MAGNUM performs as an economic alternative to the PDP-11/44. Both ABLE computers evidence design emphasis on system integrity and application with stringent up-time demand.

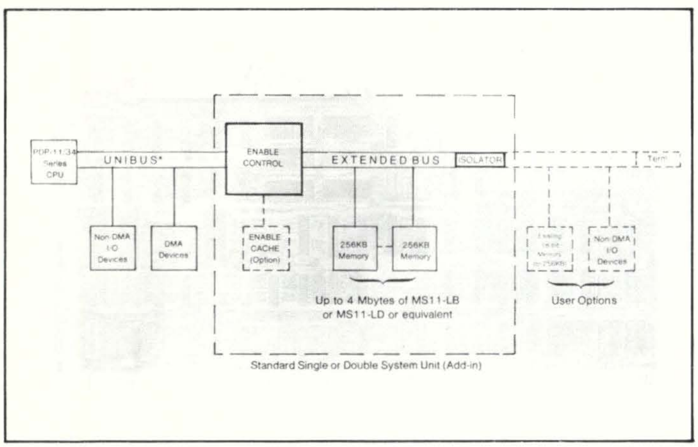
Standard system configuration for the ABLE alternatives consists of a CPU with floating point and memory management, serial console interface, extended memory addressing to 4M bytes, dual TU58 cartridge tape units including interface, programmable line time clock, bootstrap loader, 8K byte cache memory, microcoded ODT and maintenance console. The 34 MAGNUM comes with 256K bytes of memory, the 44 MAGNUM with 512K bytes as standard. Other ABLE Unibus compatible controllers are optional. Several standard MAGNUM items such as floating point, cache and the integral tape units are extra-cost options on certain DEC equivalents.

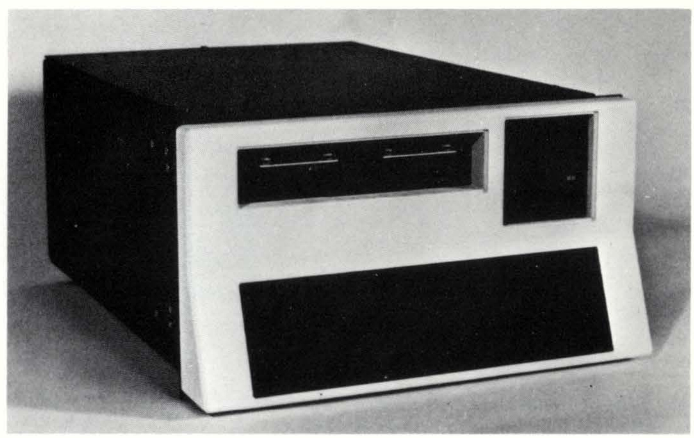
The two ABLE MAGNUMS are designed for high commercial reliability and dependability. They fill the quality gap between DEC models and those intended for MIL-SPEC operation. Mean-Time-to-Repair has been reduced to a minimum by features such as easy accessibility, a built-in front panel voltmeter and display of critical signals on the front panel. Other important features include a bus bandwidth greater than the alternatives, inexpensive loading of diagnostics or software updates and self-adaptive remote sensing of the power supply.

The ABLE 34 MAGNUM and 44 MAGNUM are supported by RSTS/E, RSX-11M and RT11 software operating systems as authorized by DEC. Both systems optimize embedded DEC CPU's in a stylish rack mountable chassis 24.25 inches deep, 17 inches wide and 10.5 inches high. U.S. list prices are $21,000 for the 34 MAGNUM and $27,000 for the 44 MAGNUM, according the user lower prices than those of the DEC equivalents. Delivery is less than 60 days ARO, another important advantage for the user.

ABLE is the world's largest independent supplier of enhancement interfaces for the VAX, PDP-11, System 20 and LSI-11 series of computers. The company manufactures an extensive line of communications, memory and general-purpose products of all which are hardware compatible with and software transparent to the host machine. They are supported and serviced worldwide with offices and plants located throughout the United States, as well as in England, Germany and Puerto Rico.

ENABLE/34 allows expansion of main memory up to four million bytes for existing PDP-11/34* RSTS/E systems.





FEATURES

- ENABLE/34 is a hardware/software solution for extending the address space of an existing PDP-11/34 series machine to 4 megabytes of main memory:
  - + Substantially increases system throughput
  - + Allows all currently executing programs to remain memory resident
  - + CPU speed becomes the only performance limiter.
- ENABLE/34 is implemented simply by a RSTS/E software enhancement supplied and supported by ABLE:
  - + Installation and removal of software are accomplished by an AUTO-PATCH procedure
  - + System can be tested, maintained, and operated with ENABLE/34 or as a standard PDP-11/34 system.
- ENABLE/34 is a most effective alternative:
  - + If your replacement system will not be delivered when you need it, or
  - + If your budget does not yet support the acquisition of a larger system, or
  - + If you simply wish to optimize your existing investment.
- Present memory can be used in concert with the added 22-bit memory or memories.

GENERAL DESCRIPTION

The ENABLE/34 consists of the ENABLE CONTROL, the BUS ISOLATOR, an optional cache memory, and unbundled software.

The ENABLE CONTROL is a single, modified hex-width board that installs in the second slot of any DD11-CK or DD11-DK system unit. It is connected to the processor simply by standard Unibus cable. All DMA devices are connected to the Unibus in front of ENABLE/34 to allow I/O mapping to occur.

The BUS ISOLATOR is a single, modified dual-width board. It is provided to allow the use of up to 256K bytes of installed 18-bit addressable memory.

The ENABLE CACHE is a quad-width board that installs in the first quad SPC slot of a system unit. The CACHE contains 8K bytes of high speed memory that can significantly increase the throughput of your system.

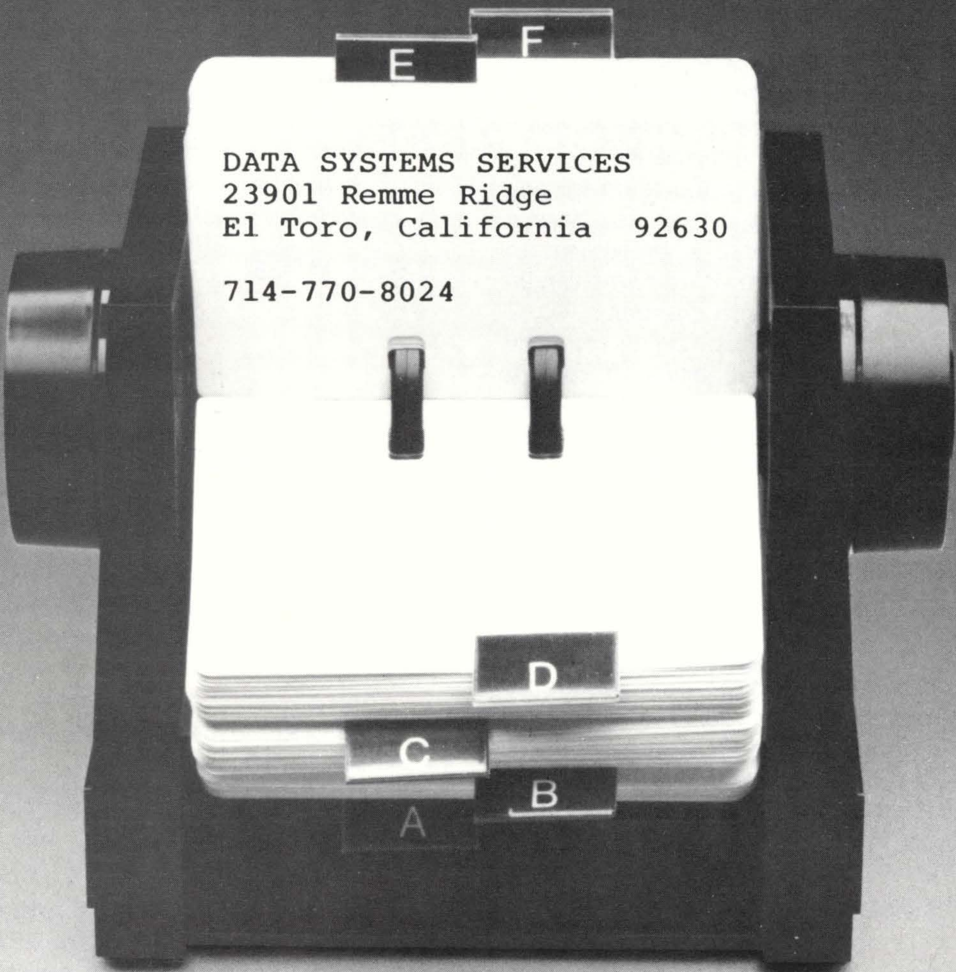
Note that the ENABLE/34 is not intended to replace newer or larger machines. It is designed to upgrade existing systems.

