RSTS PROFESSIONAL

Volume 4, Number 2

April 1982
$10⁰⁰/issue, $35⁰⁰/year

INSIDE:

Financial Decision Support Canned For Easy Use!

Fast Flexible Information

MAPS™ IS FRESH! It's the ideal decision support software product for today's financial manager. With MAPS you get fast, fresh, flexible information handling the moment you want it. Plans, forecasts, models, reports can all be created and manipulated right in your own department. Changes that affect your company can be reflected and evaluated on the spot. And best of all, MAPS is canned for easy use! Call Ross Systems right now. Ask for a demonstration of MAPS. You'll be amazed what a little fresh decision support can do. Currently available on worldwide timesharing, or purchase for DEC's,* RSTS/E and VMS operating systems. Contact Ross Systems for more information today.

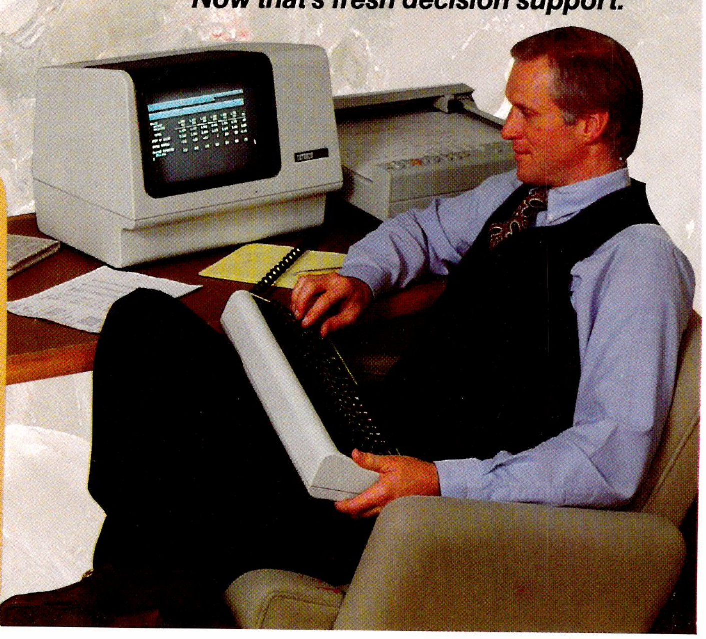
*DEC is a trademark of Digital Equipment Corporation.

ross systems

1800 Embarcadero Road
Palo Alto, CA 94303
(415) 856-1100

Regional offices: New York, Dallas, San Francisco, Los Angeles

"Now that's fresh decision support."



MAPS meets your daily requirements:

Financial reports	100%
Financial planning	100%
Consolidations	100%
Financial modeling	100%
Performance forecasts	100%
Full color graphics	100%

CIRCLE 1 ON READER CARD

Adding a printer?
Add-on the printer company instead.
Southern Systems.

You need maximum performance and minimum downtime from your printer. So, don't add-on just a machine. Add-on the unmatched printer expertise of Southern Systems, recognized by users as the leader in printer systems, from technological innovation through installation to the long-term service you need.

In more than 4,000 installations, Southern Systems has proven it's **the** best source for printer systems from 200 to 2,000 lpm. Printer systems that are guaranteed compatible with your computer. Printers that are serviced by SSI specialists nationwide. Printers that work for you at less cost than any other printer source can offer.

Add-on a Southern Systems printer system and you add-on Southern Systems. Just ask our customers.

SSI

Southern Systems

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

Tell me about your printers from:
□ 200–300 lpm
□ 600–900 lpm
□ 1000 plus lpm

My computer system is________________
Name________________
Title________________
Company________________
Address________________
City________State________Zip________
Telephone________________CW

CIRCLE 2 ON READER CARD

SOFTWARE

CONTRL

LINK YOUR TERMINAL TO ANY KEYBOARD ON THE SYSTEM FOR:

USER TRAINING AND SUPPORT

DYNAMIC SECURITY

REMOTE DEMONSTRATIONS

Link Up Whenever You Like. When you run the program CONTRL at your terminal you may elect to capture and link up with any keyboard on the system. CONTRL performs the link requested immediately. There is no interruption to the user's task. It makes no difference what the user may be doing. The user may be in mid-keystroke or logged off the system. The user's keyboard may even be turned off.

Link Is Invisible to the User. The linking process is invisible to the user. Except on a heavily loaded system the user will not notice so much as a hesitation from one keystroke to the next when the link up takes place. In fact, an inspection of job status will appear normal to the user.

Do Remote User Training. When a new procedure or application is put onto your system, CONTRL may be used to do remote training. The user logs onto the system and then calls you by telephone. You run CONTRL at your terminal. While speaking to the user you link up with the user's keyboard. Now you walk the user through the new procedures while you watch at your screen. Each user keystroke together with the system's responses is presented to your terminal.

Interact for Remote User Support. With CONTRL you may interact with the user. Anything you do at your keyboard after linking with the user is as though you did it at the user's keyboard. When a user calls you with a question or concern about his job you may link up and give assistance directly from your keyboard.

Provide Remote Demonstrations. If you need to demonstrate an application to a remote group, CONTRL will solve the problem. Don't pay the travel costs to get your team together with their team to see some programs run. Consider what many are now doing with CONTRL. The application review team gets together at their own site. They gather around a terminal that is logged into your system. Then they call you on the telephone. Most often they will use a speaker phone. As you exercise the application at your terminal they see everything at their remote screen. If you wish, they may be instructed to interact with the application themselves. This serves to convey the dynamic nature of your demonstration, while involving your listeners.

Inspect User Activity. CONTRL allows you to inspect a user's activity on the system. It is often necessary for management to observe training effectiveness among their clerical personnel. With CONTRL a clerk's grasp of an application can be observed unintrusively.

Do Dynamic System Security. The inappropriate, unwise, and covert use of your system can be monitored. Experience with CONTRL in this area indicates that knowledge of its existence on the system and its potential for invisible use on selected keyboards is an effective threat to covert users. For inappropriate and unwise use
the system, CONTRL gives manag
ment a means for taking specific co
rective action.

Keep a Log File of the Activit
This is well worth noting. A complet
log file of the user activity is kept b
CONTRL. Every keystroke entere
at either your keyboard or the user
keyboard goes to the log file togeth
with every response from the system
The session in its entirety is capture
The keystrokes are underlined to di
tinguish the user from the syste
when the log file is played back.

Release Link Whenever You Lik
The link can be released immediate
and at any time. Releasing CONTR
has no effect whatever on the user
job. The user may be in mid-keystrok
or logged off. The user's terminal ma
even be turned off when the keyboa
is released.

Some of the CONTRL Option
You may get a log file or elect to turn
off. You may disable the user's key
board or prevent output from the us
er's job from going back to the user
screen. CONTRL gets its name fro
being in control of the linked keyboar
Everything that moves between th
user's keyboard and the system goe
through CONTRL.

Clyde

How to Get More Informatio
Call Janet at (617) 275-6642, or write
Clyde Digital Systems, Inc., P.C
Box 348, Bedford, MA 01730.

CIRCLE 6 ON READER CARD

Contents

THE VAX-SCENE

Coming . . .

- ■ RSTS Security
- ■ A DEC of Cards
- ■ Word Processing
- ■ Another Dungeon Map (in living color)
- ■ Disc Inversion Map
- ■ TYPE.RTS
- ■ VT100 Printer Port
- ■ More Networking
- ■ More VAX Macro
- ■ More RSTS News
- ■ More . . .

The RSTS Professional Magazine, April 1, 1982, Vol. 4, No. 2. Published bi-monthly. Single copy price $^{\$}10^{00}$, $^{\$}35^{00}$ per year. Published by M Systems, Inc., 753 Johns Lane, Ambler, Pa. 19002-0245, telephone (215) 542-7008. Send all correspondence and change of address to: Box 361, Ft. Washington, Pa. 19034-0361. Copyright © 1982 by M Systems, Inc. No part of this publication may be reproduced in any form without written permission from the publisher.

From the editors. . .

IS IT REALLY FAIR?

Carl Marbach

The late Chief Justice Earl Warren of the United States Supreme court would sometimes stop an eloquent argument from an attorney at the bar with a wave of his hand, "Yes, yes, yes", he would say, "but is it really fair?" Sometimes all the rational arguments in the world don't suffice when people are at stake. My question to DEC is, "why are you leaving your loyal PDP-11 people alone at the top? What do I do now that the 11/70 is going to be gone?" And they reply, "Go VAX young man".

I attended a meeting recently with a major OEM and DEC people discussing the subject of what do they sell now that they won't be able to get 11/70s after next summer. The OEM hunched up his shoulders, gruummpffed a few times, took a deep breath and said, "now let me get this straight: I can expect 80% of the performance, 120% of the price, a lower discount schedule, and I must make a major software conversion to VAX-11 BASIC." He was comparing 11/70s to a VAX 11/780. It is also true that he runs more than one 11/70 with 120+ terminals, and doesn't think that 96 terminals is unusual for his software on the "70". The VAX SPD only admits to 96 terminals and with DZ's we know it will be character bound.

Memory is getting very cheap. Computers have more bang for the buck. Packaging has improved and downtime is less and less frequent. Programmers make more money than ever and are harder to find. Software is now a larger investment on most machines than the machine itself. All this argues for an environment where programmer time and effort are spent in the most productive way; Building co-trees and overlays to fit into the 16 bit addressing space is inefficient use of their expensive time. 32 bits solve lots of problems. The operating system can be more complex(?) and do more for the user. Languages can be compiled faster and optimized more into native mode rather than the 16 bit "threaded code". Programmer space limitations becomes a thing of the past. All in all 32 bits beat the devil out of 16 bits.

Despite all these facts, there are many people and small businesses out in PDP-land with large investments in hardware and software. It really is a "bet your business" move when you marry a computer. OEMs, software houses and bureaus are now required to shoulder a heavy burden in moving to the 32 bit VAX. Wouldn't it have been better if we had been given a good migration tool rather than the 'it's easy to convert to VAX BASIC and RMS' panacea that has been handed us. Shouldn't we be treated to a product that fills the needs of an established user base, instead of being forced into a world we don't need or may not want. Are we really to accept 80% of the performance, 120% of the price and less attractive terms? I know, 32 bits and VAX makes a lot of business sense to DEC; and 32 bits is the wave of the future. Yes, Yes, Yes; but is it really fair?

SOFTWARE SOUP

Dave Mallery

Recently, I wrote an editorial praising two software packages that I had bought and especially liked. Overnight, I was quoted in ads in every computer journal under the sun. I never realized the power of my 'speechlessness'!

I have recently experienced another 'fallout' from that editorial. It seems that every nascent software house in the world wants me to have a free home demonstration of their product in hopes that I will also recommend their product.

Obviously, this is not what I had in mind.

There are two reasons why I can't be everyone's beta test. First, I am very busy. Second, I am also an author and seller of software (under another hat).

So, once and for all, a policy on software editorializing in the RSTS Pro:

1) We will review anything and everything.

2) We will review only what we have bought.

3) We don't want any free home demonstrations.

Another major area of concern. We publish a LOT of programs. We even make the sources available on magtape. We are a little magazine, not a three billion dollar computer company.

ALL PROGRAMS PUBLISHED IN THE RSTS PROFESSIONAL ARE WARRANTEED TO PERFORM NO USEFUL FUNCTION. THEY ARE GUARANTEED TO CONTAIN BUGS. THEY ARE DESIGNED TO GET YOU OFF YOUR BUTT AND THINKING. THEY ARE INTENDED TO EDUCATE AND ENTERTAIN. THEY ARE PUBLISHED ON THE PREMISE THAT IT IS BETTER TO SPREAD PEOPLES' BEST EFFORTS AROUND EVEN IF THERE IS AN OCCASIONAL PROBLEM. IF YOU USE THEM, MAKE THEM YOUR OWN, AND YOU WILL NOT GO WRONG. IF YOU WANT CLASS 'A' SOFTWARE SUPPORT, GO PAY DEC SIXTY FIVE DOLLARS AN HOUR.

I hate to sound so clear, but I have had it with complaints.

See y'all in Atlanta! ♥

RSTS PROFESSIONAL

Editors
R.D. Mallery
Carl B. Marbach
Assistant Editor/Advertising
Helen B. Marbach
Assistant Editor/Operations
Peg Leiby
Administrative Assistant
Hope Makransky
Subscription Fulfillment
Kathi B. Campione
Claire Hollister
United Kingdom Representative
Pauline Noakes
RTZ Computer Services Ltd.
P.O. Box 19, 1 Redcliff Street
Bristol, BS99-7JS
Phone: Bristol 24181
Contributors
Rudy Bazelmans
Joe Doyle
Steven L. Edwards
Ken Fleming
Richard W. Hill
Patrick Holmay
Austin Kinsella
Michael H. Koplitz
Bob Meyer
Stephen Munyan
Jonathan M. Prigot
Jacob F. Ruf
Robert Schilmoeller
David Spencer
Bob Stanley
Brad Smith
Stanley Zuk
Cartoons
Douglas Benoit
Game Verification Team
Bill Marbach, Mary Marbach,
Sydra Mallery, Sam Mallery
Design & Production
Grossman Graphics

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

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

LOW COST RSTS/E NETWORKING.

DMG/NET provides RSTS/E users with easy access to packet-switched (X.25) networks. It permits two-way file transfer and interactive dialogue with other RSTS/E systems and locally initiated communication with non-RSTS/E systems. From a RSTS/E host to other RSTS/E computers, to other DEC computers, even to non-DEC computers...communication is quick, simple and extremely inexpensive.

REDUCES COSTS

No matter how large or how small your RSTS/E system is, DMG/NET can substantially reduce communication costs by utilizing packet-switched (X.25) networks. With DMG/NET, you can save up to 90% of your cost of communication, compared to "dial-up" or leased lines.

EASY TO USE

Even a non-technical person can access the entire communication network quickly and easily with DMG/NET. There is no need to memorize a long series of network codes, numbers and procedures. Instead, a user accesses a remote database with only the familiar local sign-on and a short, easily remembered identification code...DMG/NET does the rest.

What's more, DMG/NET allows any locally connected terminal to access any specified remote database, eliminating the need for separate terminals, complicated switch boxes or terminal setting changes.

To find out how DMG/NET can meet *your* RSTS/E networking needs, contact Digital Management Group Ltd.

DMG/NET

RSTS/E and DEC are registered trademarks of Digital Equipment Corporation
DMG/NET is a registered trademark of Digital Management Group Ltd

DMG

Digital Management Group Ltd.

4800 Yonge Street, Willowdale, Ontario, Canada M2N 6G5
Telephone: (416) 225-7788

CIRCLE 88 ON READER CARD

LETTERS to the RSTS Pro . . .

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

Nice magazine! Keep up the good work. I'd like to comment on the "Top-down" article on p.8 in the Dec. 1981 issue. I make a living rewriting and reworking systems that are designed by people who think there is just ONE way to properly implement. I've rewritten beautifully structured code and achieved speed improvements of 20 times. Were I to utilize ALL of the structure rules that I learned in college, I would be cranking out code that ran slow, overlayed too much, and was bigger than necessary. Structured design is a tool; but, as with all tools, if it isn't used wisely, it can't be used to create a masterpiece.

Steve Roy, Diversified Consulting Co.
Bloomfield, CT 06002-0284

—

If you clowns think I'm going to pay a 75% rate increase, you're crazy. N.D. Harris, IN

Look closer, Dale. Our rate has gone DOWN. We increased the number of publications per year not the cost per issue. Old rate: $25. for 4 ($6.25 each); New rate: $35. for 6 ($5.83 each). That's a 7% DECREASE. Com'on back!

—

I recently got myself a subscription to your wonderful magazine. Each new copy gives me a few hours of pleasant reading and contains a lot of valuable information.

The December '81 issue however, contained at least two errors. One of them could be serious the other one makes me wonder where the author of the article did his writing.

On p. 81 David Leffen wrote that memory has to lie physically between the CPU and the Able Cache/434. This is exactly where the memory should not be.

The Able CACHE memory's have to be installed BETWEEN the CPU and the MEMORY it has to cache. Normally the 434 will replace the jumper between the CPU-backplane and the next backplane. All memory can then be installed in the second backplane. The CPU-backplane can be used for simple I/O devices like DL-11's, p.t. equipment and the like. DMA devices should follow the memory's.

Ten years of experience with PDP 11's gives me the following preference:

First	:	CPU plus attached boards
Then	:	Simple I/O devices (non DMA)
If applicable	:	External cache memory's (like the Able 434)
Always	:	All memory
Followed by	:	Fast DMA devices
And last of all:		Slow DMA devices

And keep the bus as short as possible. A good system will even run without any problem with a DC bus-load of more than 20.

On page 16, the right hand column 8 lines from the bottom, Michael Schwartz states that a data encryption utility should be available for use in BATH. I wonder why. Is he afraid of spying submarine's circling underneath the water surface in his tub? Does he store confidential information disguised as tooth paste or do we need plumbers for maintenance on these utilities? On reflection the above opens complete new uncovered grounds for research.

Keep up the good work!

Jan Willem Brier, Datacare B.V.
3700 AA Zeist, The Netherlands

—

Dear Dave and Carl,

Thank you for your kindness in dedicating your February, 1982 issue to me. Besides expressing my heartfelt appreciation for your act of friendship, I must tell you that I want to share this recognition with the "RSTS Team", to whom the credit must belong.

I speak not only of legendary superstars such as Mark Bramhall and Anton Chernoff. To name but a few, Jim Miller, Jim "Wooly" Wooldridge, Nancy Covitz, Joe Mulvey, Mark Goodrich, Rich Witek, Steve Morris, Bill Noyce, Jim Condict, Andy Riebs and Bill Sconce are among the scores of competent professionals I have had the privilege of working beside over the years. And I speak not only of the software engineers (more familiarly known as "developers"), but also of the writers, who are the unsung hero(in)es, as well as Software Support specialists such as Martin Minow, my outspoken "conscience."

Equally important, it was my pleasure to have met hundreds of users, talked to and listened to them. Some of them, symposia coordinators, SIG chairmen and other leaders, I grew to know better than others as we worked together, but the collective experience of interacting with the entire community of the RSTS SIG has been most educational to me. I thank them all.

Lastly, I would like to reciprocate by wishing you and your readers continued success with RSTS, as I am sure you will have. And whatever I am working on, RSTS will always be close to my heart.

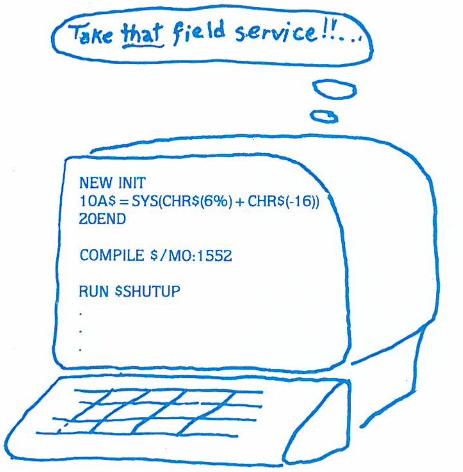
Yours sincerely, Simon Szeto

Thanks again, Simon, you are a real "gentleman and scholar" from tip to toe. Just to make sure that RSTS will stay close to you, we would like to grant you a lifetime subscription to the RSTS PRO. Therefore, we are returning your subscription check to you. You might notice RSTS right here: Your own DEC credit union, on which this check is drawn, uses RSTS. Why, RSTS wrote this check! Of course, they are moving to VAX.

Please keep in touch and maybe even be our "conscience". Carl and Dave

—

Many thanks to you and Paul R. Laba for his fine article on FIP's Alignment Algorithm (Sept. '81). This undocumented design feature has bitten me several times. Please note: If you have a disk giving "Bad Directory for Device" errors which the various CLEAN's won't fix, there is a good chance that the FIP Alignment Algorithm is the problem. It does seem like DEC could at least report these alignment problems as errors in the CLEAN's.



```
NEW INIT
10A$ = SYS(CHR$(6%) + CHR$(-16))
20END

COMPILE $/MO:1552

RUN $SHUTUP
.
.
.
```

Your magazine has proven so valuable that my copies are disappearing! I need replacement copies of several issues.

Thanks, John W. Nunnally, Director
Administrative Computing, Harding University

Your copies are on the way, John. Thanks for the cartoon, it speaks for a lot of our readers.

I just entered the FILMAP program in the Feb., 1982 issue. Features such as this are very useful to those of us who have to maintain more than one system with little or no help. Keep up the good work.

However, it did have a problem of getting stuck in a look under certain conditions. I believe the enclosed lines will take care of the problem. Line 2010 is just a slightly different way of arranging the statements that ensures getting to line 2000 if ppn.inx% is negative, regardless of the value of fil.cnt%. In line 2015, statement 7, the goto was changed from 2000 to 2010, to allow wildcard ppn searches where the first account in the range is zeroed.

```
LISTNH 2010
2010    gosub 10000%                                                  :&
        if ppn.inx%>=0% then goto 2012%                                &
        else if fil.cnt% then goto 2000%                               &
        else print 'no matches for ';f$ : goto 2000%

Ready

LISTNH 2015
2015    ufd.$ = dev$ + '['+num1$(swap%(ppn%) and 255%)+','             &
                +num1$(ppn% and 255%)+']'                             :&
        open ufd.$ for input as file 1%                               :&
        s1$ = sys(chr$(12%))                                          :&
        s% = swap%(cvt$%(mid(s1$,13%,2%)))                            :&
        ufd.clu% = u%(31%,0%)                                         :&
        if s% = 0% then                                                &
                ppn.inx% = ppn.inx% + 1%                              :&
                goto 2010%                                             &
        !       return if no more                                      &
        !       ufd.$ = name of ufd to check                           &
        !       open the ufd for the lookups                           &
        !       check to be sure ufd not zeroed

Ready
```

Chris Rapp, System Manager
CSULA Computer Center

P.S. I wear large, if you're still giving away T-shirts.

Chris, your t-shirt is coming — thanks for the help!

—

I have just finished reading the "Benchmark DIBOL vs BASIC + 2" in your December issue and feel that a few comments are warranted.

1. The article is hardly a comparison between DIBOL and Basic + 2. It really compares RMS Isam file handling and DIBOL Isam file handling. It is very much a case of apples and prunes, is it not? On the one hand we have RMS written in macro with who knows how many thousands of man days behind it and on the other a very simple single key Isam structure written mostly in a high level language.

2. There would appear to be no justification for the comparison, given that DIBOLR is available which also utilizes the RMS file structures. Now that would be a comparison worth taking note of. I wonder what the I/O would be like when BASIC + 2 starts reading on its RMS overlays (especially for storing new records) while the DIBOLR program can do it all from its task image.

When I open the pages of RSTS Professional, I expect to read balanced, unbiased articles that have been filtered through a panel of competent referees. If you wish to print largely irrelevant comparisons hiding behind the term "Benchmark", then perhaps your publication should be called the RSTS Amateur.

I must confess to being a DIBOL fan ever since COS350, however, I think my comments are reasonable. I would just hate to think of someone being turned away from DIBOL because of an article in a professional journal.

Many thanks to Mr. Lomasky. I have never been to the Temple and Altar, but I will tonight.

Kindest regards and Best wishes,
Peter M. Jones, Systems Manager
Charles David Pty. Ltd.

Apples, prunes, professionals, amateurs, referees, fans — sounds like you're fightin' mad, Peter. We're sorry, but at least we brought you back to Temple!

Seriously, thank you for taking time to share you views with us.

—

I have just attempted to implement CALLER.BAS (v.3, #4, p.76) with a fair degree of success. However, the printed version contains some drawbacks.

1. The "Help" feature precludes use of the DEC HELP package.

2. The CCL parser expands commands to their

. . . continued on page 71

There really is a difference . . . .

THE DOPTER DIFFERENCE

DOPTER is an **easy to use** RSTS/E disk copying program which produces an **optimized** disk structure providing greater system throughput and improved response time.

Here's Why

With only two short commands, DOPTER **automatically** performs the following:

Properly places and pre-extends the MFD. Yes, the first cluster of the MFD must start at device cluster zero, but DOPTER only writes in the first block of the first cluster. All of the rest of the MFD entries are in the interior of the volume where they belong.

Places and contiguously pre-extends the UFD's. The UFD's, as well as the MFD, are only extended as much as is necessary to contain their current information **plus some room for expansion.**

Places the UFD's with the most activity toward the front of the MFD for quickest access.

With its **proprietary techniques,** produces better optimized MFD/UFD's which remain that way longer than does REORDR or any other process.

Deletes unused file attributes from source, task, and object library files saving UFD and cache accesses.

Places the most used files at the front of the UFD's for quicker access.

Places the most used files in the center of the active files. Places "unused" files separate from the active files leaving the active files more compactly placed.

Performs all steps starting with disk initialization and ending by "hooking" the output volume and installing the current SIL **without operator intervention.**

Please Send More Information To:

Name________________________

Company______________________

Address______________________

City_________________________

State______________ Zip________

Telephone_____________________

RSTS/E is a registered trademark of Digital Equipment Corporation

Furthermore

DOPTER optimizes file cluster size and makes files contiguous where appropriate.

DOPTER preserves previous bad block information on the output volume so that patterns need not be re-run.

DOPTER prevents accidentally copying an older DOPTER'd volume onto a newer one.

DOPTER preserves all accounting data.

DOPTER allows all volume, UFD, MFD, and file defaults to be manually overridden.

For More Information

If you would like more information on the DOPTER DIFFERENCE, mail the coupon, circle the Reader Card number, or call us. We'll send you a free copy of the DOPTER User Manual and License Agreement.

Single CPU price: $2,500.

SPH

System
Performance
House, Inc.

5522 Loch More Court · Dublin, Ohio 43017 · 614-265-7788

CIRCLE 108 ON READER CARD

Logging Into An Account Without LOGIN

By Patrick Holmay & Robert Schilmoeller, Computation Laboratory, St. John's University

"JUMP" provides RSTS users the ability to cross from one account to another without using the "LOGIN" program. It was mainly designed to minimize the frustrations and headaches of having passwords for every privileged account, reduce the number of times one has to look up a password for any account whether it be privileged or non-privileged, and to be able to log into those accounts that may have an "*" for a password.

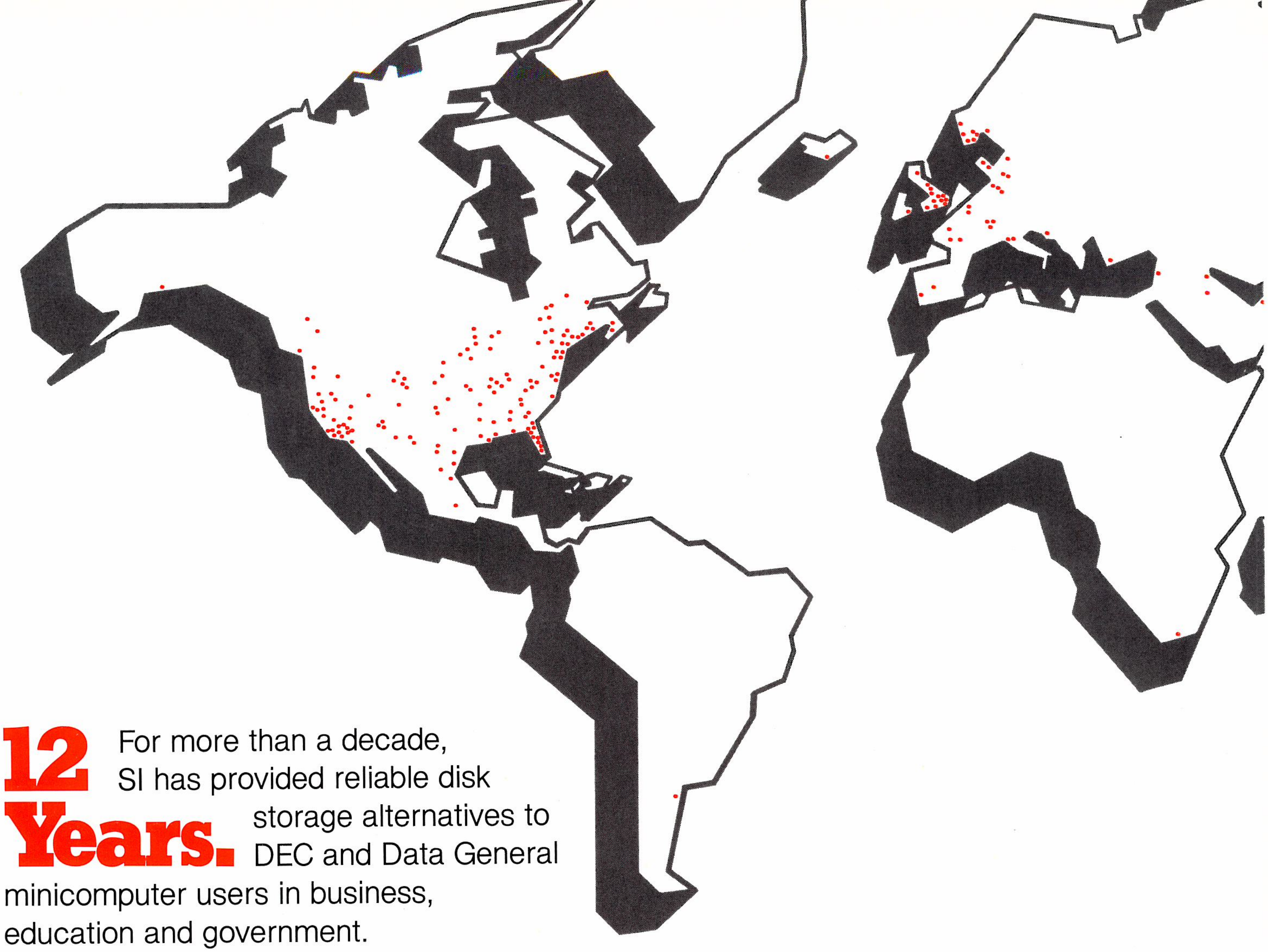
This program was written in BASIC-PLUS for RSTS/E Version 7. It was developed to run as a stand-alone, CCL or Chain Entry program (Line 30000 should be specified for CCL entry; line 30999 has been designated for Chain Entry).

The user can login to a specific account using the following three methods. The first method is by entering the project-programmer number separated by a comma. The second method is by entering the project-programmer number separated by a slash. Finally, the third method is by entering a wildcard for a specific account (the entered wildcard is checked with those present in WILDCARD$ in the program).

Once the user has typed the specific account number he/she would like to jump to, all temporary files created with that user's job number in the original account will be deleted. All system accounting data is updated. If the user types a comma to separate the project-programmer number, those jobs detached under the specified account, if any, will be displayed. If there are any detached jobs, the user will be prompted for the job number to attach to. The total number of users logged into the new account, if any, will also be displayed.

This program has been running smoothly for the last year. It has saved a tremendous amount of time logging into a specific account. Should there be anyone out in RSTS-land who would be interested in obtaining this program, you can contact us. (This software is being provided for nothing, therefore, we do not feel obligated to maintain it.)

```
1       !

        !       J U M P

2       !       PROGRAM  :      JUMP.BAS
        !       VERSION  :      2.0
        !       EDIT     :      0
        !       EDIT DATE:      04-MAY-81

4       !       WRITTEN BY:     HOLMAY/SCHILMOELLER
        !       WRITTEN FOR:    COMPUTATION LABORATORY
        !                       ST. JOHN'S UNIVERSITY
        !                       COLLEGEVILLE, MN

10      EXTEND
                ! EXTENDED BASIC

20      !

                !M O D I F I C A T I O N   H I S T O R Y

                ! VERSION       EDIT DATE       REASON

100     !
        ! 'JUMP' PROVIDES RSTS USERS THE ABILITY TO CROSS FROM
        ! ONE ACCOUNT TO ANOTHER WITHOUT USING 'LOGIN'.  THIS
        ! PROGRAM CAN BE RUN AS A STAND-ALONE, CCL OR CHAIN
        ! ENTRY.  ALL TEMP FILES FOR THAT JOB NUMBER ARE PURGED
        ! BEFORE THE USER IS LOGGED INTO THE SPECIFIED ACCOUNT.
        ! IF THE USER ENTERS A COMMA TO SEPARATE THE PROJECT
        ! PROGRAMMER NUMBER, THE NUMBER OF USERS AND THOSE
        ! JOBS DETACHED UNDER THAT SPECIFIC ACCOUNT WILL BE
        ! DISPLAYED.  THE USE OF A SLASH WILL SUPPRESS ANY OF
        ! MESSAGES MENTIONED ABOVE.  WILDCARD ACCOUNT NUMBERS
        ! CAN BE SPECIFIED (WHICH ARE FOUND IN THE VARIABLE
        ! 'WILDCARD$').

200     !

                !V A R I A B L E   D E F I N I T I O N S

201     !  VARIABLE NAME            USED FOR
        !
        !  ACCOUNT$                 NEW ACCOUNT NUMBER
        !  ATT.JOB%                 JOB NUMBER TO ATTACH TO
        !  BELL$                    TO PROMPT USER OF ANY ERRORS
        !  COMMA%                   PPN SEPARATOR
        !  COMMON$                  CORE COMMON IF CHAIN
        !  CR$                      <CR>
        !  CUR.PROJ%                USER PROJECT NUMBER
        !  E$                       SYSTEM ERROR MESSAGE
        !  FILE%                    FILE DELETION CHANGE VARIABLE
        !  IOB%                     USER I/O BLOCK ADDRESS
        !  JOB%                     USER JOB NUMBER
        !  KB.NUMBER%               USER KB: NUMBER
        !  LOGIN$                   NEW ACCOUNT DATA
        !  M%                       LOGIN CHANGE VARIABLE
        !  MAX.NO.JOBS%             MAXIMUM JOB NUMBER
        !  NULL$                    EMPTY INPUT
        !  PASSWORD$                PASSWORD FOR NEW ACCOUNT
        !  PROG%                    NEW PROGRAMMER NUMBER
        !  PROJ%                    NEW PROJECT NUMBER
        !  RET.LINE%                LINE NO. OF PROGRAM TO RETURN TO
        !  RET.PGM$                 PROGRAM TO RETURN TO IF CHAIN
        !  SLASH%                   PPN SEPARATOR
        !  STRIP%                   VARIABLE TO SETUP ACCOUNT INPUT
        !  USER%                    JOB STATUS CHANGE VARIABLE
        !  WILDCARD$                PRIVELEDGED WILDCARD ACCOUNTS
        !

500     GOSUB 10000
                ! OBTAIN JOB STATUS DATA IF USER EXECUTES
                ! PROGRAM WITHOUT USING A CCL OR CHAIN ENTRY

899     !

                !D I M E N S I O N   S T A T E M E N T S

900             ! UTILITY DIMENSION STATEMENTS

910     DIM USER%(30%), FILE%(30%), M%(30%)
                !>> USER%( ) = JOB STATUS INFORMATION
                !>> FILE%( ) = FILE INFORMATION FOR CURRENT ACCOUNT

999     !

                !I N I T I A L   P R O G R A M   L O G I C

1000    ON ERROR GOTO 19000
                ! SETUP ERROR HANDLING ROUTINE

1010    IF      ENTRY%
        THEN    GOTO 1060
                !>> ENTRY% = CCL OR CHAIN ENTRY PARAMETER

1020    PRINT
        \ PRINT CVT$$(RIGHT(SYS(CHR$(6%)+CHR$(9%)+CHR$(0%)),3%),4%);
                TAB(27%);"Job ";NUM1$(JOB%);
                TAB(35%);"[";NUM1$(CUR.PROJ%);",";NUM1$(CUR.PROG%);"]";
                TAB(44%);"KB";NUM1$(KB.NUMBER%);
                TAB(50%);DATE$(0%);
                TAB(61%);TIME$(0%)
                ! PRINT THE SYSTEM HEADER LINE CONTAINING THE
                ! SYSTEM NAME AND THE LOCAL INSTALLATION NAME

1060    !

                !C O N S T A N T   D E F I N I T I O N S

1070    BELL$ = CHR$(7%)
        \ NULL$ = ""
        \ STRIP% = 2%+4%+32%
                !>> BELL$ = INDICATOR THAT ERROR HAS OCCURRED
                !>> NULL$ = DETERMINES IF INPUT AS BEEN ENTERED
                !>> STRIP% = SETUP INPUT CORRECTLY

1080    WILDCARD$ = " $!%&"
                !>> WILDCARD$ = PRIVELEDGED WILDCARD ACCOUNTS

1199    !

                !P R E L I M I N A R Y   L O G I C

1999    !

                !M A I N   P R O G R A M   L O G I C
```



12 Years. For more than a decade, SI has provided reliable disk storage alternatives to DEC and Data General minicomputer users in business, education and government.

20,000 Installations. SI has given thousands of DEC and DG users just what they wanted. Reliability. Flexibility. Broad choice. Fast delivery. And savings in both money and floor space.

A Worldwide Service Network. SI maintains fully staffed and equipped service centers throughout the U.S. and Europe to ensure prompt response and complete repair capability.

The reliable alternative for over a decade.

System Industries

United States: 1855 Barber Lane, Milpitas, CA 95035, (408) 942-1212, Telex 346459.
Europe: System Industries (Europe), System House, Guildford Road, Woking, Surrey, GU22 7QQ, England, (44) 4862 5077, Telex 859124.
California (714) 851-6289, (213) 822-3161; **Colorado** (303) 986-1591; **Georgia** (404) 955-2252; **Illinois** (312) 948-9330; **Massachusetts** (617) 695-4022; **Michigan** (313) 663-4925;
New Jersey (201) 839-8650; **New York** (516) 482-6082; **New York Metro** (212) 785-9300; **Ohio** (513) 771-0075; **Texas** (713) 497-7224, (214) 386-8776;
Virginia (703) 734-9700; **Washington** (206) 451-8791; **West Germany** (49) 6102 5464; **Sweden** (46) 08-63 16 20

CIRCLE 20 ON READER CARD

Structured disks for all!
Announcing
REACT2.TSK

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

Introductory Price: $200.00
Single CPU License

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

Call For Quick Service

CIRCLE 7 ON READER CARD

SPSS
PDP-11

SPSS® makes data analysis simple for DEC PDP-11 users! Now PDP-11 users can enjoy all the benefits that have made SPSS the world's largest selling Data Analysis System. It's easy to use and learn, thanks to its response to English language commands and comprehensive documentation. It's also sophisticated, giving researchers and business managers alike a full range of capabilities for statistical analysis and report generation.

For full information, call or write SPSS today: Sue Phelan, SPSS, Inc., 444 N. Michigan Avenue, Chicago, IL 60611, 312/329-2400.

SPSS inc.
DATA
ANALYSIS
MADE SIMPLE

© Copyright 1981 SPSS, Inc.

SPSS-11 runs on DEC LSI-11 through PDP-11/70. Compatible with DEC Systems RSTS, RT-11, RSX-11M, IAS/Bell Labs UNIX/S&H Computer Systems TSX.