*Trademark of Digital Equipment Corporation

ENABLE and ENABLE/34 are trademarks of ABLE Computer.

We reserve the right to improve our products at any time.
10042X08-0581

THERE'S A WAY TO SAVE MONEY ON DEC TECHNOLOGY: DON'T CALL DEC.



They're the world's leading supplier of general purpose minicomputers. And they make equipment recognized for its high technology.

They also sell it at very high prices.

You can do better.

You can select from a wide range of disk drives—some with more capacity and more efficiency than anything DEC is offering. For instance, our DRM05 300 MB drive was available a year ahead of DEC. And our DRM07 600 MB drives are being installed now—again, ahead of any DEC announcement. And, you can choose from subsystems that include tape decks, add-in or add-on memory, multiplexers, disk drive controllers, and more.

All are media compatible. Software transparent. Easy to install. In fact, most of our peripherals come from DEC's own suppliers, but we offer them at a fraction of DEC's prices. For example, their RM05 drive (CDC's 9766) is listed at $34,000; while an identical drive, our DRM05, is available for $17,000. That's half DEC's price.

We're Data Systems Services. And we're the people who can do what we believe DEC can't: solve your DEC problems.

DSS: Whenever you need DEC, call us.

RSTS/E MONITOR INTERNALS Part 2

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

This is the second in a series of four articles describing the internal structures and operation of RSTS/E V7.0. The last article described the job control structures used for each active job. This article will describe the control structures necessary to control the memory used by a job.

My goal in writing these articles is to provide a complete description of the internals of the RSTS monitor in a format that is understandable to someone who is not a systems programmer. By using this information you can better understand what RSTS is attempting to do for you and work with it, instead of against it (alas, the normal case). The result should be an overall improvement in system performance and a more informed user who can intelligently interact with DEC on technical questions.

This may seem like a lofty goal, but judging from the responses I've received from the first article, it can be achieved. However, I need your help. Please let me know what you like and don't like about these articles. Your input will be useful in writing the next articles as well as in some others I have planned for the future.

2.0 MEMORY CONTROL

Memory is used for many different purposes. The monitor and cache buffering use a large chunk. Runtime systems and resident libraries take their toll. And, of course, let's not forget application programs. They are what we bought this machine for in the first place.

With all these demands on memory, RSTS has to make some pretty smart decisions to control this resource efficiently. The primary way it provides this control is through memory control sub-blocks.

2.1 MCB — MEMORY CONTROL SUB-BLOCKS

The available memory on a system is typically broken into many pieces, each being used for a different purpose. Memory control sub-blocks (MCB) are used to keep track of each of these pieces of memory.

An MCB is not a structure by itself. It is a part of other structures that describe functions which use memory, such as the job data block (JDB). The structures which contain memory control sub-blocks are: job data blocks (JDB), runtime system descriptor blocks (RTS), library descriptor blocks (LIB), the RSTS monitor, XBUF, locked out memory and non-existent memory.

The memory control sub-blocks are always in one of three states: (1) linked into a list of current memory users, (2) linked into a list of structures that desire memory, (3) not in memory and not desiring to come into memory.

The memory control sub-block is defined as follows:

Offset	Symbol	Description
0	M.PPRV	Pointer to the previous memory control list at offset M.PNXT.
2	M.PNXT	Pointer to the next memory control sub-block in this memory control list.
4	M.TSIZ	Number of K-words mapped by this MCB. This size includes the amount of memory used by the structure which contains this MCB plus any available memory that follows it.
6	M.SIZE	This byte contains the number of K-words in use by the structure containing this MCB. This size subtracted from M.TSIZ yields the number of K-words following this structure which are available for other uses.
7	M.CTRL	This byte contains the memory status information about the portion of memory mapped by this MCB (see 2.1.1).
8	M.PHYA	This word contains the physical starting address of the piece of memory mapped by this MCB divided by 64.

2.1.1 M.CTRL — Memory Status Information

The memory status information bits contained in M.CTRL are defined as follows:

Bit	Symbol	Description
8	REQ	Residency is requested but the MCB is not linked into RESLST because it is currently being swapped out.
9	OUT	Entry should be removed from memory or is currently removed from memory.
10	IN	Entry should be brought into memory.
11-13		Unused
14	SWP	A swap is desired. OUT and IN determine the swap direction.
15	LCK	The memory segment described by M.SIZE is not available for allocation for other uses or swapping out.

Some typical combinations of bits in M.CTRL are:

LCK, SWP, OUT	The entry is resident but should be swapped out.
LCK, OUT	The entry is in the process of swapping out.
LCK, SWP, IN	The entry has been allocated memory and should be swapped in now.

LCK, IN	The entry is in the process of swapping in.
LCK	The entry is not available for swapping out.
OUT	The entry is not currently in memory and does not desire to be made resident.

2.2 MEMLST — RESIDENT MEMORY LIST

All of the memory on a system is defined by the resident memory list, MEMLST. As memory is divided among several different usages the memory control sub-blocks for each usage are linked into MEMLST in ascending order. Thus, by following the links between each MCB we have seen all available memory in the order it is allocated.

The memory control list is based at the location MEMLST. This location is the address of the first entry in the memory control list, rather than a pointer to the first entry as in most other linked lists. The first entry describes the memory used by the monitor and any free memory following it.

The memory control list always contains at least three entries. These are the root MCB, the system default runtime system, and the tail MCB. The root is actually the monitor MCB. The tail terminates the list and shows the highest memory location addressable on the system.

2.2.1 Root Memory Control Sub-Block

The first entry in the resident memory list is the root MCB. This entry starts at location MEMLST and describes the memory used by the monitor and any free memory following it.

The format of the root MCB is as follows:

Offset	Symbol	Description
0	M.PPRV	The link to the previous entry is 0 since this is the first entry in MEMLST.
2	M.PNXT	Pointer to the next entry in MEMLST.
4	M.TSIZ	The total of the monitor's size plus any free memory following the monitor.
6	M.SIZE	This byte contains the size of the monitor in K-words.
7	M.CTRL	The status bit LCK is set to show that the monitor's memory is not available for other uses.
8	M.PHYA	The starting physical address is 0 since the monitor always starts at location 0.

2.2.2 Tail Memory Control Sub-Block

The tail MCB is the last entry in MEMLST. It terminates the list and defines the highest memory address available on the system. The format of the tail MCB is:

Offset	Symbol	Description
0	M.PPRV	The backwards link points to the previous entry in MEMLST at its M.PNXT entry.

2	M.PNXT	The forwards link contains a 0 to show that it is the end of the list.
4	M.TSIZ	The total size for this entry is set to a value of 1 but is not used by the monitor.
6	M.SIZE	The size of this entry is set to a value of 1 but is not used by the monitor since the LCK bit is set in M.CTRL.
7	M.CTRL	The status bit LCK is set to show that this memory segment is not available for other uses.
8	M.PHYA	The starting physical address of this memory segment corresponds to the address of the first non-existent memory on the system (divided by 64). Therefore, this value is the total system memory size (in K-bytes, divided by 64).

2.3 RESLST — DESIRED RESIDENCY LIST

When an entry is not currently resident in memory and wants to become resident, it is linked onto the end of the desired residency list, RESLST. The memory manager uses this list in a first-in/first-out basis to keep track of requests for memory beyond what is available in MEMLST.

When an entry is added to RESLST, the memory manager is scheduled to fulfill the desired residency request. If there is not enough memory available to honor the residency request, entries that are currently resident in memory are reviewed. Those which are eligible for swapping are then scheduled to be swapped out. Once sufficient memory is available, the requestor is made resident and the MCB is removed from RESLST and added to MEMLST.

The first entry in RESLST is pointed to by the location RESLST. If no entries are in the desired residency list the location RESLST will contain a 0.

Memory control sub-blocks in RESLST have the following format:

Offset	Symbol	Description
0	M.PPRV	Pointer to the next entry in RESLST or 0 if this is the last entry.
2-5		Unused
6	M.SIZE	The size of this entry in K-words.
7	M.CTRL	The control bits LCK, SWP and IN are set to show that swap-in is desired.
8	M.PHYA	This word contains either a 0 to show that no specific memory address is required or it contains the desired memory address divided by 64.