CIRCLE 123 ON READER CARD

```
2000    IF      ENTRY%
        THEN    2010
                ! IF CCL ENTRY, THEN SKIP ACCOUNT PROMPT

2005    PRINT "Account number";
        \ INPUT LINE ACCOUNT$
        \ ACCOUNT$ = CVT$$(ACCOUNT$,STRIP%)
        \ GOTO 9000     IF      ACCOUNT$ = NULL$
                ! PROMPT USER FOR ACCOUNT # ONLY IF THE
                ! CCL WAS TYPED AND NOTHING MORE.
                ! STRIP GARBAGE FROM ACCOUNT #.
                ! IF THERE IS NOTHING IN THE STRING THEN EXIT.

2010    COMMA% = INSTR(1%,ACCOUNT$,",")
        \ SLASH% = INSTR(1%,ACCOUNT$,"/")
        \ IF    COMMA%  OR  SLASH%
          THEN  2020
          ELSE  PROJ% = 1%
                \ PROG% = INSTR(1%,WILDCARD$,LEFT(ACCOUNT$,1%))
                \ IF    PROG% > 0%
                  THEN  2030
                  ELSE  PRINT "?Can't find file or account"
                        \ GOTO 9000
                ! DETERMINE IF USER HAS TYPED IN AN
                ! ACCOUNT # OR A WILDCARD SYMBOL.

2020    P% = COMMA% + SLASH%
        \ PROJ% = VAL(LEFT(ACCOUNT$,P%-1%))
        \ PROG% = VAL(RIGHT(ACCOUNT$,P%+1%))
                ! IF NOT A WILD CARD ENTRY, THEN STRIP
                ! THE PROJECT-PROGRAMMER NUMBER FROM ACCOUNT$

2030    CHANGE SYS(CHR$(6%)+CHR$(-10%)+"????"+
                RIGHT(NUM1$(100%+(255% AND PEEK(518%))/2%),2%)+
                ".TMP") TO FILE%
        \ FILE%(1%) = 6%
        \ FILE%(2%) = 17%
        \ FILE%(3%), FILE%(4%) = 0%
        \ CHANGE FILE% TO TEMP.CH$
                ! SET UP TO DELETE THE TEMP FILE "TEMPNN.TMP"
                ! (WHERE NN IS THE USER'S JOB NUMBER) FROM THE
                ! USER'S FILE DATA.

2040    CHANGE SYS(TEMP.CH$) TO FILE%
        \ KILL RAD$(FILE%(7%)+SWAP%(FILE%(8%)))+RAD$(FILE%(9%)+
                SWAP%(FILE%(10%)))+"."+RAD$(FILE%(11%)+SWAP%(FILE%(12%)))
        \ GOTO 2040
                ! PLOW THROUGH USER'S "????NN.TMP" FILES, KILLING EACH.

2050    PASSWORD$ = MID(SYS(CHR$(6)+CHR$(14)+CHR$(0)+CHR$(SWAP%(0))+
                CHR$(0)+CHR$(0)+CHR$(PROG%)+CHR$(PROJ%)),9%,4%)
                ! FETCH PASSWORD FOR NEW ACCOUNT

2060    Z$=SYS(CHR$(6%)+CHR$(5%))
                ! LOGOUT USER FROM CURRENT ACCOUNT

2070    LOGIN$ = SYS(CHR$(6%)+CHR$(4%)+CHR$(0%)+CHR$(0%)+
                CHR$(PROG%)+CHR$(PROJ%)+PASSWORD$)
        \ CHANGE LOGIN$ TO M%
        \ GOTO 8000     IF      RET.PGM$ <> NULL$
        \ GOTO 9000     IF      SLASH%
        \ GOSUB 12000   IF      M%(4%) > 0%
        \ GOSUB 11000
        \ GOTO 9000
                ! IF USER WANTS TO RETURN TO PROGRAM...GO
                ! LOGIN USER TO NEW ACCOUNT
                ! CHECK AND SEE IF USER WANTS TO SEE THE NUMBER
                ! OF USERS AND DETACHED JOBS FOR THIS ACCOUNT.
                ! IF NOT, THEN EXIT FROM PROGRAM

8000    CHAIN RET.PGM$ LINE RET.LINE%
                ! CHAIN TO SPECIFIED PROGRAM

9000    !

                ! E N D   O F   T H E   P R O G R A M

9010    Z$ = SYS(CHR$(9%))
        \ GOTO 32767
                ! CLEAR PROGRAM FROM MEMORY
                ! EXIT FROM PROGRAM

9999    !

                ! U S E R   S U B R O U T I N E S

10000   !

                ! O B T A I N   J O B   S T A T U S


10010   JOB% = (PEEK(518%) AND 255%)/2%
        \ IOB% = PEEK(PEEK(520%))
        \ KB.NUMBER% = (SWAP%(PEEK(PEEK(IOB% + 0%) + 2%)) AND 255%)
        \ P.PN% = PEEK(PEEK(PEEK(520%)+8%)+24%)
        \ CUR.PROJ% = SWAP%(P.PN%) AND 255%
        \ CUR.PROG% = P.PN% AND 255%
                ! JOB%          -> CURRENT JOB USER IS LOGGED UNDER
                ! IOB%          -> I/O BLOCK ADDRESS
                ! KB.NUMBER%    -> TERMINAL KEYBOARD # ON CHANNEL #0
                ! P.PN%         -> PROJECT-PROGRAMMER NUMBER OF CURRENT
                !                  USER.  THE PROJECT NUMBER IS STORED
                !                  IN THE VARIABLE 'CUR.PROJ%'.

10020   IF      CUR.PROJ% <> 1%
        THEN    PRINT "?Protection Violation"
                \ GOTO 9000
                ! IF USER IS NOT PRIVELDGED THEN LET THEM KNOW
                ! AND EXIT FROM PROGRAM.

10030   RETURN

11000   !

                ! N U M B E R   O F   U S E R S   L O G G E D
                !          I N T O   A C C O U N T

11010   IF      COMMA%  AND  M%(3%) > 1%
        THEN    PRINT NUM1$(M%(3%)-1%);" other user";
```

```
                \ PRINT "s are";          IF       M%(3%)-1% > 1%
                \ PRINT " is";  IF       M%(3%)-1% = 1%
                \ PRINT " logged in under this account"
                ! PRINT THE NUMBER OF USERS FOR THIS ACCOUNT
                ! EXCLUDING THE ACCOUNT JUST JUMPED TO.

11020   RETURN

12000   !

                ! A T T A C H   T O   A   J O B

12010   PRINT "Job";
        \ PRINT "s";    IF       M%(5%) <> 0%
        \ PRINT " ";NUM1$(M%(INDEX%));
                FOR      INDEX% = 4%
                         WHILE   M%(INDEX%) <> 0%
        \ IF    M%(5%) <> 0%
          THEN  PRINT " are ";
          ELSE  PRINT " is ";
12020   PRINT "detached under this account"
                ! LIST THOSE JOBS THAT ARE DETACHED UNDER
                ! THE NEW ACCOUNT
```

```
12030   PRINT "Job number to attach to";
        \ INPUT ATT.JOB%
        \ RETURN         IF      ATT.JOB% = 0%
                ! ASK FOR WHICH JOB TO ATTACH TO

12040   IF      ATT.JOB% < 1%  OR  ATT.JOB% > ~~MAX.NO.JOBS%~~ 63%
        THEN    PRINT "?Job number out of range"
                \ GOTO 12030
                ! CHECK TO SEE IF THE JOB NUMBER FALLS IN THE
                ! ALLOWABLE RANGE.

12050   INDEX% = 4%
        \ WHILE M%(INDEX%) <> 0%
        \ IF    M%(INDEX%) = ATT.JOB%
          THEN  12070
          ELSE  INDEX% = INDEX% + 1%
                \ NEXT
12060   PRINT "?Job not detached under this account"
        \ GOTO 12030
                ! DETERMINE IF JOB IS ACTUALLY DETACHED UNDER
                ! THIS ACCOUNT
```

. . . continued on page 89

DECWORD/DP
or
How I Tried to Get a Free VT100 Advanced Video Option

By Joe Doyle, 2822 Truman Drive, Hatfield, PA 19440

It started the day the cardboard keyboard arrived. You know; you all received one. Certainly not your typical four color glossy folder-in-an-envelope promo. Rather, a full size, full color, cardboard keyboard announcing DECWORD, a new DEC software product. It seemed that DEC had made an arrangement with Data Processing Design to market a version of their product, WORD-11. By simply mailing in a postcard, you would receive a version of the Computer Based Training Program, a VT100 Advanced Video Option, and a genuine, authentic, brand new VT100 keyboard. Now I have a confession to make. First, we already have WORD-11 on our system. We have no need for DECWORD. Second, we have a vast mixture of terminal types already set up with WORD-11, DPD supplied keycaps or stickers. Third, all our VT100s already have the advanced video option. But, who can resist a freebee? After all, Word-11 does not have a computer aided instruction system. Also, who wouldn't like to have a spare keyboard available. (An industrial site I worked at, always had keyboard that were destroyed by dirt and dust). Finally, we could buy another VT100 (which we needed) without the AVO, add the free part, and save some money. We sent in our postage paid postcard that morning.

The CBT tape, keyboard, AVO as well as associated pieces of paper arrived in a couple of weeks. After unpacking, the first thing to do was test the keyboard. Truthfully, the keyboards are nice. Colored keycaps are certainly much more impressive than the plastic translucent stickies provided by DPD. I mean, it is classy looking! I unplugged my old keyboard. Plugged in the new one. Viola! It worked . . . except for the line feed and the back slash. The 'inside' story is that the Field Service Department didn't want to be bothered switching keycaps on all those VT100s out there; so they convinced the company to simply (and more economically??) provide free, new, cap colored keyboard. In spite of it all, somebody has managed to put two keys on wrong. These can be easily pried up and switched. (Who needs Field Service for keycaps?) After verifying that the keyboard worked, I compared the DECWORD keyboard layout to the WORD-11 keyboard layout. Just as I suspected — different. OK, perhaps it was somewhat compatible with the EDT layout? I remember that about a year ago, the EDT people at DEC were very interested in providing keycaps. No, only the gold key was EDT compatible. I didn't even consider TECO. I also noted that the DEC keyboard did not have as many functions as the WORD-11 setup.

The next step was to load the training tape. No problem, PIP!!! The direction sheet is clear. No CCL's to eat up those small buffers. One logical. I ran the product on my VT100 with the original AVO. It worked excellently. It also worked on a DT-80 and a VT52. The Hazeltine 1552 had some functions, but not all. I think it will work to help train out new personnel in WORD-11. Unfortunately, the program does change the terminal characteristics without restoring them. Namely, CTRL/R and ESC SEQ. It appears DEC has accepted this type of inconsistent terminal hocus-pocus as a standard.

Now for the Advanced Video Board. I didn't need it, but I wanted to try it. The included installation book is excellent, with the exception of the page describing the AVO switches (page 14). First of all, there are two sets of switches. The manual describes how to set one set but not the other. The manual describes setting the "E19 switch pack". There was no E19 switch pack on my board, only E18. The manual describes the switches as being ON or OFF. Mine were OPEN. (We know from DZ experience that OPEN can mean both OFF and ON). My switch pack was upside down compared to the switch pack in the picture or were the numbers on backward? No matter, I tried what I thought was the correct setting. And another. And another . . . Then I cut my hand with the screwdriver.

After many more attempts, I had experienced the ultimate high in visual display fantasies. However, no advanced video. The manual (EK-100CK-IN-001) clearly states,

IF YOU NEED HELP:

* Call Digital Field Service

I did. The call-logging person had never heard of DECWORD or the promo package. However, DEC would be glad to send over one of those Field Service Technicians at $76.20 per hour. When I re-explained the problem, she gave me the phone number of the terminal group. I called. After three conversations, I was passed to a terminal engineer. So far, no one had ever heard of DECWORD. The engineer sounded friendly and wanted to help. He did suggest I call Field Service. He also admitted that he had heard of a new AVO board with switches, but had never met one in person. He assured me that E18 meant E19, and that OPEN meant OFF except on DZ's. He took the manual number and told me if he could find some additional information he would call back. I tried his ideas. (I had tried them all before — honest.) The board still failed. All that was left was the DEC SALES OFFICE. I dialed the number on the cover letter. The employee who answered the phone had never heard of DECWORD . . .♥

IMPRS

A Productivity Relational Data Base Language

By Jacob F. Ruf, Ruf Corporation, Olathe, Kansas

ABSTRACT

IMPRS (Information Management Processing Reporting System) is a software language product of Ruf Corporation.

IMPRS provides the capability of developing complete application software from five to ten times faster than would be possible using conventional programming approaches. This productivity is accomplished through the modular, operation-oriented structure of IMPRS and its powerful relational data base language. The IMPRS user is commanding his file management processing and reporting in terms of "What needs to be accomplished?" as opposed to the detail level of "How?" required by conventional data base managers.

The flexibility and efficiency of IMPRS with its imbedded high-speed sort, versatile record subset selection, and interactive report generation make it the most powerful relational data base programming language currently operating on hardware in this size and price range. The execution speed of IMPRS programs is two to three times faster than that of COBOL or BASIC-PLUS.

IMPRS has been in constant use on Ruf Corporation public time-sharing DEC 11 systems and many other computer systems in the Kansas City area since March, 1977. Application packages (over 2,500 programs) are available supporting practically all information management fields. The basic IMPRS package consists of over 500 Fortran subroutines linked into over thirty (.SAV) programs complete with documentation.

IMPRS is currently supported on DEC 11 systems running under RT11, RSX, and RSTS/E operating systems.

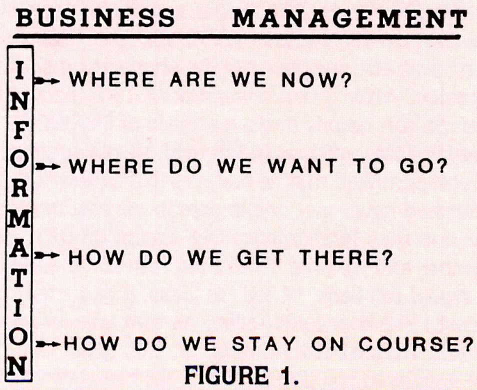
INTRODUCTION

A systems analyst must first understand the philosophy under which a business operates before he can successfully develop an information system to serve that business. This is necessary because the information system which he develops must operate under the same philosophy.

Successful businessmen are goal-directed. Their decision making is performed with the use of rational processes. They measure the attainment of their goals through quantifiable evaluations.

To effectively place this philosophy into operation in the management of our business we continually ask the questions outlined in Figure 1.

In order to answer those questions, we are continually drawing upon information. With the advent of the computer, the information system has become more structured in serving management's needs. Today's information systems consist of people, data, hardware, software, and education or procedures.



BUSINESS MANAGEMENT

INFORMATION

WHERE ARE WE NOW?

WHERE DO WE WANT TO GO?

HOW DO WE GET THERE?

HOW DO WE STAY ON COURSE?

FIGURE 1.

The value of the computerized information system is illustrated in Figure 2. The purpose of an information system is to (1) collect and manage data, (2) process data into information and (3) to disseminate the information to people for their use in answering the questions of Figure 1 (above).

FUNCTIONAL USE OF DATA

INFORMATION DELIVERY SYSTEM

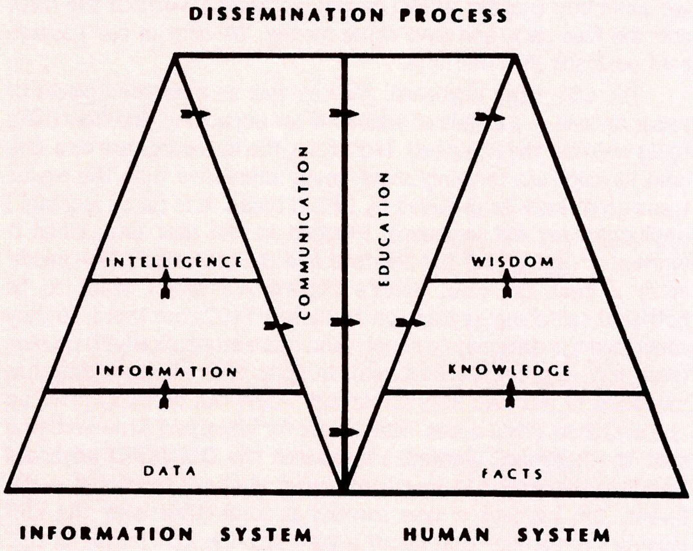
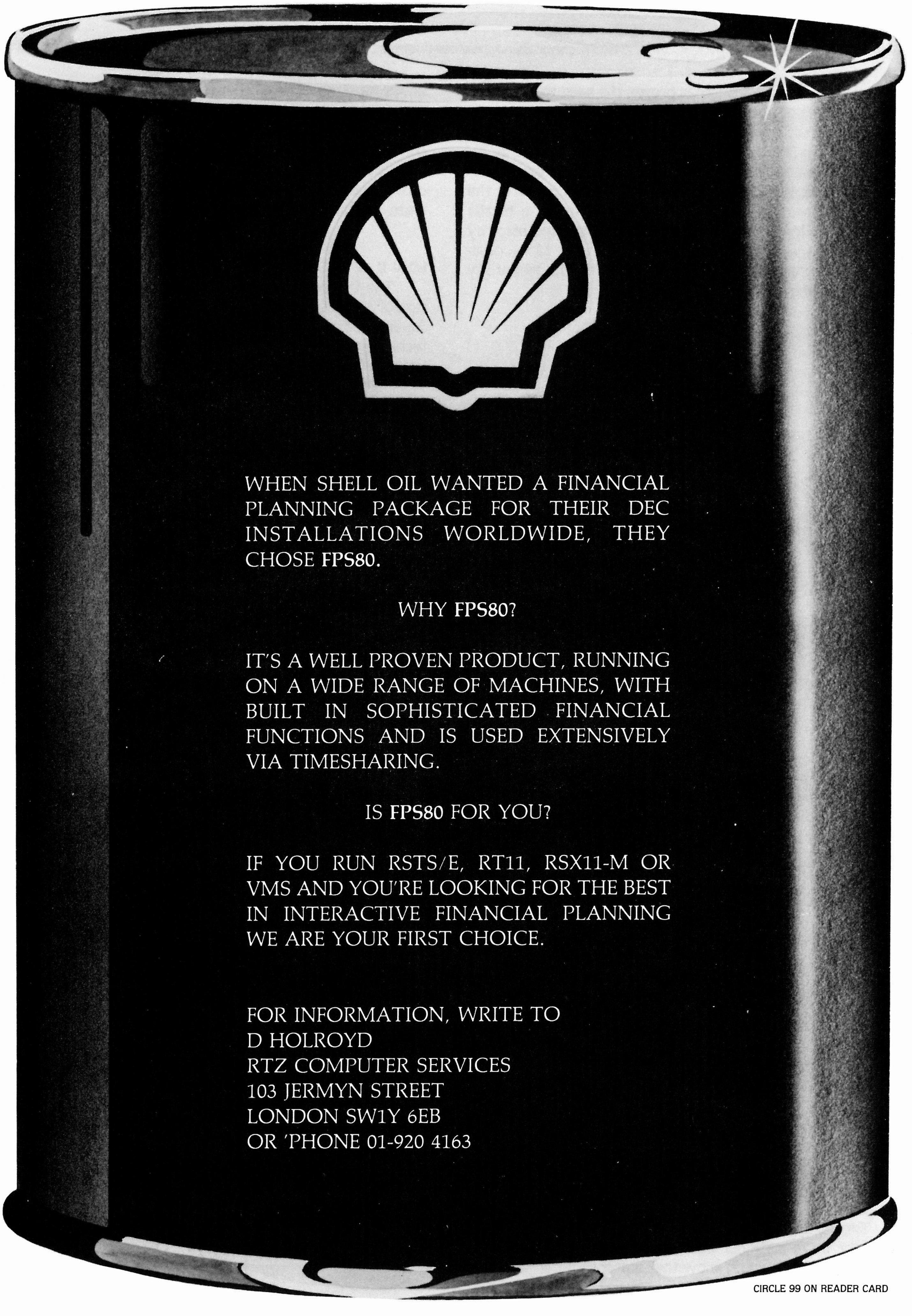


FIGURE 2.

DATA MANAGEMENT — THE KEY

The one component of the information system that "flows" and involves all the other components is data. Suc-



WHEN SHELL OIL WANTED A FINANCIAL PLANNING PACKAGE FOR THEIR DEC INSTALLATIONS WORLDWIDE, THEY CHOSE **FPS80.**

WHY **FPS80**?

IT'S A WELL PROVEN PRODUCT, RUNNING ON A WIDE RANGE OF MACHINES, WITH BUILT IN SOPHISTICATED FINANCIAL FUNCTIONS AND IS USED EXTENSIVELY VIA TIMESHARING.

IS **FPS80** FOR YOU?

IF YOU RUN RSTS/E, RT11, RSX11-M OR VMS AND YOU'RE LOOKING FOR THE BEST IN INTERACTIVE FINANCIAL PLANNING WE ARE YOUR FIRST CHOICE.

FOR INFORMATION, WRITE TO
D HOLROYD
RTZ COMPUTER SERVICES
103 JERMYN STREET
LONDON SW1Y 6EB
OR 'PHONE 01-920 4163

CIRCLE 99 ON READER CARD

cessful system engineering has proven the need to single out the categories of data and the functional use of this data in the operation of the business. See Figure 2A for an example. James Martin (1) has been referred to as the "computer industry's most widely-read author and best-attended lecturer". He states that "by the end of the 1970's, much of the computing in industry and government will relate to the data bases which have been painfully constructed piece by piece and management effectiveness will relate to the quality of their organization's data sources and the versatility with which they can be used".

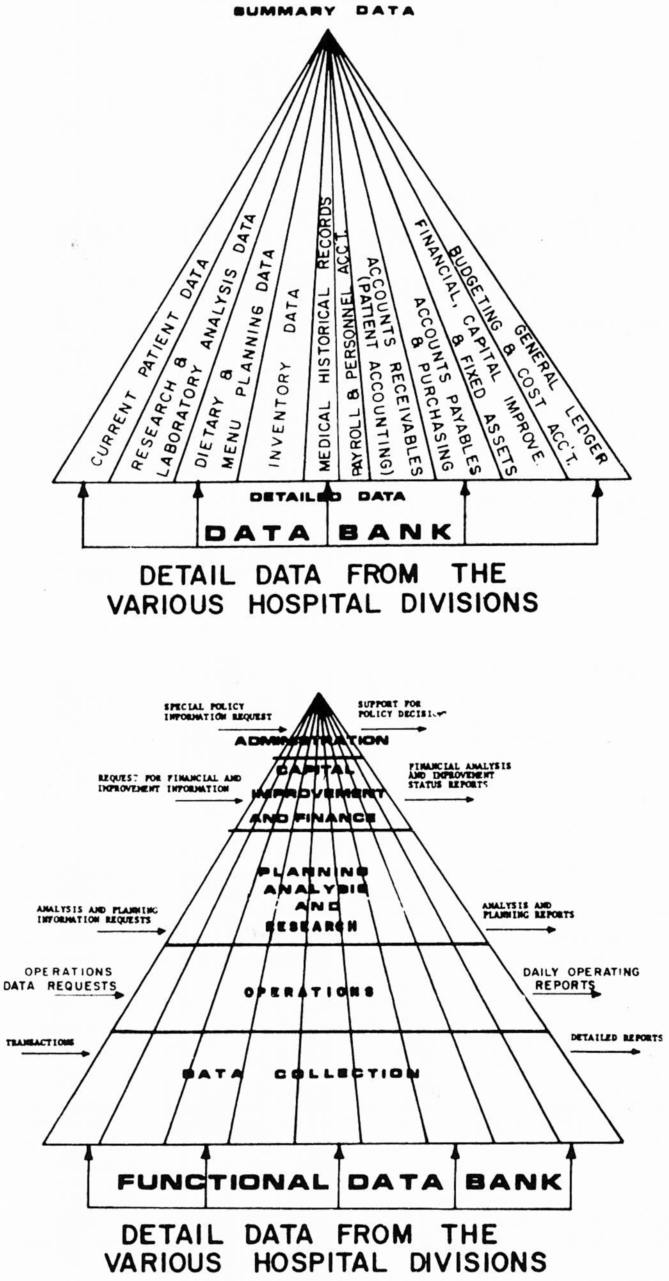

SUMMARY DATA
CURRENT PATIENT DATA
RESEARCH & LABORATORY ANALYSIS DATA
DIETARY & MENU PLANNING DATA
INVENTORY DATA
MEDICAL HISTORICAL RECORDS
PAYROLL & PERSONNEL ACC'T.
ACCOUNTS RECEIVABLES (PATIENT ACCOUNTING)
ACCOUNTS PAYABLES & PURCHASING
FINANCIAL, CAPITAL IMPROVE. & FIXED ASSETS
BUDGETING & COST ACC'T.
GENERAL LEDGER
DETAILED DATA
DATA BANK
DETAIL DATA FROM THE VARIOUS HOSPITAL DIVISIONS
SPECIAL POLICY INFORMATION REQUEST
SUPPORT FOR POLICY DECISIONS
ADMINISTRATION
REQUEST FOR FINANCIAL AND IMPROVEMENT INFORMATION
CAPITAL IMPROVEMENT AND FINANCE
FINANCIAL ANALYSIS AND IMPROVEMENT STATUS REPORTS
ANALYSIS AND PLANNING INFORMATION REQUESTS
PLANNING ANALYSIS AND RESEARCH
ANALYSIS AND PLANNING REPORTS
OPERATIONS DATA REQUESTS
OPERATIONS
DAILY OPERATING REPORTS
TRANSACTIONS
DATA COLLECTION
DETAILED REPORTS
FUNCTIONAL DATA BANK
DETAIL DATA FROM THE VARIOUS HOSPITAL DIVISIONS

FIGURE 2A.