2.4 RTS — RUNTIME SYSTEM DESCRIPTOR BLOCK

Every runtime system that is currently installed in the system has an RTS descriptor block associated with it. This structure contains all the information about the runtime system, including its name, memory control information, disk address and characteristics.

The RTS block has the following format:

```
               ---------------------------------------
               |     Pointer to the next RTS block    | 0  R.LINK
               |--------------------------------------|
               |                                      | 2  R.NAME
               |     Runtime system name (RAD50)      |
               |                                      | 4
               |--------------------------------------|
               |     Default extension (RAD50)        | 6  R.DEXT
               |--------------------------------------|
               |     Memory control sub-block         | 8  R.MCTL
               |                                      |
               |                                      | 10
               |                                      |
               |                                      | 12
               |                   -------------------|
               |                   |   Size of RTS    | 14 R.KSIZ
               |                   -------------------|
               |                                      | 16
               |--------------------------------------|
               |  Starting block number of RTS (LSB)  | 18 R.DATA
               |--------------------------------------|
               |  FIP unit number  |  MSB of R.DATA   | 20
               |--------------------------------------|
               |  Block number of RTS directory entry | 22 R.FILE
               |--------------------------------------|
               | Offset in block/2 |  MSB of R.FILE   | 24
               |-------------------+------------------|
               |  Residency count  |   User count     | 26 R.CNT
               |-------------------+------------------|
R.MSIZ 29      | Minimum job size  | Maximum job size | 28 R.SIZE
               |-------------------+------------------|
               |  Characteristics  |   EMT prefix     | 30 R.FLAG
               ---------------------------------------
```

Offset	Symbol	Description
0	R.LINK	This word contains the address of the next runtime system descriptor block. If this entry is the last RTS block in the list it will contain a 0.
2	R.NAME	These two words contain the runtime system name in RAD50.
4	R.DEXT	This word contains the default extension (in RAD50) for executable files used by this RTS. If a RUN command is issued without specifying an extension for the file to be executed this value will be used for the extension on files executed under this runtime system.
6	R.MCTL	These five words are the memory control sub-block for the runtime system. See section 2.1 for more information.
14	R.KSIZ	This word (within the memory control sub-block) contains the size of the runtime system in K-words.
18	R.DATA	These three bytes contain the FIP block number of the first block of the runtime system image. When a runtime system is loaded into memory it is accessed on disk by this block number. Byte 20 is the most significant byte of the block number.
21		This byte is the FIP unit number for the disk containing the runtime system. It is used when loading the runtime system image and when closing the runtime system file when the RTS is removed.
22	R.FILE	These three bytes contain the FIP block number of the block that contains the UFD name entry for this runtime system. It is used to close the RTS file when the RTS is removed.

25		This byte is the offset to the name entry of the RTS file within the directory block specified by R.FILE divided by 2.
26	R.CNT	This byte contains a count of the number of jobs currently using this runtime system.
27		This byte contains a count of the number of jobs using this runtime system which are currently resident in memory. If the residency count is 0 the runtime system is eligible for "swapping out". If a runtime system is loaded with the /STAY switch the high bit of this byte is set, ensuring that the residency count will never be 0 and the runtime system will always remain in memory.
28	R.SIZE	This byte contains the maximum size for a job image using this runtime system.
29	R.MSIZ	This byte contains the minimum size for a job image using this runtime system.
30	R.FLAG	If the PF.EMT bit (see 2.4.1) is set in the high byte of this word this low byte contains the special EMT prefix value. See the RSTS Programming Manual for more information.
31		This byte contains a set of bits that describe the characteristics of the runtime system (see 2.4.1).

2.4.1 R.FLAG — Runtime System Characteristics Flags

The runtime system characteristics flags contained in the word R.FLAG are defined as follows:

Bit	Symbol	Description
0-7		Special EMT prefix (see PF.EMT below)
8	PF.KBM	The runtime system can act as a keyboard monitor.
9	PF.1US	The runtime system is single user, non-sharable.
10	PF.RW	Map runtime system read/write.
11	PF.NER	Do not log errors occurring under this runtime system.
12	PF.REM	Remove the runtime system image from memory when R.CNT becomes 0.
13	PF.CSZ	Compute initial job size.
14	PF.SLA	Load at the address specified by M.PHYA of the MCB.
15	PF.EMT	Low byte of R.FLAG is special EMT prefix code.

2.4.2 NULRTS — Disappearing RSX RTS

One of the options at sysgen time is to embed support for the RSX emulator into the monitor. When this is done the RSX runtime system disappears after initiating program execution.

One of the requirements of every job on the system is that it have a runtime system associated with it at all times. RSTS meets this requirement when using the disappearing RSX runtime system through use of the null RTS descriptor block, NULRTS.

The null RTS descriptor block is not linked into RTSLST, the list of RTS blocks. It is only used to give the job descriptor block (JDB) an RTS block to point to.

The format of the null RTS block is the same as a normal RTS block. All the fields contain 0 except the following:

Symbol	Value
R.NAME	"...RSX" in RAD50
M.CTRL	LCK bit set
R.SIZE	System swap maximum
R.MSIZ	1

2.4.3 RTSLST — Runtime System List

The RTS blocks are linked together in a list pointed to by the location RTSLST. The first entry is always the system default runtime system. It links to the other RTS blocks in the order displayed by SYSTAT.

2.5 LIB — RESIDENT LIBRARY DESCRIPTOR BLOCK

Each resident library installed in the system is described by a resident library descriptor block (LIB). A library descriptor block is very much like an RTS block in that it contains information about the resident library's name, memory control information, disk address and characteristics.

Refer to the description of the .PLAS call in the RSTS System Directives Manual for a complete description of resident library support and the use of memory windows.

The format of a resident library descriptor block is as follows:

```
            ---------------------------------------
            |     Pointer to the next LIB block   | 0  R.LINK
            |-------------------------------------|
            |                                     | 2  R.NAME
            |     Resident libary name (RAD50)    |
            |                                     | 4
            |-------------------------------------|
            |           PPN of library            | 6  L.PPN
            |-------------------------------------|
            |       Memory control sub-block      | 8  R.MCTL
            |                                     |
            |                                     | 10
            |                                     |
            |                                     | 12
            |                   ------------------|
            |                  |   Library size   | 14 R.KSIZ
            |                   ------------------|
            |                                     | 16
            |-------------------------------------|
            |  Starting block # of library (LSB)  | 18 R.DATA
            |-------------------------------------|
            |  FIP unit number  |  MSB of R.DATA  | 20
            |-------------------------------------|
            |  Block # of library directory entry | 22 R.FILE
            |-------------------------------------|
            | Offset in block/2 |  MSB of R.FILE  | 24
            |-------------------+-----------------|
            |  Residency count  |   User count    | 26 R.CNT
            |-------------------+-----------------|
L.PROT 29   |  Protection code  |     Status      | 28 L.STAT
            |-------------------------------------|
            |     Library characteristics         | 30 R.FLAG
            ---------------------------------------
```

Offset	Symbol	Description
0	R.LINK	This word contains the address of the next resident library descriptor block. If this entry is the last LIB block in the list it will contain a 0.
2	R.NAME	These two words contain the resident library name in RAD50.
4	L.PPN	This word contains the account number (PPN) of the resident library file. The project number is in the high byte. The programmer number is in the low byte.
6	R.MCTL	These five words are the memory control sub-block for the resident library (see 2.1).
14	R.KSIZ	This word (within the memory control sub-block) contains the size of the resident library in K-words.
18	R.DATA	These three bytes contain the FIP block number of the first block of the resident library image. When a resident library is loaded into memory it is accessed on disk by this block number. Byte 20 is the most significant byte of the block number.
21		This byte is the FIP unit number for the disk containing the resident library. It is used when loading the resident library image and when closing the resident library file when the resident library is removed.
22	R.FILE	These three bytes contain the FIP block number of the block that contains the UFD name entry for this resident library. It is used to close the resident library file when it is removed.
25		This byte is the offset to the name entry of the RTS file within the directory block specified by R.FILE divided by 2.
26	R.CNT	This byte contains a count of the number of jobs currently attached to this resident library.
27		This byte contains a count of the number of jobs using this resident library which are currently resident in memory. If the residency count is 0 the resident library is eligible for "swapping out". If a resident library is loaded with the /STAY switch, the high bit of this byte is set, ensuring that the residency count will never be 0 and the resident library will always remain in memory.
28	L.STAT	This byte is used to differentiate between an RTS block and a LIB block. If bit 7 (symbolically, LS.LIB) is set this is a LIB block, otherwise it is an RTS block. LS.LIB is the only bit currently defined for L.STAT.
29	L.PROT	This byte is the library protection code. The protection code is used to control access to the memory space of a res-

		ident library. It is identical in usage to the file protection codes except that bits 6 and 7 have no meaning.
30	R.FLAG	This word contains a set of bits that describe the characteristics of the resident library (see 2.5.1).