SOARING SOFTWARE COST

Throughout the last decade, program development productivity on a national basis has not increased. Recent studies by Ferrentino (2) show that a programmer produces about ten lines of program source code per day, as was the standard ten years ago.

Today, the cost of software equals 70% of the total cost of operating a computerized information system. Hardware cost is equivalent to only 30% of that total. By the end of this decade it is estimated that the ratio will be 90% software compared to 10% hardware.

LOW U. S. PRODUCTIVITY

The United States has an annual productivity growth rate of about zero compared to more than 11 percent for the Japanese. The time lag between the development of a new technology and putting it to use averages seven years in the United States. In contrast, the technology transfer time for West Germany is five years and that for Japan is only three years.

It is a fact that over 50% of all American technology is originated in small businesses. However, large financial resources are required to adequately implement technology. Due to the lack of capital in small business and the status quo approaches of large business, the United States has fallen drastically behind in our productivity. The low productivity growth is a contributing factor to the increasing inflation in this country.

PAST SOFTWARE QUANTUM LEAPS

Computer language productivity has developed in a series of leaps throughout the history of the computer. Figure 3 illustrates these leaps in productivity. Language technology advances have generally been accompanied by a seven to one increase in productivity. It usually takes seven times longer to develop a program in assembly language as it does in COBOL or Fortran.

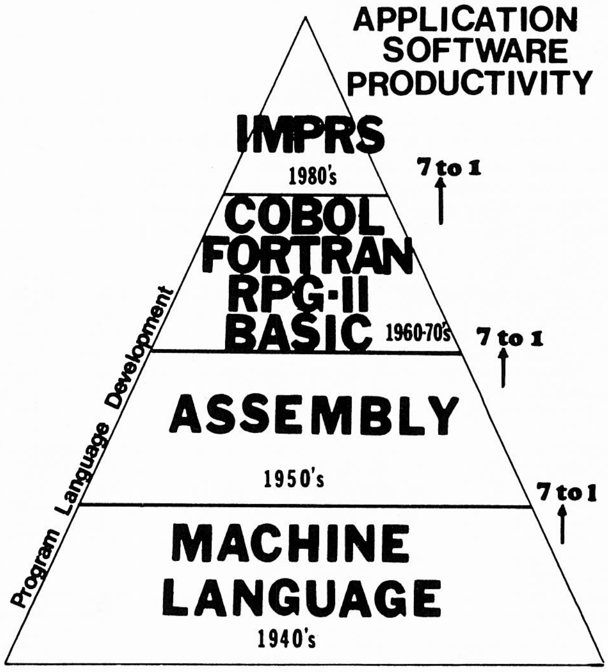
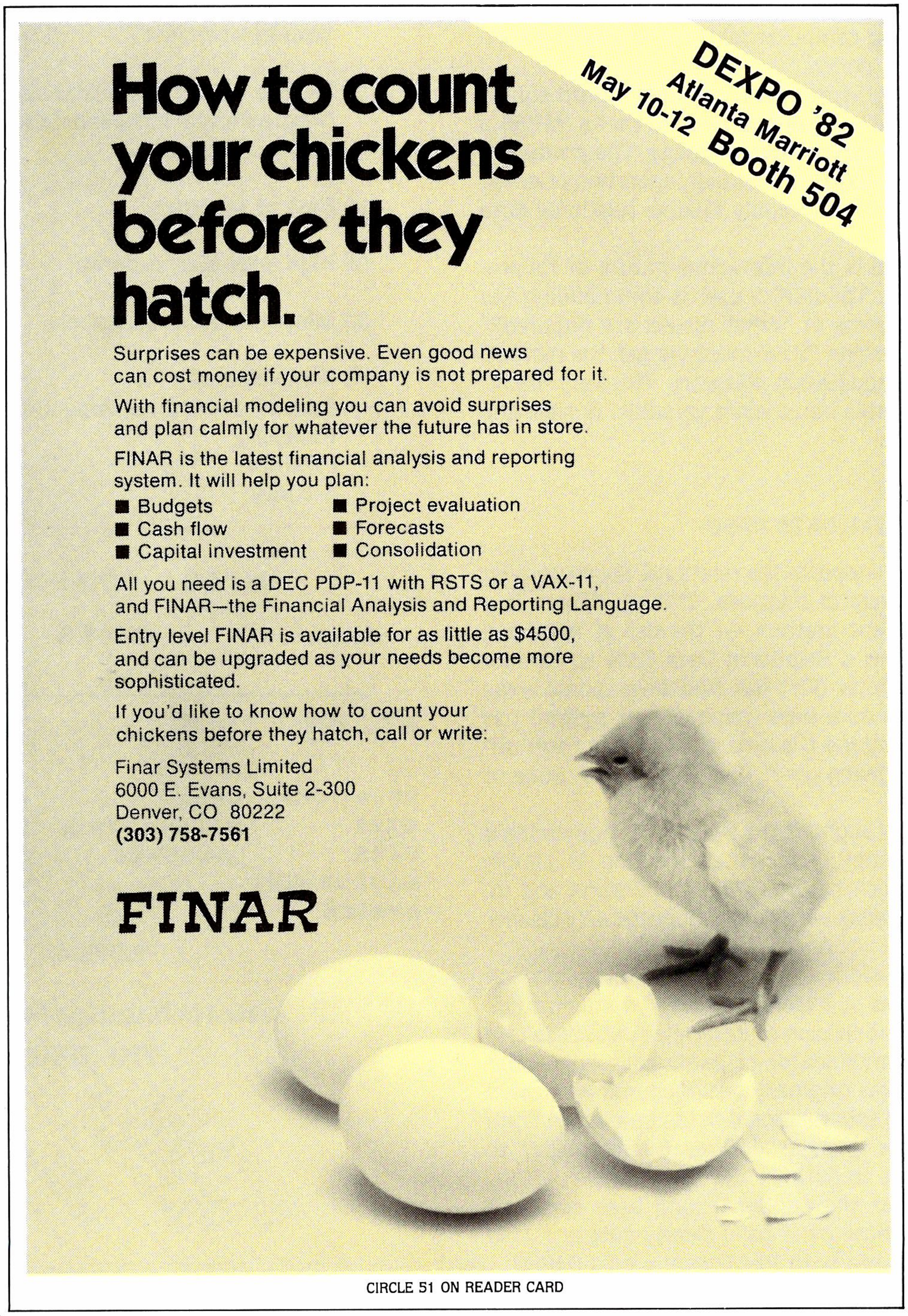


FIGURE 3.

According to James Martin (5), "Until recently, 90% of commercial programming was done in COBOL. We cannot go on using COBOL", he said, "because COBOL requires too many scarce programming resources. PL/1 faces the same fate." Martin believes we need a fourth generation of programming languages to begin to solve the "colossal problem" of programming productivity. Recent language developments do not solve this problem. It appears that we are overdue for a quantum leap in software productivity.


DEXPO '82
Atlanta Marriott
May 10-12 Booth 504

How to count your chickens before they hatch.

Surprises can be expensive. Even good news can cost money if your company is not prepared for it.

With financial modeling you can avoid surprises and plan calmly for whatever the future has in store.

FINAR is the latest financial analysis and reporting system. It will help you plan:

- Budgets
- Cash flow
- Capital investment
- Project evaluation
- Forecasts
- Consolidation

All you need is a DEC PDP-11 with RSTS or a VAX-11, and FINAR—the Financial Analysis and Reporting Language.

Entry level FINAR is available for as little as $4500, and can be upgraded as your needs become more sophisticated.

If you'd like to know how to count your chickens before they hatch, call or write:

Finar Systems Limited
6000 E. Evans, Suite 2-300
Denver, CO 80222
(303) 758-7561

FINAR

CIRCLE 51 ON READER CARD

PAST "BRIGHT" APPROACHES

During the 1970's, formalized software was developed to act as the "file secretary" and to augment the host language. This software supported the programmer with the retrieval and filing of data upon his programmed request. These systems were called Data Base Management Systems (DBMS).

With the stress on centralized control of data, it appeared that the DBMS of the 70's offered the best solution to information processing. But, because of the complexities and lack of flexibility of the hierarchical tree and plex approaches used in the early DBMS, disappointments and failures have been widespread. As with many innovations, users sometimes got wrapped up in the methodology of the system and lost track of the desired goal. These "LBJ's" of the New Frontier in the 70's had taken over. These "Lightning Bug Jocks" were so concerned about "How it works" that they lost track of providing what was needed. They were bright boys, but it was all in their rear end. I am reminded of a saying of my Chemical Engineer Thesis Professor at the University of Kansas, the late Dr. Fred Kurata. He was teaching the thermodynamic law on work where work is equal to the product of Pressure times Volume, i.e. W = PV. He made the observation to the class that "We can have all the P in the world, but if we don't have any V, we will not have any work". As we look back on the history of the DBMS, in many cases it seems that the mountain toiled and brought forth the mouse.

"The structured techniques of the 1970's are completely inadequate," says James Martin (6). "We need new types of methodology, with end user involvement, user-driven computing." It appears that we have overdone the methodology of the hierarchical DBMS. Data processing organizational goals have become increasingly unrelated to the goals and objectives of the institution they serve. Those data processing people who have not experienced the problems of the early prototype DBMS should feel fortunate that there are much more efficient approaches available today.

IMPRS PRODUCTIVITY

Information Management Processing Reporting System (IMPRS) is a relational data base software product developed by Ruf Corporation which gives that quantum leap in software productivity. Since the development of IMPRS in 1976, we have experienced and documented software development productivity factors of 5 to 30 times faster with IMPRS. In a recent comparison of 54 payroll and accounts payable programs, we were able to develop at the rate of 300 equivalent COBOL lines per day compared to the national average of 10 per day. This is 30 times faster with IMPRS.

We have developed complete business systems in COBOL and BASIC-PLUS and executed them for several years on our time-sharing systems. We then redeveloped the systems in IMPRS. We are not only seeing the quantum increase in program development performance with IMPRS, but are experiencing execution speeds of two to three times faster than the equivalent COBOL or BASIC-PLUS programs. These observations were based on reliable studies with the use of our time-sharing computer resource accounting software over a four year period.

The imbedded high-speed sort, versatile record subset selection, and interactive report generation make IMPRS a powerful data base programming language. The productivity is accomplished through the modular, operation-oriented structure of IMPRS, with its highly flexible relational data base.

The key to IMPRS is the interactive nature of its processing and reporting. The IMPRS user is commanding the file management in terms of "What results are required?" as opposed to the question "How must we get the result?" used in conventional application software. This very important difference translates into a more versatile, flexible and faster system.

IMPRS — A RELATIONAL DATA BASE

James Martin (1) supports the relational approach over the hard-linked hierarchical methods, stating, "There is a simple and more elegant method —- the use of relational data bases." He defines a Relational Data Base as "a data base made up of relations. (Flat file, two-dimensional array of data elements). Its data base management system has the capability to recombine the data elements to form different relations thus giving great flexibility in the usage of data".

He continues, "Throughout the history of engineering a principle seems to emerge: great engineering is simple engineering. Ideas which become too cumbersome and inflexible tend to be replaced with newer, conceptually cleaner ideas."

It is possible to avoid the tangled webs that build up in tree and plex structures by a technique called normalization. Normalization is the subdivision of a complex data base into its lowest common denominator of relationally linked flat files. This technique was originally designed and advocated by E. F. Codd (3). Codd sets relational data bases apart from hierarchical and other popular designs by its ability to automatically navigate to data.

The advantages of the relational data base approach outlined in Table 1 below have been clearly substantiated over five years experience with Ruf Corporation's IMPRS.

IMPRS COMPONENTS

Figure 4 portrays the more visible components of IMPRS, which are Relational Data Base Management Systems (RDBMS), Data Manipulation Language (DML), Query Language Interface (QLI), and Utilities.

Figure 5 more accurately illustrates the components of IMPRS and their interrelation.

TABLE 1

Relational Data Base Advantages

(1) Simple and clear
Supports structured modular approaches

(2) Transparent to change
Data independent

(3) Flexible access and relatability
Geographically independent links
Logical linkages

(4) Ease of reporting

(5) High level DML support

(6) Minimum data redundance

(7) Completeness
Ease of use and implementation
Flexibility
Precision
Security

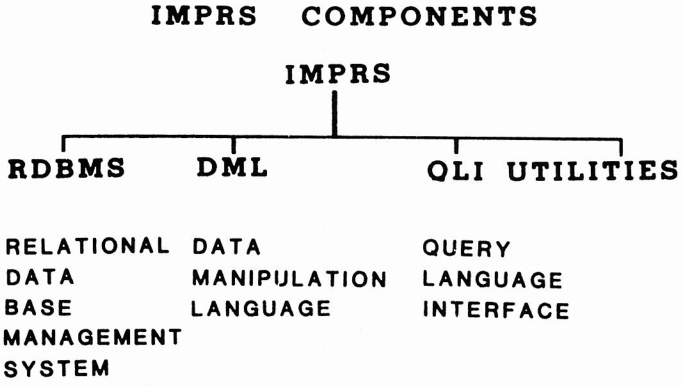


FIGURE 4.

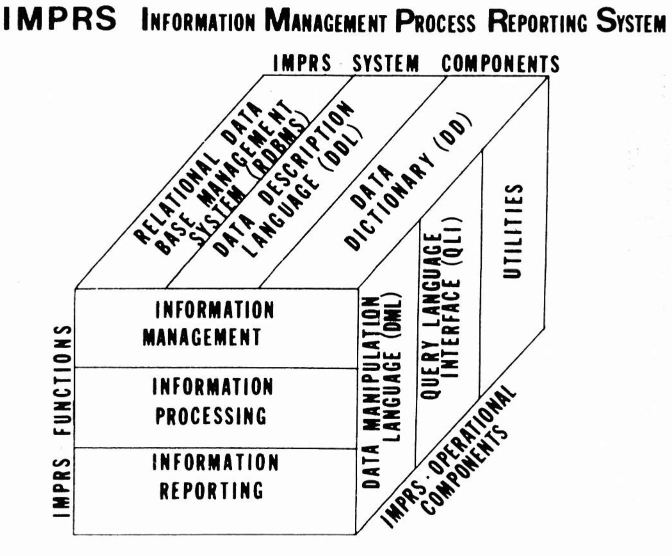

IMPRS INFORMATION MANAGEMENT PROCESS REPORTING SYSTEM
IMPRS SYSTEM COMPONENTS
RELATIONAL DATA BASE MANAGEMENT SYSTEM (RDBMS)
DATA DESCRIPTION LANGUAGE (DDL)
DATA DICTIONARY (DD)
UTILITIES
QUERY LANGUAGE INTERFACE (QLI)
DATA MANIPULATION LANGUAGE (DML)
INFORMATION MANAGEMENT
INFORMATION PROCESSING
INFORMATION REPORTING
IMPRS FUNCTIONS
IMPRS: OPERATIONAL COMPONENTS

FIGURE 5

Don't just "get by" . . .

Call MACRO MAN

for superior performance
• RSTS Internals •
• Custom Macro Programming •
• RSTS or 11M •

LEARN MACRO FROM THE MASTER!
One week course in Philadelphia — call for details.

MACRO UTILITY LIBRARY
Available now!
Call for information.

Soon —
XBUF Display.
XBUF contents revealed!

Also Available Now —
VAX MACRO Consulting

Bob 'Macro Man' Meyer
9 Lockwood Avenue, Fieldsboro, NJ 08505
609—298-9127

CIRCLE 84 ON READER CARD

IMPRS

Information • Management • Processing • Reporting • System

A PRODUCTIVITY RELATIONAL DATA BASE LANGUAGE
DATA MANIPULATION/QUERY LANGUAGE

IMPRS Reduces system development and programming time for application systems.

IMPRS Has more features and runs more efficiently than any other Data Base software.

IMPRS Is available for the following DEC operating systems: RSTS/E • RT11 • RSX

For price quotes and information contact:
(913) 782-8544
1533 East Spruce Olathe, Kansas 66061
"We think smarter and our software works harder for you."

RUF

CIRCLE 109 ON READER CARD

The RDBMS, Data Description Language (DDL), and Data Dictionary (DD) serve as the system level components and are imbedded throughout the operational components which are the Data Manipulation Language (DML), Query Language Interface (QLI) and Utilities. The heart of the IMPRS operational components is the DML and QLI.

With the DML we have written a complete interactive order entry program in four hours. This program built the Order Master and Item files, checked on customer credit and part availability, relieved inventory and priced the order. This was completed with the use of 150 DML program steps.

The QLI provides on-the-spot new management reporting, utilizing menu directed approaches, guiding the novice step by step to the desired reports. Utilizing the Boolean record selection, the user has almost unlimited subset selectivity in his reporting.

IMPRS FEATURES

IMPRS is written in a high-level ANSI language - Fortran IV - and consists of over 500 linked subroutines . . . complete with documentation. We have found Fortran a most appropriate language for developing systems of this nature. IMPRS is a portable system, providing the capability to operate on a multitude of different computers as it rigidly adheres to ANSI programming standards.

IMPRS has been in constant use on Ruf Corporation's public time-sharing systems, and other systems, since 1976. It has proven to be an effective system in actual information management situations ranging from private business to medical research labs; education to manufacturing information systems.

IMPRS is currently supported on Digital Equipment Corporation PDP 11 systems running under RT11, RSX, and RSTS/E operating systems.

The main features of IMPRS are listed in Table 2 below.

TABLE 2. IMPRS Features

Information Management (RDBMS)
- Full Relational Linkage (Interactive/Batch)
- Interactive or Batch DB Management
- File Maintenance Utilities
- Full Cursor Control
- High-Level Structured DML

Information Processing (DML)
- Interactive Multi-File Processing
- Multi-User Simultaneous File Access
- Inter-Job Communication
- Imbedded Sorts
- Structured Data Processing Language

Information Reporting (QLI)
- Interactive Multi-File Report Generator with Boolean Record Selection
- Full Report Writer & Report Generation
- Multi-Contingency Frequency Reporting
- Data Quality Consistency Reporting
- High-Level Structured Reporting Language

Information System (RDBMS)
- High-Level Structured Programming Language
- Security - System to Item Levels
- Multiple Key Access to Files
- Variable Length Records and Fields
- Interfaces to All Languages
- Data Dictionary
- Full Interactive Programming Support
- Automatic System Documentation
- Automatic System Flowcharting
- Full Debugging Aids
- Data Base Disk Usage Management Aids
- Interface to Computer Utilization Accounting
- Interface to High-Level Simulation Language
- Full Cross Reference Programming Aid

IMPRS EXAMPLE

Figure 6 illustrates a daily work-in-process and labor performance system which was developed in a four hour period using IMPRS. A similar system developed previously using COBOL took two man-months. The IMPRS system consisted of a DML program (DAYWK1.TRN), utilizing three files (EMP, STD, and DAYWK), and a QLI interactively reporting labor performance. The DML program interactively received daily labor statistics from the keyboard and extended them using the employee master and job standard files. It then calculated the labor performance and wrote the record into the DAYWK.DAT file for future reporting.

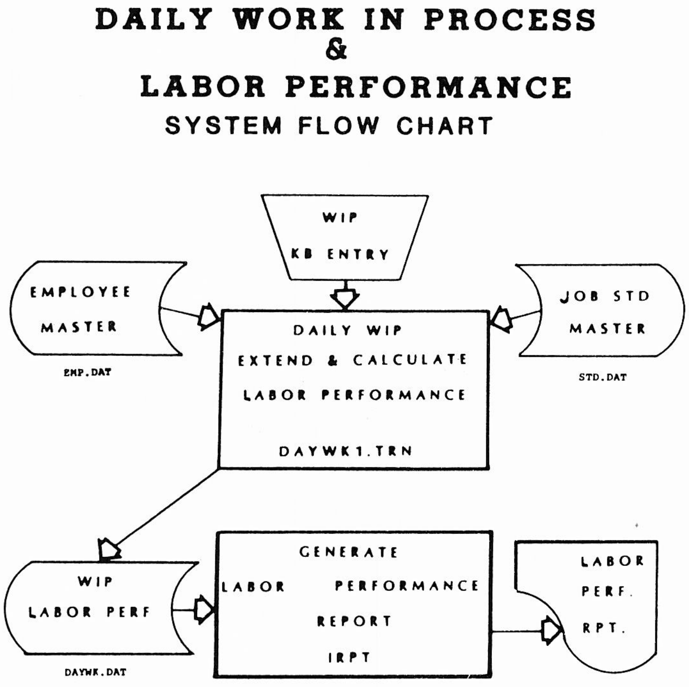


FIGURE 6.