2.5.1 R.FLAG — Resident Library Characteristics

Bit	Symbol	Description
0-8		Unused
9	PF.1US	The resident library is a single user, non-sharable.
10	PF.RW	The resident library may be mapped read/write if requested by a privileged user.
11		Unused
12	PF.REM	Remove from memory when R.CNT becomes 0.
13		Unused
14	PF.SLA	Load at specific address. This bit is always set for a resident library since libraries must be loaded at a specific address.
15		Unused

2.6 WDB — WINDOW DESCRIPTOR BLOCK

A job's memory space consists of the user low segment, the runtime system and up to five resident libraries mapped by up to seven windows. If a job is not using any resident libraries the job's memory requirements are totally described by the memory control sub-blocks in its JDB and RTS blocks.

However, when a job attaches to one or more resident libraries an additional control structure is needed to keep track of the extra memory windows. This structure is the window descriptor block (WDB).

The WDB consists of up to three small buffers of information and describes up to seven memory windows and five resident libraries. Up to two windows and five resident libraries can be described with a single small buffer. An additional small buffer is required for each three additional windows.

The format of the first window descriptor block is as follows:

```
 -------------------------------------
|        Pointer to next WDB          | 0  W.LINK
|-------------------------------------|
|    Pointer to LIB descriptor #1     | 2  W.ALIB
|-------------------------------------|
|    Pointer to LIB descriptor #2     | 4
|-------------------------------------|
|    Pointer to LIB descriptor #3     | 6
|-------------------------------------|
|    Pointer to LIB descriptor #4     | 8
|-----------------------------/-------|
|    Pointer to LIB descriptor #5     | 10
|-------------------------------------|
|                                     | 12 W.WIN1
|                                     |
|                                     | 14
|                                     |
|          Address window #1          | 16
|                                     |
|                                     | 18
|                                     |
|                                     | 20
|-------------------------------------|
|                                     | 22 W.WIN2
|                                     |
|                                     | 24
|                                     |
|          Address window #2          | 26
|                                     |
|                                     | 28
|                                     |
|                                     | 30
 -------------------------------------
```

Offset	Symbol	Description
0	W.LINK	If this job has more than two windows in use this word contains the address of the second window descriptor block at offset W.ALIB. Otherwise this word contains a 0.
2	W.ALIB	These five words contain pointers to the library descriptor blocks for up to five libraries. If less than five libraries are in use the unused entries will be 0.
12	W.WIN1	These five words are the first address window (see 2.6.1)
22	W.WIN2	These five words are the second address window (see 2.6.1)

2.6.1 W.WIN? — Address Windows

If a window is not in use its first word will be 0. An address window has the following format:

```
           -------------------------------------
W$NSTS 1  |  Window status    |      Base APR   | 0  W$NAPR
          |-------------------------------------|
          |     Window size divided by 64       | 2  W$NSIZ
          |-------------------------------------|
          | Pointer to library descriptor pointer| 4  W$NLIB
          |-------------------------------------|
          |  Offset into library divided by 64  | 6  W$NOFF
          |-------------------------------------|
          |         Map length in bytes         | 8  W$NBYT
           -------------------------------------
```

Offset	Symbol	Description
0	W$NAPR	This byte contains the APR number for the base of the window.
1	W$NSTS	This byte contains a bit pattern describing the status of the window (see 2.6.1.1).
2	W$NSIZ	This word contains the desired window size in bytes divided by 64.
4	W$NLIB	This word is the address of the pointer (in W.ALIB) to the library descriptor for the library associated with this window.
6	W$NOFF	This word is the map offset into the library divided by 64.
8	W$NBYT	This word is the map length in bytes.

2.6.1.1 W$NSTS — Window Status

The current status of each window is described by the bits contained in the byte W$NSTS in each address window. These bits have the following meaning:

Bit	Symbol	Description
8	WS$WRT	Write access is desired.
9-14		Unused
15	WS$MAP	The window is currently mapped.

2.6.2 Extended Window Descriptor Blocks

If more than two windows are defined for a job an additional window descriptor block is allocated and linked to be the primary window descriptor block. If more than five windows are defined a third WDB is allocated and linked to from the second one.

The format of the second WDB is as follows:

```
 --------------------------------------
|       Pointer to third WDB           | 0   W.LINK
|--------------------------------------|
|                                      | 2
|                                      |
|                                      | 4
|                                      |
|        Address window #3             | 6
|                                      |
|                                      | 8
|                                      |
|                                      | 10
|--------------------------------------|
|                                      | 12
|                                      |
|                                      | 14
|                                      |
|        Address window #4             | 16
|                                      |
|                                      | 18
|                                      |
|                                      | 20
|--------------------------------------|
|                                      | 22
|                                      |
|                                      | 24
|                                      |
|        Address window #5             | 26
|                                      |
|                                      | 28
|                                      |
|                                      | 30
 --------------------------------------
```

The format of the third WDB is as follows:

```
 --------------------------------------
|             Always 0                 | 0   W.LINK
|--------------------------------------|
|                                      | 2
|                                      |
|                                      | 4
|                                      |
|        Address window #6             | 6
|                                      |
|                                      | 8
|                                      |
|                                      | 10
|--------------------------------------|
|                                      | 12
|                                      |
|                                      | 14
|                                      |
|        Address window #7             | 16
|                                      |
|                                      | 18
|                                      |
|                                      | 20
|--------------------------------------|
|                                      | 22
|                                      |
|                                      | 24
|                                      |
|             Unused                   | 26
|                                      |
|                                      | 28
|                                      |
|                                      | 30
 --------------------------------------
```

The format of each address window is identical to that of the first WDB. See 2.6.1.1 for more information.

In the next issue: FILE AND DEVICE CONTROL

Notes from a Ride on the AMTRAIN(ing)

By Peg Leiby and Sue Smith

Ambase is a data base management system offered by AMCOR, Inc. of Louisville, KY. We recently attended the training at AMCOR Headquarters.

One of the first things that become obvious about AMSCHOOL is that EVERYTHING seems to have a prefix of "AM". Please bear with the "AMing", since we are now AMBASE graduates.

AMDAY 1:

The training began early on Monday morning, sometime around 8:00 A.M. We were surprised since other classes we've attended tended to have shorter days. Our instructor, Cliff Jeffries, introduced the three other students in the class, two from LIFEBOAT Associates, located in New York, and one an AMCOR employee trainee. AMCOR insists that the class size be small. In fact, to keep the classes small, they were running a second section of training at the same time with another teacher. We were each given a training manual with a 38 point outline breaking down the week's course into the lectures and labs to be covered. Along with the manual, all handouts for the day were delivered separately in a packet.

We began with an introduction to AMBASE: an overview of the product, review of the file structures, methods of fielding, sorting, etc. To use AMBASE, application programs must access what is known as the schemata, a file in which all fields, their names, verification checks, aid messages, packing techniques, and sort definitions are stored. This is the "black box" of AMBASE which accesses the key and data files, which together, are known as "data-sets". Each data-set can be formatted up to three different ways, which enables more than one kind of record of different sizes in a data file. This saves channels, among other things. Each data-set may have up to 42 different keys, although the fewer the keys, the more efficient the processing.

An overview of the features offered seemed a bit overwhelming at first. What do two BASIC+ programmers know about terms like Screen Generator, Report Generator, Query Language, Code Generation... ah yes, no wonder the days are so long—there's a lot to learn, and we did as the week progressed.

After the overview, we discussed eleven of the AMBASE utilities: how to define a dataset, display its contents, and modify it with a general file maintenance, etc., all of which are easily mastered.

We had three lab sessions on AMDAY one, two students to a terminal. The labs that day were spent defining datasets, which were provided in the training manual on nice lay-out sheets describing the field and sort structures. By the end of the lab sessions, we had used AMBASE utilities to define datasets, and then to add and display records. During dataset definition an ascii definition file is created which when 'teco-ed' for desired changes and passed thru the definition utility will re-define the data-set in a matter of minutes. Two utilities we'd like to mention are AMPOR and AMSTAT. AMPOR creates a "portrait" which displays file definitions, ambase control information, sort keys, etc. A super documentation utility—any time a data-set is redefined AMPOR offers immediate updates for documentation. AMSTAT is the statistics utility which returns the number of insertions, deletions and changes that have occured to the key files, shows amount of used and unused space of data, and gives a degradation analysis which is used to determine key optimization. AMSTAT statistics are easily reset when desired.

AMDAY 2:

We learned how to interface AMBASE with BASIC+2 code via the AMBASE standard calls which access the schemata. The calls reviewed included AMOPEN—opening the dataset files, AMREAD and AMWRIT—reading and writing to the files, AMPACK and AMUNPK—getting and replacing fields, and AMASK—the method used for all operator interaction and prompting. The nice thing about using the AMASK call is making use of the data dictionary (schemata) for valid answers, CVT$$ conversion, default values, aid messages, and oh yes, the BACK feature, which sends the program back to the last AMASK! By using these AMCALLS, you accomplish one of the things that AMCOR can boast about its package; data independence, since the code written will then be flexible enough not to change when changes have been made to the file structures. By the way, AMBASE is now supplied as a MACRO-based resident library.

There were three labs again. Because of time restrictions, AMCOR supplies shell programs for the trainees to use and modify. By the end of the lab sessions we had successfully passed through the datasets which were defined the day before, using partial keys in BASIC 2. We unpacked and displayed fields using the AMCALLS, compiled and TKB'd the code and ran the programs. The lab assignment was to give everyone in the dataset an immediate 10% pay raise. We had placed our names in the dataset the previous day, with huge salaries and naturally didn't mind this task. We were both a bit humbled, however, when we noticed what appeared to be a loop—continuous 10% increases. Our attitude became one of—well, so what, you know computers; how could we be upset about increases that kept us so far above the inflation rate. "Wait a minute,—why is my zip code getting larger, and larger, and larger . . ." As it turned out we not only were updating the zip codes by 10% to infinity, we weren't even getting a raise! Cliff didn't seem to be mildly amused, he was down right laughing. He quickly showed us our error, which we did fix and re-run. Quite a long day, this one was; no swimming at the motel pool tonight, but a day well spent.

AMDAY 3:

Today we spent time discussing key considerations in great depth, basically to determine the pros and cons of having duplicate or blank keys defined for the dataset. More of the AMBASE calls were reviewed as well.

The bulk of AMDAY three was spent on the Screen Generator, a method of quickly producing file maintenance or inquiry programs by answering a dialogue. The AMBASE screen generator creates code which is appended and compiled to the supplied utility shells. An ascii definition file is created for easy editing and re-run to make changes at a later time. We both found the utility dialogue somewhat confusing; however after practicing, most of the confusion was overcome. Calculations can be defined to occur upon operator input, and data can be verified against other data-sets in the schemata, (although not updated, one of the shortcomings of the ambase product as it is today.) The cursor/echo control can be defined for most terminals through their standard definitions. Ambase really generates a lot of code!

Again, three labs, with lots of practice using the screen generator. We defined and laid out the screen with little problem and then added, deleted and updated fields easily. Lots of strange looking zip codes... Our program used the calculations entered through the dialogue as various fields were updated.

AMDAY 4:

A thorough explanation of data-set relationships, called logical views, a way of re-defining your data into specific and limited groups of fields, perhaps from several related files. This is accomplished through a utility known as the subschemata librarian. Much time was spent discussing "parent" to "child" relationships as either one-to-one, or one-to-many types.

Much time was also spent on AMDAY 4 learning to utilize the Report Generator facility. This utility is a powerful programmer tool to quickly create reporting programs. The dialogue is somewhat complex definitely not meant to be run by the everyday user of the system. By answering the utility dialogue, the entire report lay-out, headings, details, breaks, paging, totals, etc can all be defined with an opportunity to modify the print-using formats if desired. The query language, which we were to learn on AMDAY 5, can also be interfaced with the report generator. The technical aspects of the programs we reviewed extensively in order to fully understand the typical modifications users make to the actual code.

During the three labs, we produced reports using the pre-defined datasets from prior days, as well as defining a subschemata, which was used in the report.

AMDAY 5:

Query Language—Among other things, this facility allows for special inquiries by non-technical people. We learned that the query language utility can be used in three different ways: directly, interfaced to any user program or interfaced to the report generator. At run time, many different field parameters can be selected by employing the basic-plus verbs, such as "if", "or", " ", "+", "unless", etc. AMBASE is somewhat deficient at this time in that only one dataset can be queried, however this is very high on the Amcor amwish list for product improvement. A somewhat complicated method to get around some of the query language limitations was discussed.

The rest of the lecture was spent on installation, design considerations and various tips.

AMHOME:

Overall, we found the training to be conducted in a very professional manner. Not only did we learn the product, but our general knowledge was also enhanced. After all, our experience in BASIC+2 was somewhat limited, neither of us ever having used the BASIC+2 calls before.

Much exhausted, but intellectually stimulated, we were both eager to return home to use what we learned. Cliff had stressed in the beginning that the best way to start was to go easy at first. No showing off with a big project. Waiting, however, was a complete conversion of 7 DMS-500 mailing list files to AMBASE, with 4-up labels needed almost immediately. After defining the file and sort structures, AMLOAD, the utility to transfer foreign data to AMBASE was used. Unfortunately, AMLOAD was not reviewed at the class, and the documentation, at least for us, wasn't the easiest to understand. It is basically geared toward magtape transfer, and this was all from disk files. The support from AMCOR was excellent, we couldn't have asked for more. The best time to do the data transfers were during the night. The report generator isn't able to produce something like 4-up labels, therefore a suggestion from AMCOR was to copy over and modify one of the utilities. (The source code is included with the AMBASE package for many of the utilities.) AMDIS, the utility that basically dumps out the records in a nice format was used. It turned into a bit of forced training since much of it didn't apply to the code needed but before long a label program was produced. I found, though, that one of the DMS-500 files had had some inconsistent file lay-outs with part of the city tagging along the prior address lines. Time to learn more, with the deadline approaching. With a quick AMCHG, a call to update non-key fields, which passed through the entire file (20,000) records, all was in order, verified by using the AMDIS and AMGENT utilities.

Once the 100,000 or so labels were produced and out of the way, screen generators, report generators and query language attempts were made with ease. All in all, a great learning experience and we're sure a valuable product.

(HOW'S YOUR SPANISH)
PEG Y SUE

INSIDE AMCOR

1900 PLANTSIDE DRIVE
LOUISVILLE, KY

CLASSIFIED

Send Classified Ads to: RSTS Classified, P.O. Box 361, Fort Washington, PA 19034. $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.]

HIGH SPEED Data Acquisition under RSTS. R.B. Lake, 2540 Derbyshire, Cleveland, OH 44106. (216) 932-5434.

SECURE YOUR RSTS/E System. Write: Security, 471 Townline Rd., Hauppauge, NY, 11788.

DEC COMPUTERS AND components bought and sold, 617—774-4942.

80MB CDC DISKS 8000 w/CTLR 12000. 9T tape 4000 w/CTLR 7500. Jon Wells 714—641-9141.

THIRD PARTY computer maintenance. General Electric Co., (913) 967-6228.

WOULD LIKE to exchange information on law enforcement systems. Livonia Police Dept., 15050 Farmington, Livonia, Michigan 48154.

TIMESHARING IN Claremont and Pomona. PDP11/34 RSTS. Reasonable. Call (714) 626-4611.

DEC PERIPHERALS WANTED. Douglas Mayers, Box 1067, Gambier, Ohio 43022.

SOFTWARE SUPPORT from Free Lance Logic, Santa Ana, CA (714) 979-7860.

DIGITAL BUSINESS Computers, DEC hardware/software, Kitchener, Ont., Canada. 1-519-893-4200.

PRO-Where Professionals Meet With Professionals. Pacific Recruiting Organization, Annie Gibbons, (408) 730-5155.

INTERACTIVE COMPUTER Services, RSTS/E Systems and Programming Consultant. Charlotte, NC, 704—523-0753.

WE BUY Used Terminals. Warner Associates, (212) 684-6955. DEC Hazeltine.

LOOKING FOR other 11/44's in the South Texas area and also for other supermarket companies under RSTS. Buz Waitz, Carl's Grocery Co., Inc., 1123-A E. 9th, Mission, Texas 78572. (512) 585-4578.

GREETINGS from First RSTS Product Manager, Tom Barnett. Digital Merrimack, NH 03054.