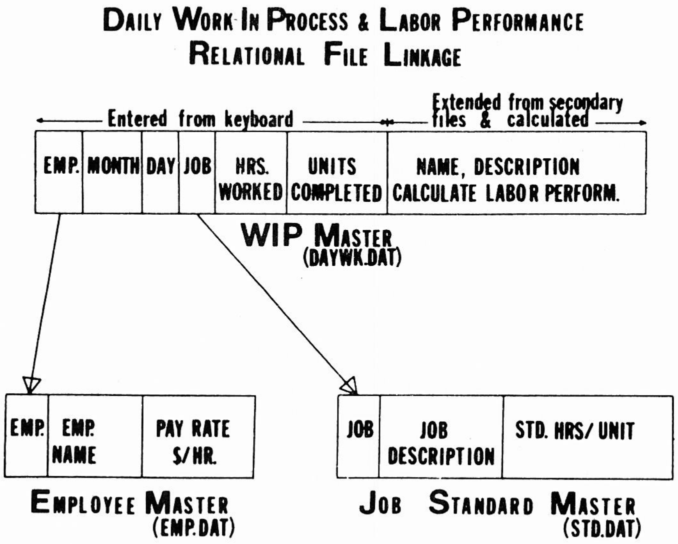
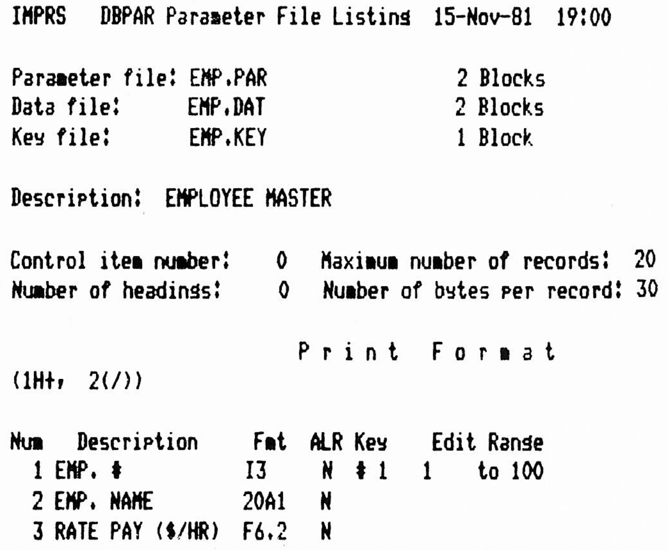


FIGURE 7.

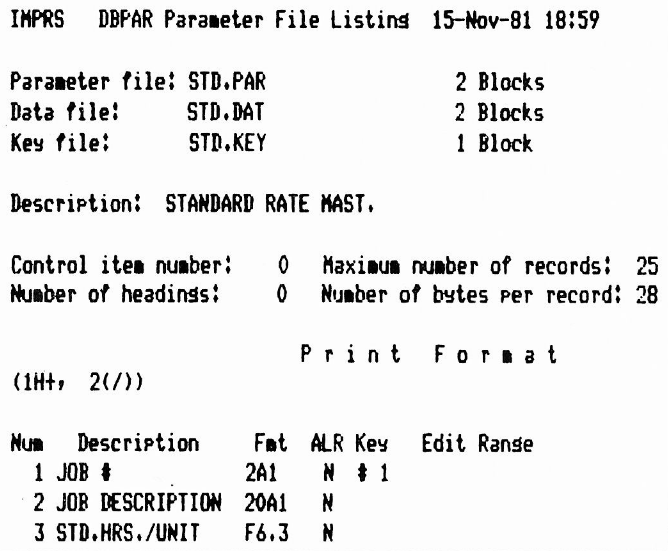
The relational linkage of the employee master and job standard master files to the WIP master is demonstrated in Figure 7. The linkage is performed without the use of hard disk pointers.

TABLE 3.



```
IMPRS   DBPAR Parameter File Listing  15-Nov-81  19:00

Parameter file: EMP.PAR                      2 Blocks
Data file:      EMP.DAT                      2 Blocks
Key file:       EMP.KEY                      1 Block

Description:  EMPLOYEE MASTER

Control item number:    0   Maximum number of records:  20
Number of headings:     0   Number of bytes per record: 30

                           P r i n t   F o r m a t
(1H+,  2(/))

Num   Description       Fmt  ALR Key    Edit Range
  1 EMP. #              I3    N  # 1   1    to 100
  2 EMP. NAME          20A1   N
  3 RATE PAY ($/HR)    F6.2   N
```

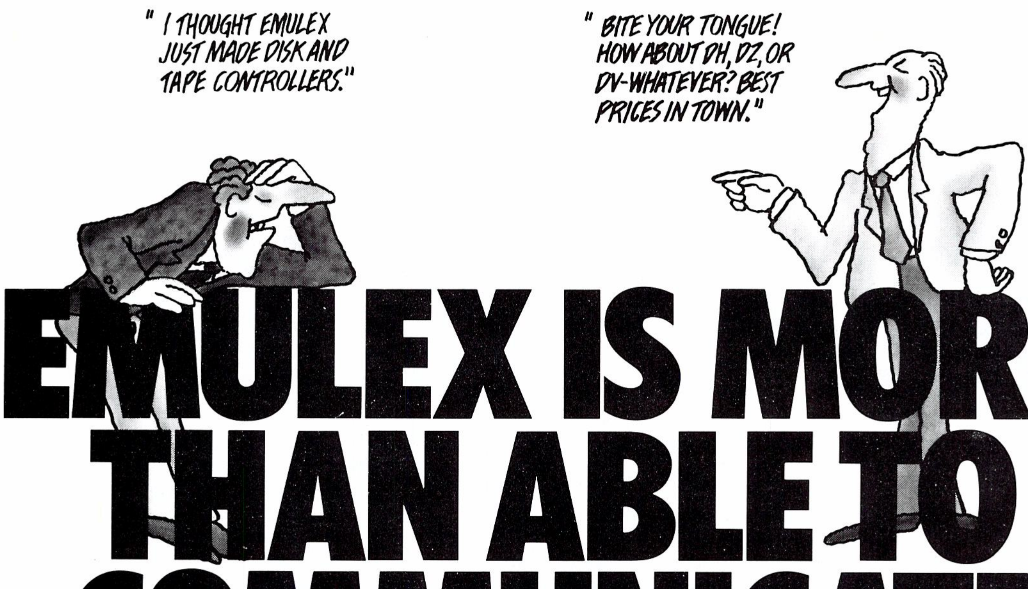
TABLE 4.



```
IMPRS   DBPAR Parameter File Listing  15-Nov-81 18:59

Parameter file: STD.PAR                      2 Blocks
Data file:      STD.DAT                      2 Blocks
Key file:       STD.KEY                      1 Block

Description:  STANDARD RATE MAST.

Control item number:    0   Maximum number of records:  25
Number of headings:     0   Number of bytes per record: 28

                           P r i n t   F o r m a t
(1H+,  2(/))

Num   Description       Fmt  ALR Key    Edit Range
  1 JOB #              2A1    N  # 1
  2 JOB DESCRIPTION    20A1   N
  3 STD.HRS./UNIT      F6.3   N
```

The listing of the data description language (DDL) for the three files is included as Tables 3, 4, and 5. These tables contain all the data attributes that the data base required for this system.

The listing of the DML program (Table 6) consists of fourteen commands and six calculations. The command structure consists of a sequence name, alpha command, and four parameters. The parameters make reference to file numbers as described in the beginning of the DML, item numbers described in the DDL (Tables 3-5), DML sequence names, and other required attributes. The calculations are directed in terms of file numbers, item numbers, and the standard arithmetic operators (+,-,*,/,↑). The function of each command is listed in the command description column.


" I THOUGHT EMULEX JUST MADE DISK AND TAPE CONTROLLERS."
" BITE YOUR TONGUE! HOW ABOUT DH, DZ, OR DV-WHATEVER? BEST PRICES IN TOWN."

EMULEX IS MORE THAN ABLE TO COMMUNICATE WITH DEC.

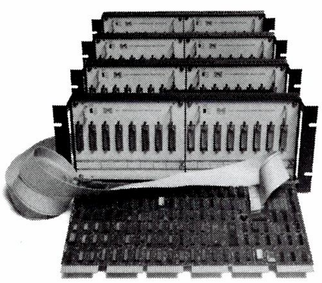
We're also able to save you plenty: For instance, you get DH11 performance for a DZ11 price. Four new space-saving single-board communications multiplexers. And an increase in VAX-11 terminal handling capacity by up to 50%. Maintained nationwide by Control Data.

Microprocessor-based architecture and common hardware deliver faster, more flexible line-handling. Self-test on power-up. Full software transparency. And Emulex reliability standards.

Communicate with Emulex now. Write or call Emulex Corp., 2001 Deere Ave., Santa Ana, CA 92705; (714) 557-7580, TWX 910-595-2521. Outside Calif.: (800) 854-7112.

For immediate off-the-shelf delivery, call our national distributor: First Computer Corporation, 645 Blackhawk Dr., Westmont, IL 60559; (312) 920-1050. In Europe: Emulex Corp., 10th floor, Cory House, The Ring, Bracknell, Berkshire, England. Telephone: 0344-84234; Telex 851-849781.

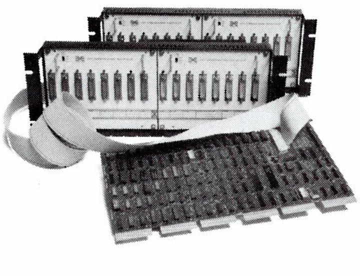
CS11/H (PDP-11) $7560 for 48 lines*
CS11/U (VAX-11) $7884 for 48 lines*



Up to 64 DH11 channels from one board.

DH11-compatible MUX lets you mix RS-232 & current loop interfaces in 8-line groups. Built-in DM11-compatible modem control. DMA output eliminates host interrupts. Self-test capabilities. Transparent to PDP-11 software. Emulex' own software on VAX.

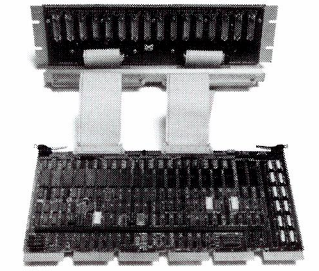
CS11/V $4464 for 16 lines*



Higher DV11 performance, lower price.

DV11-compatible multiplexer. Mixes 8-lines synchronous & asynchronous on PDP-11s. Ideal for Bisync & DECNET. 8-32 lines per controller. DMA input & output. Software transparent under DECNET. Compact package offering higher line-handling speeds & improved throughput.

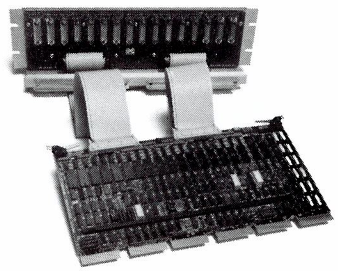
CS21/Z $2550*



Replace DEC DZ11/E and save.

Perfect if you don't need DH11 performance. Software-transparent to all DEC operating systems. Easy PROM change enables quick upgrade to DH-11 performance. Saves one slot per 16 lines.

CS21/U (VAX-11) $2844 for 16 lines*
CS21/H (PDP-11) $2520 for 16 lines*



New economical DH11-type multiplexer.

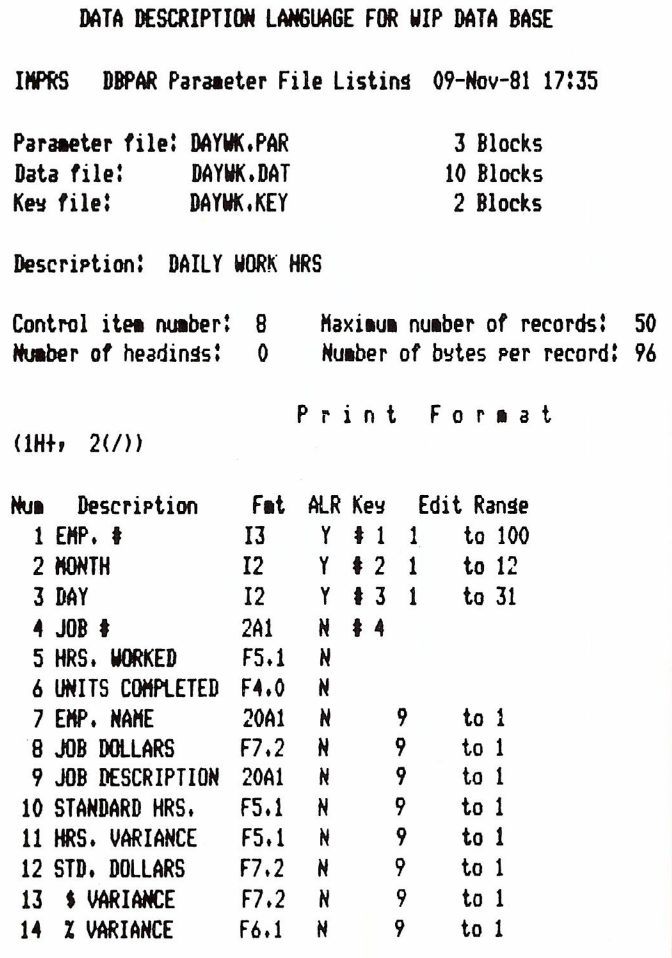
Lowest cost, high-performance communications MUX. Priced way less than DEC's DZ11, with DMA to boot. 16 RS-232 lines per board, modem control included. Can use H317 distribution panel. Transparent to PDP-11 software; Emulex software on VAX.

EMULEX
The genuine alternative

*Price each in 100 quantities. All Emulex disk, tape, and communications products can be combined to reach quantity price breaks.

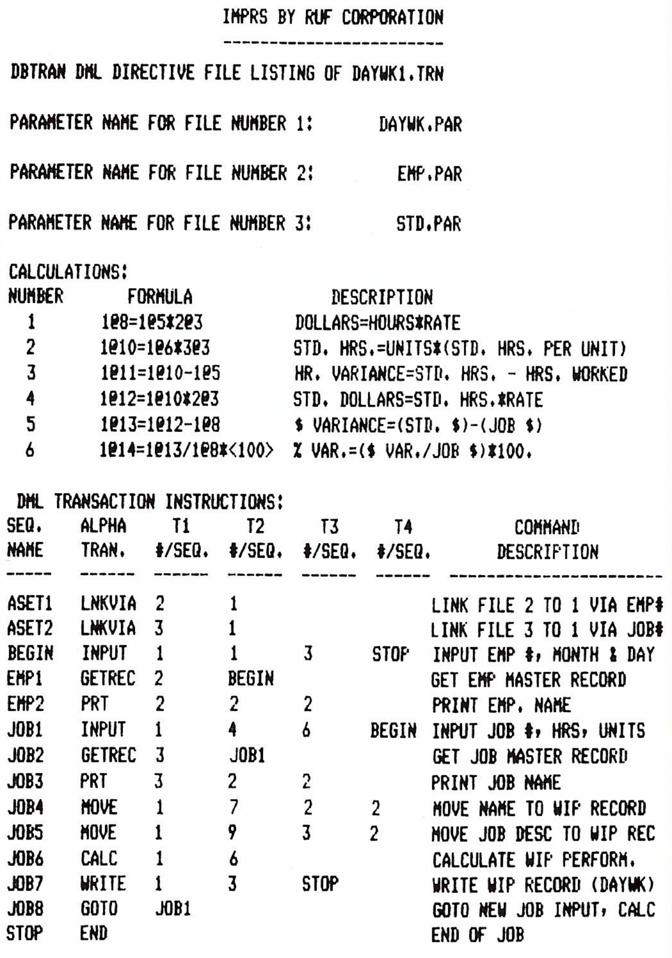
CIRCLE 58 ON READER CARD

TABLE 5.



```
       DATA DESCRIPTION LANGUAGE FOR WIP DATA BASE

IMPRS    DBPAR Parameter File Listing  09-Nov-81 17:35

Parameter file: DAYWK.PAR                 3 Blocks
Data file:      DAYWK.DAT                10 Blocks
Key file:       DAYWK.KEY                 2 Blocks

Description:  DAILY WORK HRS

Control item number:  8    Maximum number of records:  50
Number of headings:   0    Number of bytes per record: 96

                        Print  Format
(1H+,  2(/))

Num   Description      Fmt  ALR Key   Edit Range
  1 EMP. #             I3    Y  # 1  1    to 100
  2 MONTH              I2    Y  # 2  1    to 12
  3 DAY                I2    Y  # 3  1    to 31
  4 JOB #              2A1   N  # 4
  5 HRS. WORKED        F5.1  N
  6 UNITS COMPLETED    F4.0  N
  7 EMP. NAME          20A1  N     9      to 1
  8 JOB DOLLARS        F7.2  N     9      to 1
  9 JOB DESCRIPTION    20A1  N     9      to 1
 10 STANDARD HRS.      F5.1  N     9      to 1
 11 HRS. VARIANCE      F5.1  N     9      to 1
 12 STD. DOLLARS       F7.2  N     9      to 1
 13 $ VARIANCE         F7.2  N     9      to 1
 14 % VARIANCE         F6.1  N     9      to 1
```

TABLE 6.



```
                    IMPRS BY RUF CORPORATION
                    ------------------------
DBTRAN DML DIRECTIVE FILE LISTING OF DAYWK1.TRN

PARAMETER NAME FOR FILE NUMBER 1:        DAYWK.PAR

PARAMETER NAME FOR FILE NUMBER 2:          EMP.PAR

PARAMETER NAME FOR FILE NUMBER 3:          STD.PAR

CALCULATIONS:
NUMBER        FORMULA               DESCRIPTION
  1       108=105*203           DOLLARS=HOURS*RATE
  2       1010=106*303          STD. HRS.=UNITS*(STD. HRS. PER UNIT)
  3       1011=1010-105         HR. VARIANCE=STD. HRS. - HRS. WORKED
  4       1012=1010*203         STD. DOLLARS=STD. HRS.*RATE
  5       1013=1012-108         $ VARIANCE=(STD. $)-(JOB $)
  6       1014=1013/108*<100>   % VAR.=($ VAR./JOB $)*100.

 DML TRANSACTION INSTRUCTIONS:
SEQ.   ALPHA    T1     T2     T3     T4          COMMAND
NAME   TRAN.  #/SEQ. #/SEQ. #/SEQ. #/SEQ.      DESCRIPTION
-----  ------ ------ ------ ------ ------ ----------------------

ASET1  LNKVIA  2      1                  LINK FILE 2 TO 1 VIA EMP#
ASET2  LNKVIA  3      1                  LINK FILE 3 TO 1 VIA JOB#
BEGIN  INPUT   1      1      3     STOP  INPUT EMP #, MONTH & DAY
EMP1   GETREC  2      BEGIN              GET EMP MASTER RECORD
EMP2   PRT     2      2      2           PRINT EMP. NAME
JOB1   INPUT   1      4      6     BEGIN INPUT JOB #, HRS, UNITS
JOB2   GETREC  3      JOB1               GET JOB MASTER RECORD
JOB3   PRT     3      2      2           PRINT JOB NAME
JOB4   MOVE    1      7      2      2    MOVE NAME TO WIP RECORD
JOB5   MOVE    1      9      3      2    MOVE JOB DESC TO WIP REC
JOB6   CALC    1      6                  CALCULATE WIP PERFORM.
JOB7   WRITE   1      3      STOP        WRITE WIP RECORD (DAYWK)
JOB8   GOTO    JOB1                      GOTO NEW JOB INPUT, CALC
STOP   END                               END OF JOB
```

For demonstration purposes, I have selected the interactive reporter from the battery of IMPRS reporting facilities. Table 7 illustrates the ease of reporting and selecting break totals. The user can select sub-sets of data for reporting from his data base with the use of up to 100 combinations of <AND>, <OR>, or <NOR>.

Table 7 illustrates only a small part of the possible reporting required for labor performance which can be accomplished from the DAYWK.DAT file.

SYSTEM DESIGN TESTING WITH IMPRS

Because of its relational orientation, IMPRS users are not required to define all data and potential uses that will be required before they can efficiently design their system. The past inflexibility of the hard-linked tree and plex approaches have forced costly unnecessary planning which seldom gets implemented in their rigid top-down design. The cost of changing the hierarchical software is so great that it forces the system planner to cover all possibilities, disallowing the use of the more productive modified top-down design explained by Samid (4).

In my 22 years of business system design and development of large information systems, I have never experienced where it was possible to define all logical relations and usages that would be required of the system throughout its life. The most productive system design and development

Word
Processing*
RSTS/E

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

On Track Systems Provides:

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

At your convenience!

At your office!

On Track
Systems, Inc.
P.O. Box 245
Ambler, PA 19002-0245
(215) 542-7133

CIRCLE 13 ON READER CARD

"I JUST LOVE EMULEX FEATURES AND QUALITY..."
"BUT CAN YOU BELIEVE THEIR NEW LOW PRICES?"

FOR ALL THOSE LSI-11 PEOPLE WHO NEVER CONSIDERED EMULEX ON PRICE.

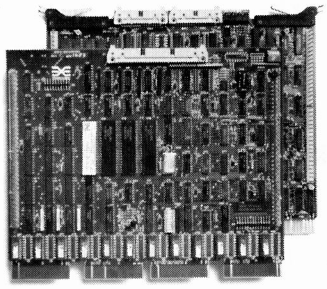
Consider the products and prices below:

You'll soon see that Emulex is as unbeatable on price as we are in quality and performance. Included are two new Winchester disk controllers with remarkably high performance and reliability (MTBF over 70,000 hours).