ANALYSIS and DESIGN, custom business software, facilities management, timesharing services, media conversions. (714) 293-3200.

CLAIMS PROCESSING - the specialists in hospital outpatient systems. Phone (201) 731-9252.

RSTS/E BASIC+, DIBOL Programmers call or send resume to Scientific Placement, Inc., P.O. Box 19949, Houston, TX 77024; (713) 496-1600.

MR. BOOKER'S Computer Concepts is Addison County Vocational's most unusual course.

DEC SYSTEMS & Components, C.D. Smith & Associates, Inc., 713—451-3112.

7-ELEVEN is Alive & Well in WV. Oh Thank Heaven!

VAX TIME AVAILABLE. Call Ron Moehler, (412) 321-7400, for very competitive rates.

E.D.S. COMMERCIAL Insurance Division announces the Comprehensive Agents Business System, 203—678-1452.

FOR SOFTWARE AND Timesharing call Data Processing Service of Findlay at 419—423-8575.

SHEA & SHEA accountants, incorporated. 80 stonestown, san francisco 94132. 415/661-6860.

DESIRE CONTACT Educational RSTS users: Anderson, Los Alamos Schools. (505) 662-3596.

COMPUTER TIME. Information Engineering, Box 155, Mt. Clemens, MI 48043. 313—286-8488.

FREE PDP-11 SOFTWARE EXCHANGE. DPI, 3169 Holcomb Bridge Rd., Norcross, GA 30071. 404—447-4736.

RSTS CONSULTING. RODAV Business Systems, 3351 Demott Avenue, Wantagh, New York 11793.

SHA COMPUTERS. Systems for softdrink bottlers. A complete system for all accounting.

MEDIA TRANSFER SERVICES - We can transfer your data between any of the following media: RL01, RL02, RP05, RK05, RK06, RK07, RX01, and 800/1600 BPI 9 Track MT. We handle most DIGITAL directory structures on a fast turn around basis and also offer programming services for any special transfer problems you may have. Call (201) 584-4026, Advanced Digital Data, 354 Route 206, Flanders, New Jersey 07836. Advanced Digital Data, Inc., 354 Route 406, Flanders, NJ 07836.. I.D. #3.

RSTS/E SYSTEM Performance Monitoring Software. Bits, Bytes & Pieces. POB 730, Rough & Ready, CA 95975.

DON'T WRITE - GENERATE YOUR REPORTS WITH RG2. Peter Dick, 267 Goldhawk Road, London, England.

SOFTWARE TECHNIQUES, INC., is happy to announce two new employees - Jerry Schnaus and Kevin Herbert.

KICK CICS — GO VAX. Brian, Canada.

FOR SALE

FOR SALE: RX11-BA and six RP02 Disk Packs, 816—834-3311.

SALE PDP 11/40 2 RK05's 160 KB Core Memory. 516—929-8500 - 238.

FOR SALE: RPG-II License. Reasonable. Central Oregon Community College (503) 382-6112, ext. 257

FOR SALE - IMSAI 8080 - send for list - SHCI, 42 W. 39th St., NYC 10018.

FOR SALE - CORE MEMORY (DEC 96K words). Call - L.R. Christiansen Co. 309—762-8886.

FOR SALE: IBM 029 Keypunch, $500.00. Phone (219) 982-2141, ext. 281.

USED RK05 DISK PACKS for sale. Contact: Bill Ferris 203—888-1567.

WANTED

WANTED - USED DEC Tape Drive. $7 to $10,000 range. Call 207—775-5601, ext. 49.

WANTED: RSTS ECAP. MSOE, P.O. Box 644, Milwaukee, WI 53201. Attn: Computer Center.

HELP AVAILABLE

STATISTICAL CONSULTATION, Dave 412—561-2569, eve., weekends.

EXPERIENCED FREE LANCE Basic Plus, RSTS/E Programmer Analyst, Boston area, call 926-6025.

FREE LANCE PROGRAMMER/ANALYST. Get the manpower to finish that project on time, or design and organize an upcoming one. 6 years RSTS experience. Box 12302, Lexington, Ky. 40582.

POSITIONS AVAILABLE

PROGRAMMER/ANALYST
Programmer/Analyst with experience in RSTS/E and Basic Plus II needed to assume challenging position developing new retail point of sale systems. Systems developed will be for in-house use, as well as for marketing to other retailers. Equity position in software company available for the right applicant. Excellent salary, bonus program, group insurance, and employee stock ownership plan. Please send resume and salary requirements to:

Mr. Ahmed V. Ismail, President
Studio Camera & Electronics
20229 Mack Avenue
Grosse Pointe Woods, Michigan 48236

POSITIONS AVAILABLE

RSTS/E PROFESSIONALS
Positions now open in exciting WALL STREET projects.
TOP MONEY —
SUBCONTRACTORS WELCOMED.
Send resume or telephone number to:
LINK SOFTWARE INC.
P.O. Box 208, Wall Street Station
New York, 10268

BASIC+ PROGRAMMERS WANTED
Call NCA 408—245-7990, x313.

PROGRAMMING INSTRUCTOR - COBOL, BASIC Plus, Excellent Salary.
DIR Data Processing, 212—255-9322.

GARBAGE OUT — GARBAGE IN?

A prize if you tell us what's really happening and why!

LIST OF ADVERTISERS

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.

March 19, 1981
LOW COST SINGLE QUAD SIZE DISC CONTROLLER THAT INTERFACES UP TO TWO SHUGART, MEMOREX OR FUJITSU DISC DRIVES TO A DEC LSI-11

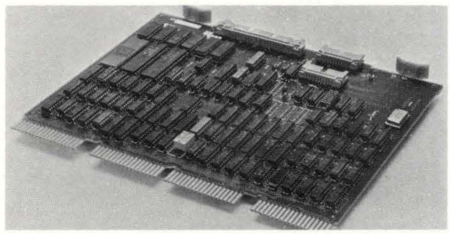
Garden Grove, California — A low cost, single quad size Disc Controller that interfaces DEC LSI-11, 11/2 or 11/23 microcomputers with two each Shugart SA4000, Fujitsu 2301 or 2302, and memorex 101 or 102 along with other hard disc drives having a Shugart SA4000 type interface, is now available from DILOG (Distributed Logic Corp.), Garden Grove, CA.

Designated the Model DQ401, the uP based controller emulates the DEC RKV-11/RK05 disc subsystem and is compatible with DEC RT-11 & RSX-11 software. It occupies one card slot of a DEC or DEC compatible LSI-11 Q-Bus chassis, and interfaces to the drive(s) via a jumper selectable single or dual cable arrangement.

Features include data buffer for the elimination of data late errors on the Q-Bus, automatic media flaw compensation, data error checking, on-board bootstrap and extended memory addressing to 256K bytes. In addition, there is an automatic self-test with indicator and data protect.

Price: $1,580. in small quantity.
Delivery: Stock to 30 days A.R.O.

DIRECT INQUIRIES TO: Mr. Les Alberts, Sales Manager, DILOG (**Distributed Logic Corp.**) 12800-G Garden Grove Blvd., Garden Grove, CA 92643. Phone: (714) 534-8950. NCC '81, Booths 5007-5009.



DILOG (Distributed Logic Corp.) introduces an inexpensive disc controller on a single quad size board that interfaces up to two Shugart, Memorex or Fujitsu Disc Drives to a DEC LSI-11.

April 27, 1981
COMDESIGN EXPANDS MULTIPLEXER LINE

Goleta, California — ComDesign, Inc. has released an enhanced version of its TC-3 Terminal Concentrator for PDP-11 and VAX users. The TC-3 Concentrator reduces the cost of adding remote terminals to DEC computers by combining DZ11 capabilities with statistical multiplexing.

The TC-3 Concentrator system consists of a Local Unit and one or two Remote Units. The Local Unit is functionally equivalent to a DEC DZ11-A, except that the terminal ports are in the Remote Unit instead of on the computer distribution panel. The Local Unit and the 4 or 8 port Remote Unit(s) are connected via a single communications link. A second Remote Unit can be connected to the Local Unit via a second link.

The TC-3 Concentrator can reduce the communications data rate requirements to 1/2 - to - 1/4 the combined data rate of terminals connected to the system. The full-duplex data link can be synchronous or asynchronous, 1200 to 9600 bps. No software changes are required to use the system.

User response to ComDesign's enhanced version of the TC-3 Concentrator has been excellent. A recent user, Intercontinental Forwarders, Inc. of Peabody, Massachusetts writes, ... "we are impressed ... have already recommended the unit to a number of other RSTS/E users and shall continue to do so in the future."

Price of the TC-3 Local Unit is $3,200, the 4-line Remote Unit $1,950 and the 8-line Remote Unit $2,650, FOB Goleta (Santa Barbara). Delivery is 30 days ARO.

For more information contact: **ComDesign, Inc.**, 340 South Kellogg Avenue, Goleta, CA 93117, (800) 235-6935 **toll-free** outside CA; (805) 964-9852 inside CA; TWX - 910-334-1189.

March 20, 1981

Louisville, Kentucky — Locally headquartered Amcor Computer Corp. (AMCOR), a leading developer and supplier of computer software, has been acquired by Houston-based Kaneb Services, Inc.

Thomas E. Aubrey, founder, president and chief executive officer of AMCOR, said the

DIRECT.TEC

. . . continued from page 38

```
! Clear screen and display heading              !
ET&512"N 2:W < G5 > J -1W '
HK M2

! Main display loop                             !

                           ! Load previous PPn  !
G* @-S/[/ \U2 @S/,/ \*256+Q2U2 HK
OUO                        ! Clear counter      !
< %0 ©[ %4                 ! Bump counters      !
  G* HX4 J @S/]/ OK        ! Show only file     !
  @S/./ R 6-.< @I/ / >     ! Fill filename      !
  @S/</ R 10-.+3< @I/ / >  ! Fill protection    !
  C 4-(Z-.)< @I/ / >       ! and move out       !
  ZL G5 HT HK
  :@EN//;                  ! More files?        !
                           ! Extract new PPn    !
  G* @-S/[/ \U3 @S/,/ \*256+Q3U3 HK
                           ! Break if different !
  Q2-Q3"N -1M3 G5 M2 %5 OUO Q3U2 '
>                          ! Loop complete      !

! Print totals                                  !
Q0-Q4M3

! Check "grand" total                           !
Q0-Q4"N
@I/
 Grand total of / Q4\ @I/ file/ Q4-1"N @I/s/ ' @I/ in /
Q5\ @I/ account/ Q5-1"N @I/s/ ' @I/ for / G1 G5
HT HK
'

! End Routine                                   !

!END!

Q1"E G5 ' HT HK
'

! Now run the macro                             !

HXO HK MO

! Turn off indirect file (like CCL entry)       !
:@EI//

! Exit if entered by CCL                        !
Q1"N EX ' ]6 ]5 ]4 ]3 ]2 ]1 ]0

! Examples of DIRECT.TEC output -- all examples use the FILES CCL

FI *.TSK
 Name .Ext  Protect    SY:[1,2]
TKB   .TSK   <104>
SLOTKB.TSK   <104>
MAC   .TSK   <104>
LBR   .TSK   <104>
PAT   .TSK   <104>
EDT   .TSK   <104>
CSPCOM.TSK   <124>
SORT  .TSK   <104>
BPCREF.TSK   <104>
BPCRF1.TSK   <104>
B2CREF.TSK   <104>
RNO   .TSK   <124>
BP2DA .TSK   <104>
RMSBCK.TSK   <104>
RMSRST.TSK   <232>
RMSDFN.TSK   <104>
RMSDEF.TSK   <104>
RMSDSP.TSK   <104>
RMSCNV.TSK   <104>
RMSIFL.TSK   <104>
DIRT  .TSK   <232>
BASIC2.TSK   <104>

 Total of 22 files in SY:[1,2]??????.TSK

Ready

! Try more than one account

FI [101,*]
 Name .Ext  Protect    SY:[101,0]
RML100.DAT   < 60>
RML110.DAT   < 60>
RML110.TSK   <232>
RML100.TSK   <232>
RML400.TSK   <232>
RML200.TSK   <232>

 Total of 6 files in SY:[101,0]

 Name .Ext  Protect    SY:[101,1]
RML500.TSK   <232>
RML600.TSK   <232>

 Total of 2 files in SY:[101,1]

 Grand total of 8 files in 2 accounts for SY:[101,*]??????.???

Ready

! Do a directory that fails

FI [0,1]FOO.FOO

%No files matching specification SY:[0,1]FOO.FOO

Ready

! Examples complete.
```

recently completed merger "provides AMCOR with the necessary financial and human resources to take advantage of existing and new market opportunities in a much earlier time frame."

AMCOR, located at 1900 Plantside Drive, will continue to operate from Louisville. Aubrey will remain president and chief executive officer.

Since 1970, the company has been in the minicomputer business providing application software and a comprehensive data base management system for Digital Equipment Corporation's PDP-11 series of computers.

According to Aubrey, the U.S. Navy uses the sophisticated system for management control on one of its ships.

"A similar program also is being used by Litemaster Products Ltd., a firm located in Johannesburg, South Africa," he added, "and by Hitachi Metals Ltd. in White Plains, N.Y."

Other customers, Aubrey said, include Equitable Life Assurance Society of the United States and Cooper Industries in Texas.

AMCOR will become part of **Kaneb's ARC Automation Group, Inc.**, whose member companies provide computer products and services to independent insurance agents, funeral homes and, most recently, have developed and marketed a family of microcomputer terminals for retail, service and commercial operations.

According to Dr. Robert Bower, Jr., president and chief executive officer of ARC Automation Group, "AMCOR gives us the software expertise which, when combined with our hardware and computer application experience, creates new market opportunities for the group."

ARC Automation Group is headquartered 90 miles north of Houston in Bryan-College Station, Texas.

Kaneb is a diversified international energy resources and financial services company with primary operations in oil and gas; coal and general contracting; and financial intermediary and specialized computer services.

March 9, 1981
NEW WORD-11 FILE SECURITY SYSTEM FOR DEC RSTS/E USERS

West Palm Beach, Florida — RAXCO Inc. announces the availability of a new software option designed to specifically protect confidential WORD-11 Files. This new option prevents access to classified or confidential WORD-11 data by unauthorized personnel. Even privileged users may be excluded from data files secured by RABBIT-4 software. Coupled with normal security measures, RABBIT-4 will ensure file integrity while monitoring all file access attempts.

For more information contact: **RAXCO, Inc.**, 3336 N. Flagler Drive, West Palm Beach, FL 33407; Telephone (305) 842-2115.

April 23, 1981
SSI'S 'PEACE OF MIND' PRINTER SYSTEM DESIGNED FOR ZERO DOWNTIME

Fort Lauderdale, Florida — The "Peace of Mind," a redundant printer system designed for users requiring zero downtime in printer functioning, has been announced by Southern Systems, Inc., (SSI) computer printer company.

Formally called the PS-10, the "Peace of Mind" consists of a primary printer, a lower speed back up printer and a switching mechanism.

The switching mechanism for the "Peace of Mind" can be connected with the computer and the two printers either with parallel or serial (synchronous or asynchronous) interfacing.

Cost of the "Peace of Mind" system is usually less than the cost of a single comparable printer from a prime computer vendor, explained James W. Rule, SSI vice president/marketing.

"The cost-efficiency of the Peace of Mind is even more apparent since the redundancy factor guarantees zero downtime and is obviously totally dependable," said Rule.

Various combinations of speeds in the dual-printer systems are offered by Southern Systems.

"A user who needs a 1,500 lpm printer may want the 900 lpm printer as his back-up for the Peace of Mind system," explained Rule. "Other customers may require a 900 lpm band as the primary printer with a 200 lpm dot matrix or a 300 lpm band as the back-up. In each and every combination the cost is below that of a single printer from a computer vendor," said Rule.

Depending on the printer combination within the system, costs will range from about $6,000 to about $55,000.

Southern Systems, Inc. headquarters is located at 2841 Cypress Creek Road, Fort Lauderdale, FL 33309, (305) 979-1000 or (800) 327-5602, Telex 522135.

April 10, 1981
FINAR INTRODUCES ENHANCED SOFTWARE AT ANNUAL USERS' MEETING

Denver, Colorado — Mr. Michael Hulme, Marketing Director at Finar Systems Ltd., announced the availability of FINAR version 5.05, and distributed tapes of the enhanced software to attendees of the Annual Users' Meeting held on April 9 - 10, 1981.

FINAR, the Financial Analysis and Reporting language, carries out budgeting, forecasting and modeling on DEC PDP-11 and VAX-11 computers. The new FINAR version 5.05 offers worksheet consolidation — a faster and more efficient technique to consolidate results, as well as new methods of data input, enhanced calculation features and improvements to the report generator.