All Emulex Q-bus tape and disk controllers have the same microprocessor architecture and all the key features of our PDP-11 and VAX-11 products, including error correction, microdiagnostics, and software transparency. Price our performance. Write or call: Emulex Corporation, 2001 Deere Ave., Santa Ana, CA 92705; (714) 557-7580; TWX 910-595-2521. Outside Calif.: (800) 854-7112.

For immediate off-the-shelf delivery, call our national distributor: First Computer Corporation, 645 Blackhawk Dr., Westmont, IL 60559; (312) 920-1050. In Europe: Emulex Corp., 10th floor, Cory House, The Ring, Bracknell, Berkshire, England. Telephone: 0344-84234; Telex 851-849781.

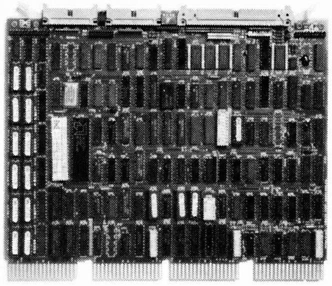
SC01 (RM02/05, RP06)



Put big SMD drives on your LSI-11.

Links Q-bus with 1-2 SMD-type drives. Software transparent & media compatible with DEC RM02, RM05, RP06. Features 3-sector data buffer, 32-bit ECC, up to half a billion bytes capacity. Over 1500 units in service!

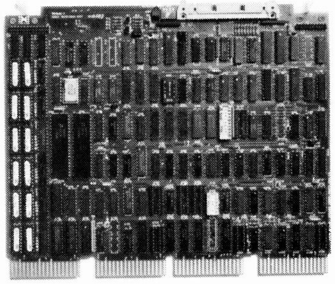
SC02 (RL01/02, RP02/03)
SC02 (RK06/07)



Low cost for smaller-sized disks.

Single quad-board interfaces LSI-11s to 8" & 14" SMD hard disk drives. Same great SC01-level performance in most applications. Software transparent. Full 32-bit ECC, self-test, 512-word bootstrap, real-time clock control, and bus terminators. Mix and match drives on one controller. 72,000 hours MTBF!

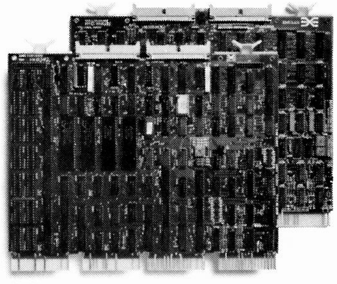
SC04 (RL01/02)
SC04 (RK06/07)



New! ANSI interfacing for 8" Winchesters.

Supports up to 8 drives per single quad-board controller. Fits into any single LSI-11 back plane quad slot. Same design, performance, and high reliability as the SC02.

TC01 (NRZ)
TC01 (PE)



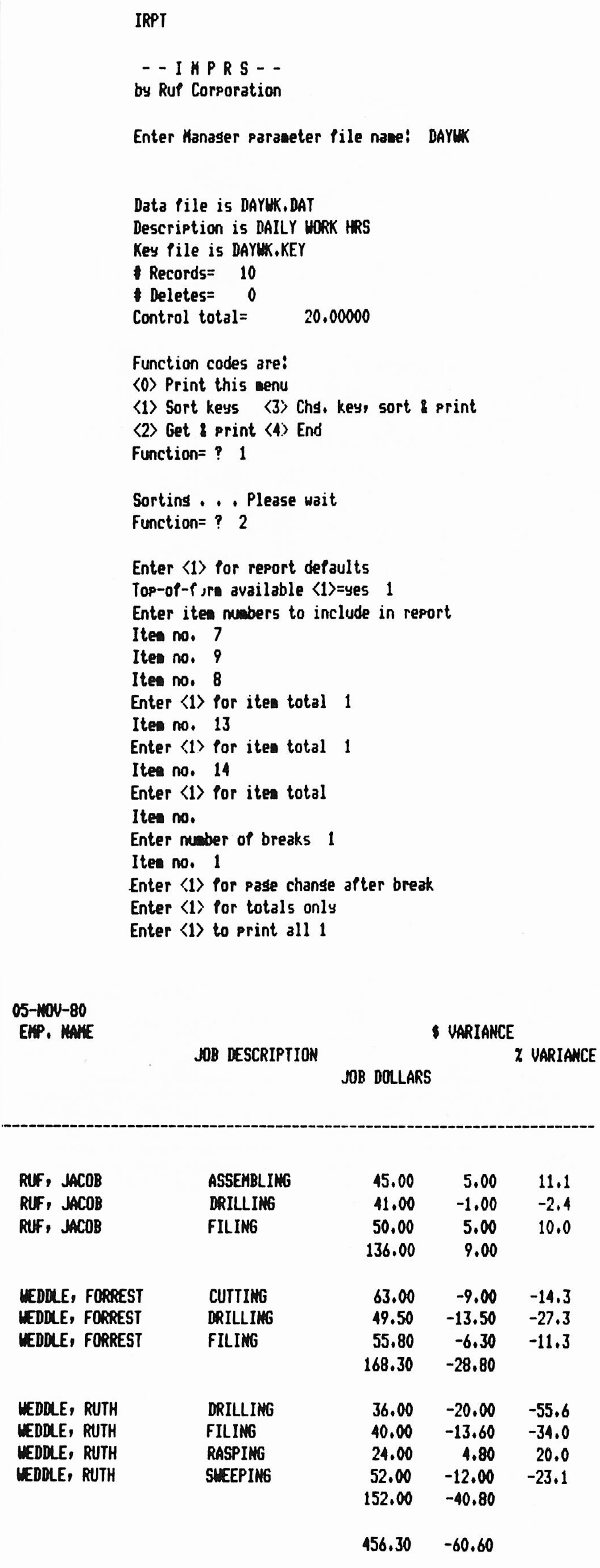
Q-bus embedded dual-density tape controller.

Handles all open-reel half-inch tapes — 800/1600 bpi, operating at 12.5-75 ips. Compatible with DEC's TU10/TM11. Daisy-chain up to 4 drives. Firmware includes a self-test and extended diagnostics. Fully embedded.

EMULEX

The genuine alternative

TABLE 7.



```
                IRPT

                 - - I M P R S - -
                by Ruf Corporation

                Enter Manager parameter file name!  DAYWK


                Data file is DAYWK.DAT
                Description is DAILY WORK HRS
                Key file is DAYWK.KEY
                # Records=   10
                # Deletes=    0
                Control total=       20.00000

                Function codes are:
                <0> Print this menu
                <1> Sort keys    <3> Chg. key, sort & print
                <2> Get & print <4> End
                Function= ?  1

                Sorting . . . Please wait
                Function= ?  2

                Enter <1> for report defaults
                Top-of-f,rm available <1>=yes  1
                Enter item numbers to include in report
                Item no.  7
                Item no.  9
                Item no.  8
                Enter <1> for item total  1
                Item no.  13
                Enter <1> for item total  1
                Item no.  14
                Enter <1> for item total
                Item no.
                Enter number of breaks  1
                Item no.  1
                Enter <1> for page change after break
                Enter <1> for totals only
                Enter <1> to print all 1


05-NOV-80
 EMP. NAME                                         $ VARIANCE
                     JOB DESCRIPTION                           % VARIANCE
                                     JOB DOLLARS

-----------------------------------------------------------------------


 RUF, JACOB             ASSEMBLING          45.00      5.00     11.1
 RUF, JACOB             DRILLING            41.00     -1.00     -2.4
 RUF, JACOB             FILING              50.00      5.00     10.0
                                           136.00      9.00

 WEDDLE, FORREST        CUTTING             63.00     -9.00    -14.3
 WEDDLE, FORREST        DRILLING            49.50    -13.50    -27.3
 WEDDLE, FORREST        FILING              55.80     -6.30    -11.3
                                           168.30    -28.80

 WEDDLE, RUTH           DRILLING            36.00    -20.00    -55.6
 WEDDLE, RUTH           FILING              40.00    -13.60    -34.0
 WEDDLE, RUTH           RASPING             24.00      4.80     20.0
 WEDDLE, RUTH           SWEEPING            52.00    -12.00    -23.1
                                           152.00    -40.80

                                           456.30    -60.60
```

philosophy that has worked effectively over the years is illustrated in Figure 8, the top-down loop-back method, used at Ruf Corp. In the design phase, the analyst defines the information system components, simulates their desired arrangement, and determines the feasibility of the planned system. These steps are repeated until the best system plan is selected which is technically, economically and operationally feasible. During the development stage the same three steps are repeated with minimal loop back into the design area assuming this phase was completed satisfactorily. It is very seldom, though, in real life, that there is not some loop back from the development to the design phase, no matter how extensively the design phase was performed.

TOP-DOWN LOOP-BACK METHOD

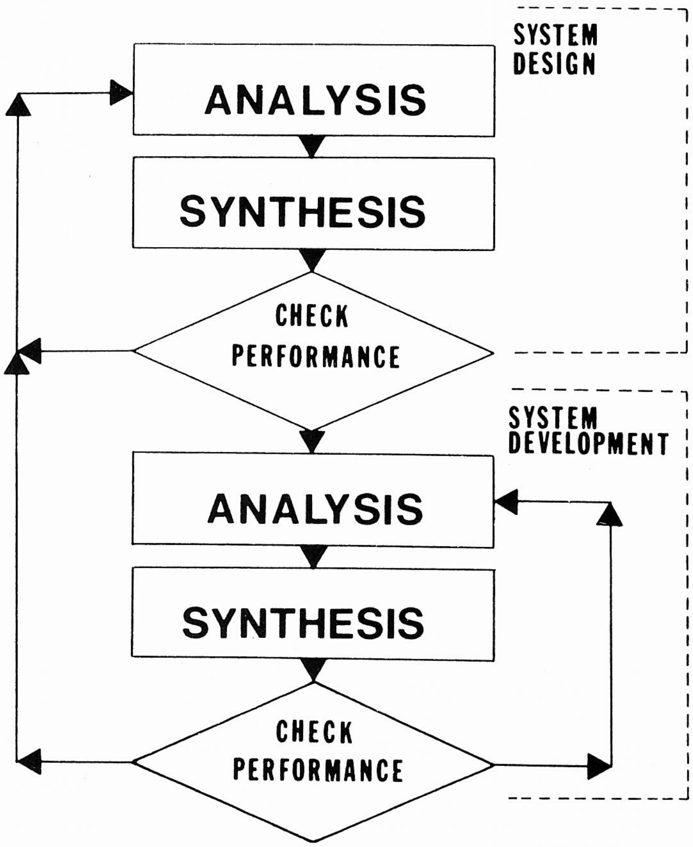


FIGURE 8.

The low cost of developing application software with IMPRS provides the ability to effectively utilize the top-down loop-back method. With IMPRS, the analyst can perform a coarse requirement definition and a coarse system design, then proceed to the development stage to test out certain design assumptions. He can therefore locate potential system problems, loop back, and retune his system design resulting in considerable overall savings.

This guided trial and error approach is not new to us. Engineers throughout the ages have developed scale models of their planned constructions to test the various design assumptions. It would definitely have paid off if this method had been used in the design of the catwalk of the Kansas City Hyatt Regency Hotel.

SUMMARY

The data base management systems of the 80's must be designed to interact with the real world. They must be flexible, providing for logical access and reporting, ease of future linkage and change. These qualities are inherent in IMPRS, a Relational Data Base Management System (RDBMS).

Because of these qualities, IMPRS users are able to isolate sub-areas and develop them with the assurance of future linkage and minimum modification.

Martin (6) says, "For the end user to program, data must be organized and represented in simple fashion. Software engineering, the process and logic of data, must be 'minimized' while information engineering, the organizing of data, must be 'maximized'".

Relational data base technology linked with data manipulation and query language capability will have a significant effect on the characteristics of our organizations. Ultimately, the availability of on-line information at much reduced cost will alter the role of management. The RDBMS and high-level languages will bring the power of the computer closer to the user. Computer hardware design will accomodate the techniques of the relational data base along with higher level languages such as that available in IMPRS.

At Ruf Corporation we have a motto which states, "We work smarter and our software works harder for you". We have accomplished this with IMPRS, a relational data base management system for the 80's.

REFERENCES

1. James Martin, **Principles of Data-Base Management**, Prentice-Hall, Inc., Englewood Cliffs, N.J., 1976, 95-110

2. Andrew B. Ferrentino, "Making Software Development Estimates 'Good'", Datamation (Sept. 1981), 179-182

3. E.F. Codd, "A Relational Model of Data for Large Shared Data Banks", Comm. ACM. 13 (June 6, 1970), 377-387

4. Gideon Samid, "Modified Top-Down Design", Datamation (Nov. 1981), 175-176

5. John M. Dodge, "Structured Programs Out, User Programs In: Martin", Software News (Dec. 7, 1981), 1-2

6. John M. Dodge, "Automators Will Be Leaders: Martin", Software News (Dec. 7, 1981), 2

XOREN IPL-II...
...THE SOFTWARE PACKAGE WHICH TRANSFERS FILES...
...BETWEEN DEC PDP-II's LSI-II's & VAX-II's...
...EVEN WHEN THEY HAVE DIFFERENT OPERATING SYSTEMS

- INTERFACE HARDWARE -
XOREN IPL-11 links the two CPU's together. No special interface hardware is required other than (in the case of remote computers) modems or acoustic couplers. Package operates via standard DEC terminal interface cards-DL11, DZ11, DH11, etc.

- DATA INTEGRITY -
CRC checking by software to CCITT recommendation V41. Recovers from errors by re-transmitting only the blocks affected.

- TRANSMISSION -
Asynchronous transmission with selectable speeds up to 9600 baud. Any type of file including binary program files may be transferred in either direction (Note RMS files should first be converted to sequential files).

- OPERATING SYSTEMS -
Versions of this package are available now to run under RSX-11M (and RSX-11M PLUS), RT-11, RSTS/E, (and equivalent versions of CTS-300 and CTS-500), TSX (and TSX-PLUS) and VAX/VMS in RSX-11M compatibility mode.

EXISTING INSTALLATIONS — Over 100 copies of XOREN IPL-11 are currently installed.

XOREN IPL-11 available from:- US: EEC Systems
286 Boston Post Road
Wayland
MA 01778,USA
Tel (617) 358 7782

UK & Europe: Xoren Computing Ltd
28 Maddox Street
London W1R 9PF
England
Tel (01) 629 5932

Note:- Xoren is currently setting up a network of distributors in the US. For further information contact Xoren Computing direct.

A2/2

CIRCLE 100 ON READER CARD

EDT HINTS & KINKS

By David Spencer, Infinity Software Corp., 2210 Wilshire Blvd., Suite 801, Santa Monica, CA 90403

1.0 INTRODUCTION

Last issue I discussed an EDT initializer file. That initializer allows EDT to perform buffer manipulation and input/output. This article is dedicated to making the most out of EDT.

2.0 EDT'S INTERNAL TABLE

Each possible editing keystroke has a unique number for it in an internal EDT table (figure 1). EDT allows access to these keystrokes by both mnemonics (such as "GOLD CONT Z") and the internal number. There are some obscure keystrokes that are definable only by their internal number, and some can be defined, but cannot used at all!

Besides being of general interest, knowledge of the numbering scheme provides us with some useful functions. First, we now know the limits to key definitions. No key that is not listed in the table may be defined for editing. Those keys which cannot be defined with a mnemonic but only with the internal number can be made of use.

Another useful by-product of a list of the internal numbers is a compressed initializer file (figure 2). Although it is more difficult to read than the initializer file from the previous issue, EDT processes it faster. The increase in speed isn't overwhelming, only ten to fifteen percent. But a savings can be made. There are those willing to accept a little unreadability for a quicker editing session start.

3.0 INTERESTING SPECIAL FUNCTION DEFINITIONS

There are some very specialized things you can do with defined keys. Here is a list of some that I have come across.

1. Macro block comment.

   This command will ask some information and create a comment block for a macro routine. It is invoked by typing "GOLD ;". You will then be asked for the routine name and a short description. These will be combined with a comment block that be inserted into the buffer.

   Insert the following text into the initializer file in the macro definition area.

```
                !+
!       MACRO BLOCK COMMENT
!-
I=M_B_C
                .SBTTL ~~/\~~ _ ~~/\~~
;+
;   ~~/\~~
;
;  DESCRIPTION:
;
;       ~~/\~~
;
;  CALLING SEQUENCE:
;
;       CALL     ~~/\~~
;
;  INPUT PARAMETERS:
;
;       NONE
;
;  OUTPUT PARAMETERS:
;
;       NONE
;
;  SIDE EFFECTS:
;
;       NONE
;-
↑Z
```

Remember the "↑Z" is uparrow Z and not CONT Z.

Insert the following text into the key definition area. Because it is so long, I have had to break it up into several lines. The phrase "<wrap>" appears as a reminder not to hit carriage-return but that I simply ran out of space on the line and continued on the next. Type it all in as one continuing string.

```
DEF K GOLD ; AS "SEL I?'Routine: '↑Z<wrap>
CUTSR=TEMPO SEL I?' Description: '↑Z<wrap>
CUTSR=TEMP1 PASTE=M_B_C<wrap>
5(-'~~/\~~') 6DC PASTE=TEMPO 2("<wrap>
6DC PASTE=TEMP1 " 6DC PASTE=TEMPO) 4V."
```

2. Redefine <cr> to insert <sp>&<cr>.

   Basic Plus Two programs require ampsersands at the end of each line. Everybody forgets to put them on all the time. The cost for missing ampersands is usually an extra program compile.

   The following key definitions allow an "ampersand" mode. Typing "GOLD &" will cause EDT to insert a space, ampersand, carriage-return for each carriage-return typed. Typing "GOLD <cr>" will exit ampersand mode.

   To add this command to EDT, insert the following text into the initializer file at the key definition area.

```
DEF K GOLD CONT M AS "EXT DEF K CONT M AS '↑M.'."
DEF K GOLD & AS "EXT DEF K CONT M AS 'I &↑Z ↑M.'."
```

3. Change lines for dial-up, VT100's with AVO

   "GOLD CONT L" toggles the screen between twenty-two lines on the screen and twelve lines. This command is very nice for use over 1200 baud lines, and with VT100's without AVO in 132 column mode.

   To add this command, insert the following lines into the initializer file at the macro definition area.

```
!+
!       SCREEN LINES MACROS
!-
DEF M LINES_12
I=LINES_12
```

The most complete line of Interface Products for PDP*-11, LSI-11 and VAX* Computers with features you can't get anywhere else

For DEC users... MDB makes the difference!

You *can* have it all! Along with the benefits of your DEC computer, you can get interface products that significantly extend the capability, flexibility and economy of DEC systems. MDB makes more DEC system boards than any independent manufacturer—some with plain vanilla compatibility; others with plain incredible performance boosting features. For example—

- Line printer controllers for every major line printer, with complete self-testing capability and optional RS-422 "long lines."
- Peripheral device controllers for card readers, X-Y plotters, electrostatic printer/plotters and paper tape readers/punches.
- DZ11 compatible multiplexors for LSI-11 and PDP-11 users; some combine RS-232 and current loop or RS-422 with a single board.
- High speed synchronous serial interfaces; one DUP-11 compatible model has all bit and byte protocols even for LSI plus X.25 capability.
- Inter processor links between Unibus computers or between Unibus and Q-bus.
- System modules including DMA modules with RS-422, general purpose parallel and digital I/O interfaces, an IEEE 488 bus controller and a programmable real time clock.
- PROM memory modules, some with an on-board PROM programmer.
- General purpose interface and bus foundation modules.
- LSI-11 based subsystems and systems with capabilities like TU-58 cartridge storage and memory management.
- LSI-11 system boxes with 22-bit addressing and switching power supplies.