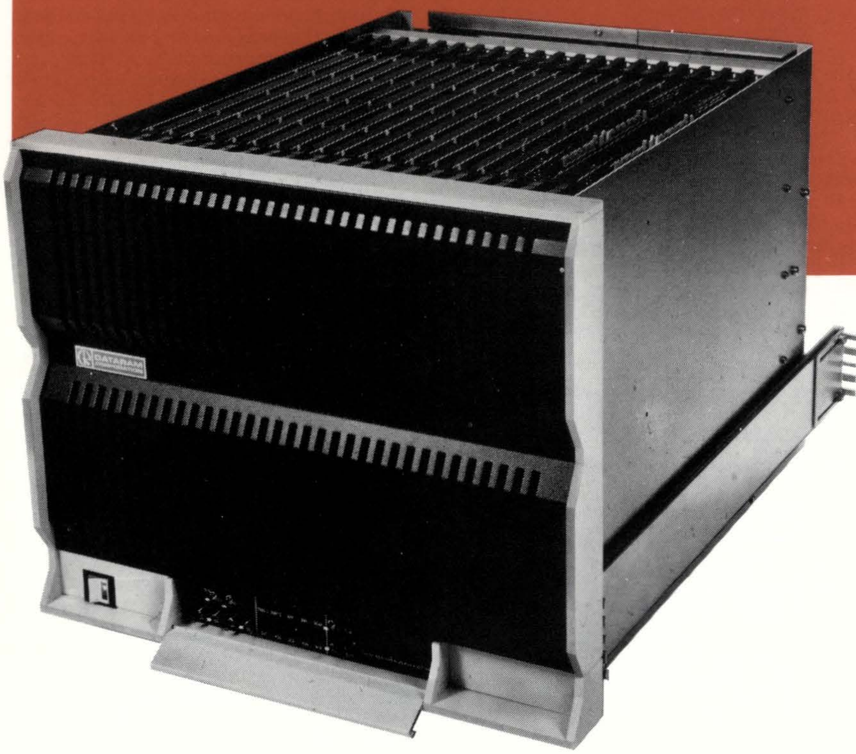
New this year to the Users' Meeting was an Advanced Techniques Seminar with several in-depth presentations on the new worksheet consolidation feature, efficient model writing, simulation and "what-if?" analysis, and how to get the most out of FINAR and your computer.

The meeting concluded with discussions on new features to be included in FINAR version 6.0, and the informative user application presentations — this year made by an oil company that uses FINAR to forecast oil and gas prices in a fluctuating economy, a paper/pulp mill making long range profit plans, a CPA firm integrating FINAR with other systems, and a manufacturer of satellite communication systems that uses FINAR in a decentralized organization.

For further information call or write: Michael Hulme, **Finar Systems Ltd.**, 6000 E. Evans, Suite 2-300, Denver, CO 80222, (303) 758-7561.

THE INDUSTRY STANDARD

BULK CORE   BULK SEMI



ALSO FROM DATARAM
JUST ANNOUNCED
PDP 11/44
MEMORY

Add a new dimension of speed and reliability to your minicomputer with economical, high-capacity BULK MEMORY from Dataram.

The world leader in minicomputer-compatible disk emulation systems. Dataram's wide range of disk emulations — twice as many minicomputer interfaces as anyone else! — brings the proven performance of BULK MEMORY to your application requirement.

Dataram's BULK CORE and BULK SEMI systems operate at speeds which are orders of magnitude faster than the mechanical disks which they replace, and do it with the reliability inherent in all-electronic devices. What's more, BULK MEMORY provides up to 8.0 MB in a 15¾" chassis, and offers dual-port capability to enable BULK MEMORY to be shared by two host minicomputers.

If you have a minicomputer and are looking for a way to get more for your storage dollar, Dataram has a BULK CORE or BULK SEMI system ready to work for you. If your minicomputer is not listed below, tell us about it. We'd like to add your name to our growing list of BULK MEMORY users.

MEMORY FROM THE LEADER

DEC, LSI-11, and PDP are registered trademarks of Digital Equipment Corporation. Eclipse and Nova are registered trademarks of Data General Corporation.

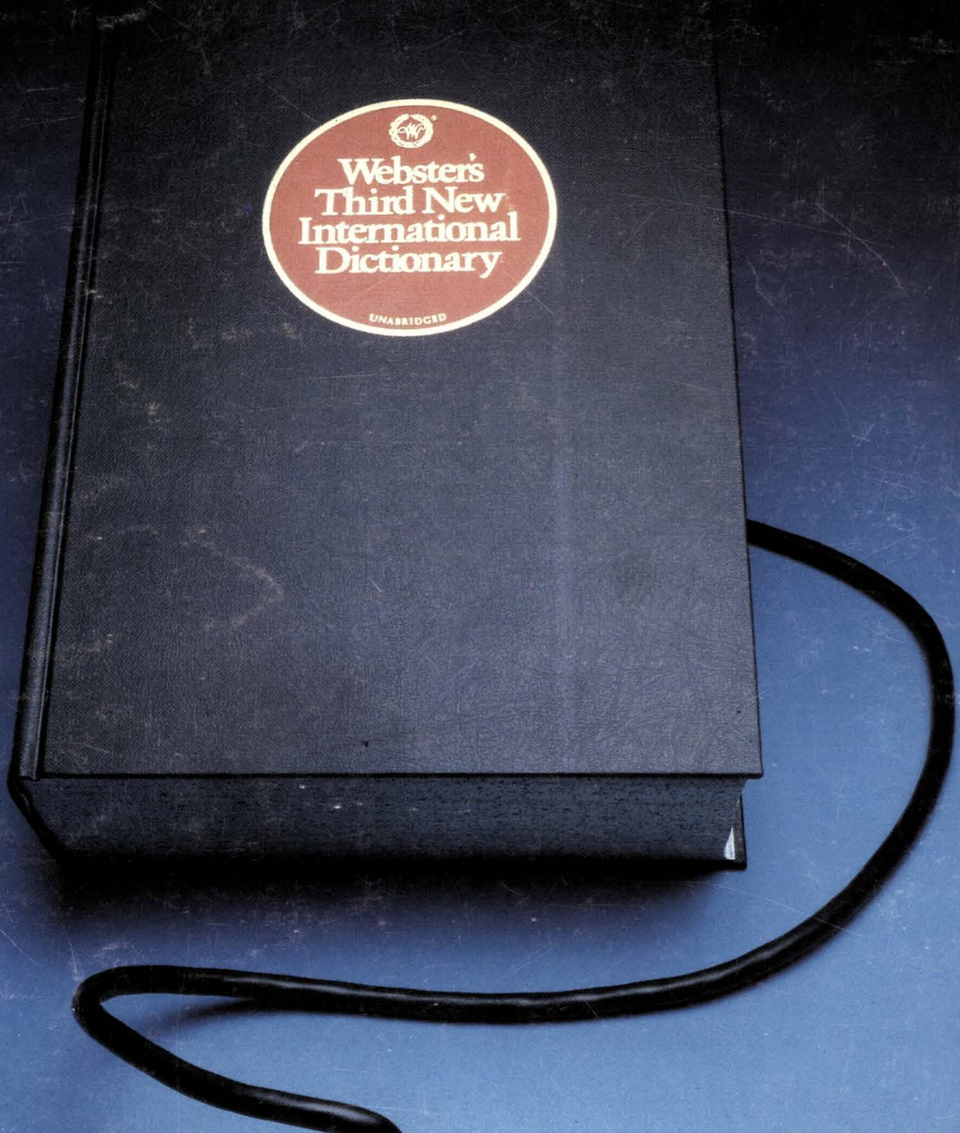
DATARAM CORPORATION

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

	DEC® PDP®-11	DEC LSI-11®	DEC PDP-15	DATA GENERAL NOVA®	DATA GENERAL ECLIPSE®	ROLM 1602	PERKIN ELMER	MODCOMP	HONEYWELL 316 and 516
BULK CORE	✓	✓	✓	✓	✓	✓	✓	✓	✓
BULK SEMI	✓	✓	✓	✓	✓		✓		
DUAL PORT	✓	✓		✓	✓				

Canada: Ahearn & Soper Ltd., 416-245-4848 • **Finland:** Systek OY, (80) 73 72 33 • **France:** YREL, (01) 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) 543-2077 •
India: Infosystems Private Limited, 79281 • **Israel:** Minix Computers & Systems Ltd., 03-298783 • **Japan:** Matsushita Electric Trading Co., Ltd., 06 (282) 5111

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.

dpd
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