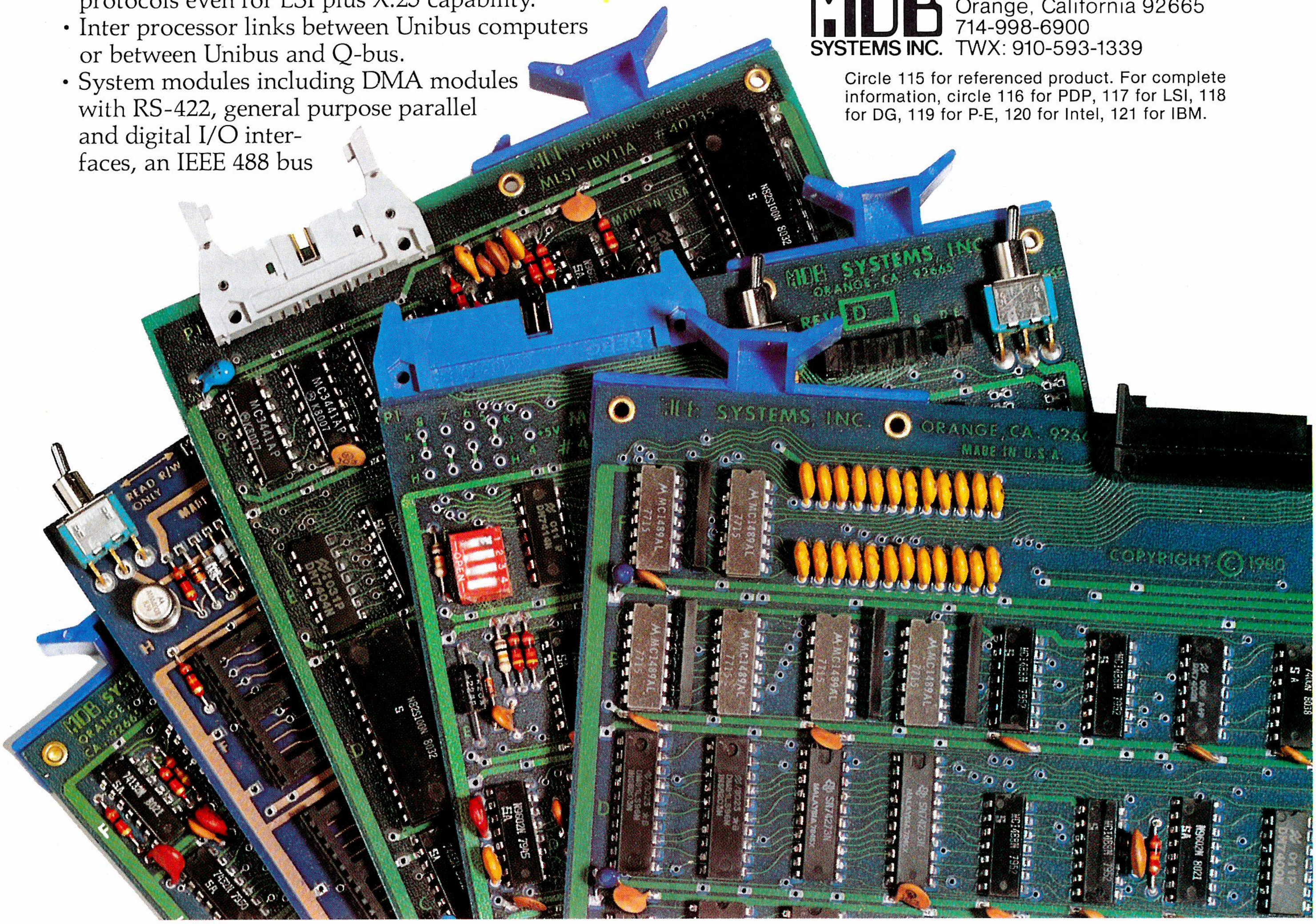
In addition to the DEC compatible products, MDB also manufactures comparable interfaces for Data General, Perkin-Elmer, Intel and IBM Series/1 computers. All MDB products are available under GSA contract #GS-00C-02851.

Give your DEC system *all* the benefits. Discover the difference MDB interface products can make.

*Trademark Digital Equipment Corp.

MDB SYSTEMS INC.
1995 N. Batavia Street
Orange, California 92665
714-998-6900
TWX: 910-593-1339

Circle 115 for referenced product. For complete information, circle 116 for PDP, 117 for LSI, 118 for DG, 119 for P-E, 120 for Intel, 121 for IBM.



```
SE LI 12
SE CU 4:7
DEF K GOLD CONT L AS "EXT LINES_22."
↑Z
!
DEF M LINES_22
I=LINES_22
SE LI 22
SE CU 7:14
DEF K GOLD CONT L AS "EXT LINES_12."
↑Z
```

Add this line to the key definition area.

```
DEF K GOLD CONT L AS "EXT LINES_12."
```

4. Define keystrokes to insert words

   A very handy timesaver for typists. A whole series of keys could be defined to insert words into the text buffer. I have here a simple definition for "CONT N" to insert the word "the". This is really a poor key to use for this purpose, but it demonstrates how easily this can be done.

   Enter this line or something like it in the key definition area.

```
DEF K CONT N AS "EXT Ithe ↑Z."
```

Some of these commands (like ampersand mode, word inserts) could be incorporated into the word delimiter macros. In other words, one could have some keystrokes defined to do useful things for programmers in programming mode, and other definitions for word processing mode.

4.0 HINTS

There are some things, when known, prevent waste and generally improve productivity.

1. Avoid journals, use "/RO"

   If you are looking at a file exclusively in an inspection mode, use the "/RO" switch. This switch will prevent any accidental changes to files, and not clutter up accounts with journal files.

2. Understand the capabilities of "/RECOVER"

   Even though I just said not to create journal files, don't ignore them altogether. The "/RECOVER" command is VERY useful to restore work obliterated by system crashes, etc. The journal file is automatically retained with the initializer "GOLD Q" quit command. Review the EDT manual and try recovering an edit or two. It's really fun to watch EDT work at warp speed reproducing your edit session.

3. Learn how to define keys

   Whenever a key is defined, try to remember to enclose it in parens and terminate it with a period. This will allow the command to be executed with a repetition count. Even though this hint is in the EDT manual, most people don't bother to read it.

4. Use "DEFINE KEY" for pseudo learn mode.

   Even though EDT doesn't have a "learn" mode, some things can be done with defining keys. For example, if you must transpose the sixth and seventh character of a number of lines, a key can be defined to do that. It's true that this isn't much, but it's the most we've got for now.

5. Use lots of buffers and pull in files.

   Buffers are cheap, so there is no reason not to use plenty. Pull in as many files as you need. Borrowing old code is a real time-saver.

6. Install Steven Edwards' EDT patches.

   Everything that I've seen Steve do has turned out to be quite useful. I recommend anything that he publishes.

5.0 KINKS

As with any new product, EDT does have a few problems. I have to admit that most of my complaints are about the screen window handler.

I don't know what the EDT development group has in store in the future, but I hope that they too have noticed these problems and are doing something about them.

1. "GOLD ." refreshes whole screen, not selected region.

   Regardless of the region size, EDT will refresh the entire screen on this command. Needless to say, this is just a little wasteful. According to the manual the "GOLD ." command is supposed to cancel the selected region. It shouldn't be too hard to refresh only the inverted area on the screen.

2. Cursor flash on end-of-line operations

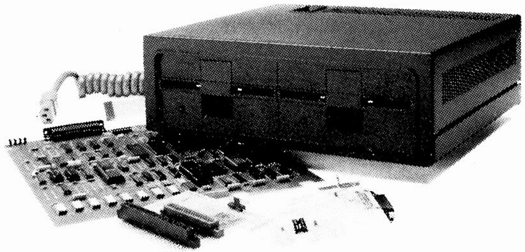
   EDT has an annoying habit of temporarily moving the cursor to the beginning of the line on any operation on the end of the line. Try it at a low baud rate. Type some text into the buffer, and go to the end of the line. Delete some characters, one at a time. On each delete, EDT will remove the character, move the start of the line, and then move back to the end of the line.

3. Unnecessarily refreshes inside tabs

   This problem shows up at low baud rates and can be best demonstrated this way. Get into EDT, type a character, a tab, and another character. Next, go back to the start of the line and type another character. Notice that EDT will refresh the entire line all the way to the end. In reality, only the line up to the tab requires a refresh. This is a simple but annoying problem.

4. "XON" determines terminal type

   When EDT is instructed to operate in character mode (usually by the initializer file), it looks at the system terminal characteristics. After determining the that the terminal is a scope, EDT checks the value of "XON". EDT assumes the terminal is a VT52 if set "NO XON". And, of course, if "XON" is set, the terminal must be a VT100.



CP/M® personal computing option for the VT100. Option owners can choose from a large library of compatible software. Exciting news !

CP/M® is a registered trademark of Digital Research, Inc.

The New VT101. Economical. Same high quality as the VT100.

The New VT125. Contains graphics package. Package can be retro-fitted.

The New VT131. Advanced video, printer port, block mode.

Digital chooses MTI for Authorized Terminal Distributor.

That's why MTI has all these new DEC terminals for you.

A lot of terminal dealers wanted to be DEC Authorized Terminal Distributors, but few were chosen. We're proud that MTI was one of the few. DEC knows MTI is a solid, growing company. We've been in business over 13 years. We've always been responsive to the latest developments in the industry and changing needs of our customers.

As an Authorized Terminal Distributor, MTI will always have the latest DEC terminals in stock, ready for delivery. And whether you are buying, renting or leasing, our prices are hard to beat. Plus, we have the expertise to match your needs to the right piece of equipment. We know what we're doing and we're proud DEC knows it.

MTI is your one source for all the terminals, peripherals, systems, applications expertise and service you'll ever need. See why DEC chose us for one of its few Authorized Terminal Distributors. Call us.

New York: 516/482-3500, 212/895-7177, 518/449-5959
Outside N.Y.S.: 800/645-8016
New Jersey: 201/227-5552
Ohio: 216/464-6688

mti

Applications Specialists & Distributors, New York, New Jersey and Ohio.
Intel, Texas Instruments, DEC, Dataproducts, Lear Siegler, Hazeltine, Diablo, Teletype, Racal-Vadic, Anderson Jacobson, General DataComm, Digital Engineering, Techtran, Cipher, Priam, SMS, Western Peripherals, Epson, Able Computer, Elgar and 3Com.

CIRCLE 107 ON READER CARD

The problem here is that EDT does not interrogate the terminal to determine its true type. Thus, unless your VT100's are set with the "XON" characteristic, they are going to be used as VT52's. This also works the other way around. Therefore, it is VERY important to review those old "$TTY.CMD" files and make sure the your scopes are set to the proper types.

6.0 WISH LIST

Here is a list of some things that could be very useful for EDT. Mostly these are things that we had in VTEDIT or KED, but didn't make the transition to EDT.

1. Learn mode

   Both VTEDIT and KED had the ability to "learn" editing keystrokes for use in later editing. This would cut down on the need for defining keys, and eliminate confusion of having to remember all the EDT nokeypad commands.

2. Default file extensions

   It would be nice to be able to define a default file extention either with a patch or a command in the initializer file.

3. View-All mode

   A TECO-like view-all mode would be very useful. Many times I have been frustrated wondering where the spaces and tabs are.

4. Standard initializer file for entire system

   Even though it's a little wasteful, I liked the idea of looking on the user account first for an initializer, and if not found, using the standard one from the system library account.

5. Make the window handler smarter

   The routine that updates the screen (window handler) does wonderful things with a VT100 terminal. However, it could be made a little smarter.

6. Allow EDT to terminate commands with any keypad key

   EDT will allow text search commands to terminate with any keypad key, why not EDT commands? This gets a little annoying at times.

7. Ability to edit search strings/ commands

   It's real nice to see what the last search string was before searching again. The same for commands.

8. TECO-like "memory" of last file edited

   Some people liked this feature, others can do without it. Is it of use? I would at least like it to be present so I could to patch it out if I didn't like it.

9. Macro local symbol reordering

   Once again, KED does this. Why not EDT?

10. Have EDT create journal files in current account

    From a system management viewpoint, it gets pretty annoying having people litter library accounts with journal files. Why not have the journal go to the same place the work file does.

RSTS/E System Managers!

Having trouble controlling usage of your computer system? Is there panic when passwords are changed? Have you ever bothered to change your passwords at all? System security is a serious problem. Announcing PASMAN, the intelligent solution.

PASMAN is a REACT account management substitute. Features include:

- Performs the common REACT functions account add and delete. Easier to use than REACT, and more "forgiving".
- New features never available before. Mass change of passwords for a single account, group, or the whole system! You supply the new password, or let PASMAN make one up.
- Identify users by a logical grouping, such as ACCOUNTING, SYSTEM, or WORD-PROCESSING. Plus, each account has its own description.
- Reports of what accounts belong to who. Prevent panic when passwords are changed. Identify user growth patterns, be able to plan ahead for upgrades.

In short, PASMAN will help you to get back control of your system.

PASMAN is available for only $350 (going up to $500 on June 1st!). For more information on PASMAN or our other products like DUMPIT, our file dump utility, or ENTRY, our callable data entry subroutine, please call or write:

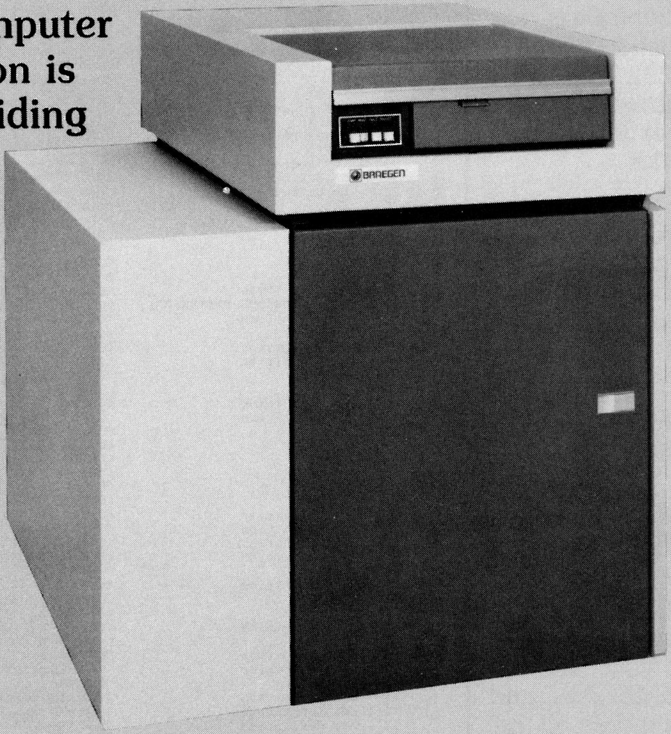
INFINITY SOFTWARE CORPORATION
2210 Wilshire Blvd.
Suite 801
Santa Monica, California 90403
(213) 820-2702

CIRCLE 87 ON READER CARD

BRAEGEN MPD HAS THE DISK STORAGE SUBSYSTEM YOU NEED

INTRODUCING THE DSL-SERIES OF DISK STORAGE SUBSYSTEMS

Braegen's Minicomputer Peripherals Division is committed to providing the systems and peripherals DEC-users need.



BENEFITS

- Compatible with *DEC *PDP-11 and *VAX Massbus systems.
- Provides *RM03/RM05 Emulation with enhancements.
- Supports mixed drive sizes.
- Supports DEC Dual Port Diagnostics.
- Supports Overlap Seeks.
- Fast delivery.
- Local service and quantity discounts available.

For more information on the DSL-Series of mass storage subsystems or any of our full line of DEC-compatible computer systems, disk and tape subsystems, printers, memories, multiplexers and more, please call or write:

Marketing Dept:

BRAEGEN
MPD
3320 East La Palma Avenue
Anaheim, California 92806
Telephone (714) 520-9200

"Be sure to ask about our field service force in over 50 cities nationwide."

*Trademarks of Digital Equipment Corporation

CIRCLE 95 ON READER CARD

Now a DEC/IBM inter-connect that's high thruput

Your high data volume transfers to IBM or CDC centrals is quickly accomplished by the COMBOARD™1231. The 1231 supports 56 Kilobaud data transfer rates over leased or private networks.

Fast transfers between systems means higher productivity in critical applications such as computer aided design. Job turnaround is decreased and system utilization is increased.

The COMBOARD1231 is the high speed model in the COMBOARD series.

COMBOARD™

COMBOARDs are 16 bit CPU based single board computers which plug straight into your DEC UNIBUS. Your COMBOARD and not the DEC host handles all the real-time interrupts and protocol processing associated with data communications.

COMBOARD1231 is the leader in low overhead and high thruput IBM inter-connects.

Contact your sales representative at 614 421-2094 or mail the coupon today.

SOFTWARE RESULTS CORPORATION

1229 West Third Avenue
Columbus, OH 43212-3090
Telephone: 614-421 2094

COMBOARD™ Software Results Corporation
DEC, UNIBUS™ Digital Equipment Corporation

CIRCLE 44 ON READER CARD

The following list shows the internal EDT key number, standard key number editing definition, and keystroke.

EDT Key	Definition	Keystroke
65535	D-C.	DELETE
0	L.	0(#)
1	W.	1(#)
2	EL.	2(#)
3	C.	3(#)
4	ADV.	4(#)
5	BACK.	5(#)
6	CUTSR.	6(#)
7	PAGETOP.	7(#)
8	(16L).	8(#)
9	APPENDSR.	9(#)
10	HELP.	PF2(+)
11	"".	PF3(+)
12	-V.	ARROW-UP(+)
13	+V.	ARROW-DOWN(+)
14	+C.	ARROW-RIGHT(+)
15	-C.	ARROW-LEFT(+)
16	SEL.	.(#+)
17	D+NL.	PF4(+)
18	DEW.	-(#+)
19	D+C.	,(#+)
20	No definition	GOLD(*)
21	.	ENTER(+)
22	(^M-C).	GOLD 0(#)
23	CHGCSR.	GOLD 1(#)
24	D+EL.	GOLD 2(#)
25	ASC.	GOLD 3(#)
26	ER.	GOLD 4(#)
27	BR.	GOLD 5(#)
28	PASTE.	GOLD 6(#)
29	EXT ?'Command: '.	GOLD 7(#)
30	FILLSR.	GOLD 8(#)
31	CUTSR=DELETE PASTE.	GOLD 9(#)
32	HELP.	GOLD PF2(+)
33	^@?'Search for: 'i^@.	GOLD PF3(+)
34	No definition	GOLD ARROW-UP(+)
35	No definition	GOLD ARROW-DOWN(+)
36	SHR.	GOLD ARROW-RIGHT(+)
37	SHL.	GOLD ARROW-LEFT(+)
38	RESET	GOLD .(#+)
39	UNDL.	GOLD PF4(+)
40	UNDW.	GOLD -(#+)
41	UNDC.	GOLD ,(#+)
42	No definition	GOLD GOLD(*)
43	(CUTSR=DELETE PASTEKS"").	GOLD ENTER(+)
44	No definition	CONTROL @(*)
45	TC.	CONTROL A
46	No definition	CONTROL B
47	No definition	CONTROL C
48	TD.	CONTROL D
49	TI.	CONTROL E
50	No definition	CONTROL F
51	No definition	CONTROL G
52	BL.	CONTROL H
53	TAB.	CONTROL I
54	DBW.	CONTROL J
55	DEFK.	CONTROL K
56	^L.	CONTROL L
57	^M.	CONTROL M
58	No definition	CONTROL N
59	No definition	CONTROL O
60	No definition	CONTROL P
61	No definition	CONTROL Q
62	REF.	CONTROL R
63	No definition	CONTROL S
64	TADJSR.	CONTROL T
65	DBL.	CONTROL U
66	No definition	CONTROL V
67	REF.	CONTROL W
68	No definition	CONTROL X
69	No definition	CONTROL Y
70	EX.	CONTROL Z
71	No definition	GOLD CONTROL @(*)
72	No definition	GOLD CONTROL A
73	No definition	GOLD CONTROL B
74	No definition	GOLD CONTROL C
75	No definition	GOLD CONTROL D
76	No definition	GOLD CONTROL E
77	No definition	GOLD CONTROL F
78	No definition	GOLD CONTROL G
79	No definition	GOLD CONTROL H
80	No definition	GOLD CONTROL I
81	No definition	GOLD CONTROL J
82	No definition	GOLD CONTROL K
83	No definition	GOLD CONTROL L
84	No definition	GOLD CONTROL M
85	No definition	GOLD CONTROL N
86	No definition	GOLD CONTROL O
87	No definition	GOLD CONTROL P
88	No definition	GOLD CONTROL Q
89	No definition	GOLD CONTROL R
90	No definition	GOLD CONTROL S
91	No definition	GOLD CONTROL T
92	No definition	GOLD CONTROL U
93	No definition	GOLD CONTROL V
94	No definition	GOLD CONTROL W
95	No definition	GOLD CONTROL X
96	No definition	GOLD CONTROL Y
97	No definition	GOLD CONTROL Z
98	No definition	GOLD CONTROL [(*)
99	No definition	GOLD CONTROL \(+)
100	No definition	GOLD CONTROL](+)
101	No definition	GOLD CONTROL ^(+)
102	No definition	GOLD CONTROL _(+)
103	No definition	GOLD SPACE
104	No definition	GOLD !
105	No definition	GOLD "
106	No definition	GOLD #
107	No definition	GOLD $
108	No definition	GOLD %
109	No definition	GOLD &
110	No definition	GOLD '
111	No definition	GOLD (
112	No definition	GOLD)
113	No definition	GOLD *
114	No definition	GOLD +
115	No definition	GOLD ,
116	No definition	GOLD -
117	No definition	GOLD .
118	No definition	GOLD /
119	No definition	GOLD 0(*)
120	No definition	GOLD 1(*)
121	No definition	GOLD 2(*)
122	No definition	GOLD 3(*)
123	No definition	GOLD 4(*)
124	No definition	GOLD 5(*)
125	No definition	GOLD 6(*)
126	No definition	GOLD 7(*)
127	No definition	GOLD 8(*)
128	No definition	GOLD 9(*)
129	No definition	GOLD :
130	No definition	GOLD ;
131	No definition	GOLD <
132	No definition	GOLD =
133	No definition	GOLD >
134	No definition	GOLD ?
135	No definition	GOLD @
136	TC.	GOLD A
137	No definition	GOLD B
138	No definition	GOLD C
139	TD.	GOLD D
140	TI.	GOLD E
141	No definition	GOLD F
142	No definition	GOLD G
143	No definition	GOLD H
144	No definition	GOLD I
145	No definition	GOLD J
146	No definition	GOLD K
147	No definition	GOLD L
148	No definition	GOLD M
149	No definition	GOLD N
150	No definition	GOLD O
151	No definition	GOLD P
152	No definition	GOLD Q
153	No definition	GOLD R
154	No definition	GOLD S
155	TADJSR.	GOLD T
156	DBL.	GOLD U
157	No definition	GOLD V
158	REF.	GOLD W
159	No definition	GOLD X
160	No definition	GOLD Y
161	EX.	GOLD Z
162	No definition	GOLD [
163	No definition	GOLD \
164	No definition	GOLD]
165	No definition	GOLD ^
166	No definition	GOLD _
167	No definition	GOLD `
168	No definition	GOLD a(*)
169	No definition	GOLD b(*)
170	No definition	GOLD c(*)
171	No definition	GOLD d(*)
172	No definition	GOLD e(*)
173	No definition	GOLD f(*)
174	No definition	GOLD g(*)
175	No definition	GOLD h(*)
176	No definition	GOLD i(*)
177	No definition	GOLD j(*)
178	No definition	GOLD k(*)
179	No definition	GOLD l(*)
180	No definition	GOLD m(*)
181	No definition	GOLD n(*)
182	No definition	GOLD o(*)
183	No definition	GOLD p(*)
184	No definition	GOLD q(*)
185	No definition	GOLD r(*)
186	No definition	GOLD s(*)
187	No definition	GOLD t(*)
188	No definition	GOLD u(*)
189	No definition	GOLD v(*)
190	No definition	GOLD w(*)
191	No definition	GOLD x(*)
192	No definition	GOLD y(*)
193	No definition	GOLD z(*)
194	No definition	GOLD {
195	No definition	GOLD |
196	No definition	GOLD }
197	No definition	GOLD ~
198	No definition	GOLD DELETE
199	No definition	[unknown]
200	No definition	[unknown]

This is a keypad key.

* This key can be defined by using the internal EDT number, but cannot be used by either case conversion constraints or key sequence interpretations.

+ This key is a valid for editing, but can be defined using the internal EDT key number only.

FIGURE 1. Default EDT Key Assignments

LETTERS TO THE RSTS Pro . . .

. . . is your column! Send us your comments, suggestions, or notes of interest to the RSTS community. We'd enjoy hearing from you.

The following initializer file creates an editing environment identical to that made by the initializer file in the last issue. This initializer uses the internal key number instead of the mnemonic to define keystrokes. EDT will start a little faster when using a compressed file.

```
DEF M DELIM_PROG
F=DELIM_PROG
I
DEF K 75 AS "EXT DELIM_WP."
^Z
C; ISE EN WO '^Z 9ASC 10ASC 11ASC 12ASC 13ASC 27ASC I ()[],-+*/='¨Z EX
^Z
DEF M DELIM_WP
F=DELIM_WP
I
DEF K 75 AS "EXT DELIM_PROG."
^Z
C; ISE EN WO '^Z 9ASC 10ASC 11ASC 12ASC 13ASC 27ASC I ,'^Z EX
^Z
DEF M WIDTH_132
I=WIDTH_132
DEF K 94 AS "EXT WIDTH_80."
SE SC 132
^Z
DEF M WIDTH_80
I=WIDTH_80
DEF K 94 AS "EXT WIDTH_132."
SE SC 80
^Z
DEF K 46 AS "-W."
DEF K 50 AS "+W."
DEF K 51 AS "PASTE=?'Put buffer: '."
DEF K 60 AS "PAR."
DEF K 68 AS "CUTSR=?'Cut buffer: '."
DEF K 34 AS "(-22V)."
DEF K 35 AS "(+22V)."
DEF K 75 AS "EXT DELIM_WP."
DEF K 78 AS "CUTSR=DELETE PASTE=?'Rep buffer: '."
DEF K 79 AS "(-C D-C C UNDC)."
DEF K 94 AS "EXT WIDTH_132."
DEF K 95 AS "EXT CO SELECT TO=?'Cop buffer: ' ; F L."
DEF K 97 AS "EXT EX."
DEF K 117 AS "I~~/\~~^Z -6C."
DEF K 118 AS "S%~~/\~~%%."
DEF K 137 AS "EXT F=?'Buffer: '.."
DEF K 138 AS "(C SEL W CHGCSR)."
DEF K 141 AS "(SEL PAR FILLSR)."
DEF K 144 AS "EXT INC ?'Input file: ' =?' Buffer: '."
DEF K 147 AS "EXT F L."
DEF K 148 AS "EXT F=MAIN.."
DEF K 150 AS "EXT WR ?'Output file: ' =?' Buffer: '."
DEF K 152 AS "EXT QUIT/SAVE."
DEF K 154 AS "EXT SH BU."
SE WR 79
SE TR
SE K
SE M C
DELIM_PROG
F=MAIN
```

FIGURE 2. Initializer Using Internal Numbers

7.0 CONCLUSION

My conclusion about EDT is: use it! It may be slightly flawed, but it's a lot faster than VTEDIT. It's a young product that will only become better.

At the Los Angeles DECUS meeting I had the chance to talk with the EDT people. They insured me that future releases of EDT would correct some of the problems I mentioned. Unfortunately, when we might see any future releases I couldn't find out.

I solicit any additional ideas, comments, and corrections. As space and volume permits, I will gladly share them with readers in future columns. Correspondence can be sent to:

INFINITY SOFTWARE CORPORATION
2210 Wilshire Blvd
Suite 801
Santa Monica, California 90403
(213) 820-2702

PDP-11/
RSTS USERS

Our Generalized Reporting System (GRS) is absolutely the most powerful information retrieval and formatting tool available for RSTS systems. It enables you to:

- Give your users ad-hoc access to your database with an English-like query facility
- Extend the life of old applications
- Develop new reports in minutes, instead of days
- Free up your programmers for more important assignments

GRS is easily interfaced to any RSTS file system or DBMS. And it's available on a one-month trial basis for only $100!

Can you really afford to be without GRS?

Call or write today for free literature and licensing information.

etc ENTERPRISE TECHNOLOGY CORPORATION
305 Madison Avenue
New York, NY 10165
(212) 972-1860
Telex 177959

CIRCLE 4 ON READER CARD

UNITRONIX
means
DEC

IMMEDIATE DELIVERY from $6MM warehouse inventory

TERMINALS
MODEMS
PDP 11/03, 11/23 + 11/44
SYSTEMS
With RL01s & RL02s
or Winchester Drives
INTERFACES
FLOPPYS
Std. & Custom Software

SAVINGS SPOTLIGHT
FREE
DEC Personal Computer
with purchase of a computer system

CALL FOR LATEST PRICES
(201) 231-9400
Sale or Lease Avail.

UNITRONIX
CORPORATION
197 Meister Ave. • Somerville, NJ 08876 • TELEX: 833184
CIRCLE 25 ON READER CARD

How To Use BUILD

VERSION: V7.1-01

By Richard W. Hill, Software Techniques, Inc., Los Alamitos, CA

No portion of this document may be reproduced for any purpose without the express permission of Software Techniques, Inc.

The information in this document is subject to change without notice and should not be construed as a commitment by Software Techniques, Inc. Software Techniques, Inc. assumes no liability for any errors which may appear in this document.

1.0 INTRODUCTION

While designing the installation procedure for our new A/P System, I discovered a serious lack of documentation regarding the BUILD program supplied with RSTS. This article is an attempt to correct this problem.

BUILD is designed to perform three basic functions:

- Read an input control file.
- Process the contents of that file.
- Produce an appropriate command file for ATPK execution.

The command file contains all of the commands necessary to build and/or patch a system. BUILD generates this command file by combining the responses to prompts with commands present in the control file. BUILD stores the responses to the prompts as values for various BUILD and user defined variables. I will refer to these variables as symbols or substitution symbols to avoid the confusion between the variables in the BUILD program and these special control file variables. As each symbol is encountered in the control file, it is replaced with the associated replacement value. The means of defining and identifying symbols will be discussed in more detail later.

2.0 BUILD COMMANDS

Seven commands are recognized by BUILD. Each command is prefixed with a dollar sign "$", and must be at least four characters long (including the dollar sign), with the exception of $BOOT which must be five characters long.

The BUILD commands are:

- $BOOT
- $BREAK
- $DOPAT
- $END
- $FORCE
- $PATCH
- $PROMPT

The first six commands are all used by BUILD for patching purposes. The last command, $PROMPT, is used for everything else. Due to the flexibility of the $PROMPT command, we will look at it before dealing with the patching commands.

3.0 SUBSTITUTION SYMBOLS

Substitution symbols in BUILD Control files are composed of:

- A tilda "~"
- Symbol name (1 to 6 characters long)
- A colon ":"

The replacement values for these symbols are character strings with a length of no more than twenty-six characters. A replacement value may be defined as null (length of zero).

Substitution symbols and their replacement values are defined by BUILD and with the $PROMPT command in the control file. The default replacement values for a symbol are denoted by placing the replacement value between slashes "/" immediately following the symbol. This default replacement is only used when the symbol has not been defined.

The following examples demonstrate the use of symbols, replacement values and default replacements.

Assume that:

OUT will be replaced by "SY:[10,21]"
IN will be replaced by "MTO:[1,2]"
MTMODE will be replaced by "/MO:2"

The BUILD control file entry is:

```
PIP ~OUT:/NL:/ = ~IN:/NL:/JUNQUE.IT~MTMODE:///W
```

This line would be translated by BUILD to:

```
PIP SY:[10,21] = MTO:[1,2]JUNQUE.IT/MO:2/W
```

Assume that:

OUT is not defined
IN will be replaced by "DM1:[1,2]"
MTMODE is null

The BUILD control file entry is:

```
PIP ~OUT:/NL:/ = ~IN:/NL:/JUNQUE.IT~MTMODE:///W
```

This line would be translated by BUILD to:

```
PIP NL: = DM1:[1,2]JUNQUE.IT/W
```

NOTE

After the substitution symbol "MTMODE", we have placed a null default replacement and then the "/W" switch of PIP. This was done to ensure that the "/W" was not interpreted as the default replacement. If we had left out the null default replacement (//), then the "/W" would only have appeared if "MTMODE" was not defined.

4.0 BUILD CONTROL FILES

The BUILD control files may contain the following types of commands:

- BUILD commands.
- Indirect BUILD control file references.
- ATPK commands recognized by BUILD.
- General commands and text to be processed later by ATPK.

BUILD will process each line read from the Control file

DILOG DISC & TAPE COMPATIBLES FOR DEC-11*

DILOG offers the widest range of single board DEC emulating disc and magnetic tape controllers for LSI-11, 11/2, 11/23, PDP-11 and VAX-11 compatibility.

DILOG/DEC controllers:

DISC

- Winchester—5¼″, 8″ and 14″
- SMD—Storage Module
- CMD—Cartridge Class
- Lark—Cartridge Class
- 2315/5440 Cartridge Class
- Flexible Disc

MAGNETIC TAPE

- ½″ reel-to-reel NRZI/PE—Streaming or conventional
- ¼″ 3M Cartridge

A wide range of emulations are offered. All fully supported by DEC operating systems.

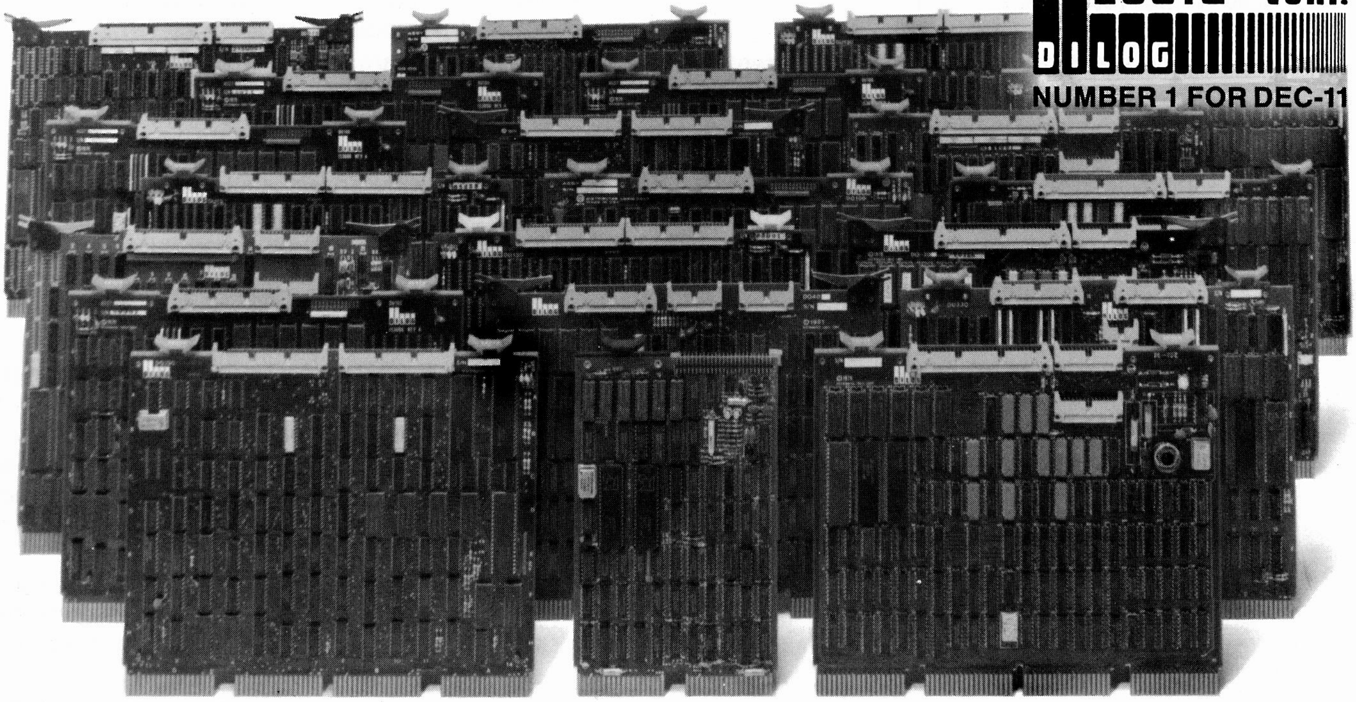
If you're interfacing *any* popular drive to *any* DEC-11 computer, contact DILOG for the largest selection of DEC-11 compatible disc and magnetic tape controllers! Over 20 available now. Several more to be announced soon.

CORPORATE HEADQUARTERS
12800 Garden Grove Blvd. • Garden Grove, Calif. 92643
• Phone: (714) 534-8950 • Telex: 681 399 DILOG GGVE

EASTERN REGIONAL SALES OFFICE
64-A White Street • Red Bank, New Jersey 07701
• Phone: (201) 530-0044

EUROPEAN SALES/SERVICE OFFICE
12 Temple Square • Aylesbury, Buckinghamshire • England
• Phone: 44-296-34319 or 34310 • Telex 837 038 DILOGI G

DISTRIBUTED
LOGIC CORP.
DILOG
NUMBER 1 FOR DEC-11



*Trademark Digital Equipment Corp.

CIRCLE 15 ON READER CARD

in the following manner:

1. All symbols are replaced by their respective replacement values.
2. Indirect command files will be opened and the next command retrieved.
3. BUILD commands which are the first thing on a line are processed.
4. The commands to the BASIC editor, OLD, APPEND, and COMPILE, will be processed to generate commands for the appropriate language processor, BASIC-PLUS, BASIC-PLUS-2, or CSPCOM.
5. Any other commands or text, will be placed as they are found in the Command file.

5.0 RUNNING BUILD

To run BUILD, log into a privileged account and enter:

```
RUN [1,2]BUILD
```

BUILD will print its header and prompt the user for five responses, along with two optional prompts.

(a) System Build <No> ?

This prompt determines what kind of action BUILD will take when processing the control file. If the response is any text starting with "Y", then BUILD will not issue the "Control file is ?" prompt. The input control file will be "BUILD.CMD", found on the input device in account [1,2], used with system generation.

If "NO" or <LF> is entered, then the control file will be specified by the user at the "Control File is ?" prompt.

(b) Source Input Device <SY:> ?

This prompt determines what device BUILD is to read the control file from. In addition this is where the input files for the installation procedure are copied from.

The format for this entry is:

```
Logfile = Device:/Switch
```

The logfile specification is optional. If present it will be passed to ATPK for use as the logfile. The default logfile name is "SY:BLDnnn.LOG", where "nnn" is the current job number.

If the input device is not specified it will default to the device from which BUILD was run. If this device is a private disk, then it will default to "SY:".

All switches are optional and are used primarily for magtape devices. The valid switches are:

- /DOS — Input magtape is in DOS format.
  Illegal switch if the input device is not magtape.
- /ANSI—Input magtape is in ANSI format.
  Illegal switch if the input device is not magtape.
- /DENSITY:n — Input magtape has a density of n.
  Legal values are 800 and 1600.
  Illegal switch if the input device is not magtape.
- /PARITY:xxx — Input magtape has a parity of xxx.
  Legal values are "ODD" and "EVEN".
  Illegal switch if the input device is not magtape.
- /DETACH — Detach before actual processing starts.

(c) Library Output Device <SY:> ?

This prompt determines on what device the library account resides. This device must be a disk. The default is "SY:".

(d) Target System Device <SY0:> ?

This prompt determines on what disk device the target system will be built. The default is "SY0:".

(e) Library Account <[1,2]> ?

This prompt determines on which account the library utilities (such as PIP, UTILTY, CSPCOM, TKB, etc.) are located. Any utilities installed by the BUILD procedure will be output to this account. BUILD will optionally create the account if it does not exist.

(f) Control File is ?

This prompt determines the main control file to be used by BUILD. This prompt is not issued if this is a system build (see prompt (a).) Unless otherwise specified, BUILD assumes this file is on the input device, in account [1,2], and has a file type of .CTL. There is no default for this filename.

(g) Additional Control File is <None> ?

This prompt determines if any additional control files are to be processed at the this time. If you wish to perform several installations at one time and the control files are all on the same device, then specify the name of the next control file. Otherwise just press <CR> or <LF> to start the actual build.

6.0 PRE-DEFINED SUBSTITUTION SYMBOLS

The following substitution symbols are predefined by BUILD.

- INPDEV — Input device.
- INPUT — "INPDEV" + Input account.
- LIBDEV — Library output device.
- SYSDEV — Target system device.
- LIBACC — Library account.
- SYSTEM — "LIBDEV" + "LIBACC"
- RUNLIB — "SYSTEM" unless ATPK was not found there or we are doing a system build, in which case it is "SY:[1,2]".
- SYSACC — "LIBACC" unless BUILD was chained to by PBUILD, in which case "SYSACC" is defined by PBUILD.
- SYSDSK — "SYSDEV" + "SYSACC"
- MTMODE — "/MO:2" if the input device is magtape. (No rewind on a file search). Otherwise it will be null.
- LB — Current location of "LB:", set by the "$PROMPT LB" or the "$PROMPT ALB" commands.
- PATLOC — Location of the patch files, set by a chain from PBUILD or by the "$PROMPT PATCH" command.
- SAVDEV — Location of the saved patched sources, set by a chain from PBUILD or by the "$PROMPT PATCH" command. Defaults to "SYSTEM" if the specified location was not on disk.
- RTS.NM — RTS to compile programs against. This is

set by the "$PROMPT RTS" command.

- DEXT: — Default file type, set by the "$PROMPT RTS" command.
- CSPCOM — Hold result of "Use CUSP compiler" prompt (YES/NO). This is set by the "$PROMPT RTS" command.
- OLB — Object library name for compiles, set by the "$PROMPT RTS" command.

7.0 $PROMPT

The $PROMPT command will cause BUILD to print a prompt and receive input pertaining to that prompt. This command will also assign a replacement value to a symbol. Both of these functions are determined by the parameters used in the command.

The $PROMPT command has two basic formats. The first format is:

```
$PROMPT xxx
```

Where "xxx" is one of the pre-defined modifiers listed below:

- ! — Comment entry internal to the Control file.
- ALB — Prompt for the location of LB:
- LB — Prompt for the location of LB:
- PATCH — Prompt for patching information
- RTS — Prompt for default RTS information

The other format is used for other special prompts, messages, and substitution symbol assignments. This format is:

```
$PROMPT String-1, String-2, Integer-1, String-3, String-4
```

Where:

- String-1 is the prompt to print
- String-2 is the default to print
- Integer-1 is a bit encoded flag word
- String-3 is the default file specification
- String-4 is the symbol

7.1 $PROMPT — Format 1

I will first discuss the five modifiers for the $PROMPT command, to see how they are used and what substitution symbols they define.

7.1.1 $PROMPT ! — INTERNAL BUILD COMMENT.

This command is used for internal comment entries within the BUILD control files. Any line starting with the "$PROMPT !" command will be ignored. This command is used to make comment entries which will not be included in the ATPK command file.

7.1.2 $PROMPT ALB — PROMPT FOR THE LOCATION OF "LB:".

This command prompts for the location of the logical "LB:". The result of this command is shown below.

```
Locate logical 'LB:' on <SY:[1,0]> ?
```

The default response to this prompt is the current location of "LB:" or "SY:[1,1]" if the logical was not defined. The response will be checked for validity. It must be a device and/or PPN specification only. If the specified account does not exist, BUILD will ask if you want to create it.

If the Library output disk and the Target System disk are both part of the public disk structure then the logical

RSTS/E SOFTWARE PACKAGES

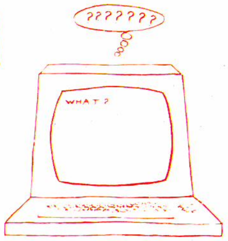
- **KDSS,** a multi-terminal key-to-disk data entry system. (Also available for RSX-11M.)
- **TAM,** a multi-terminal screen-handling facility for transaction-processing applications. (Also available for RSX-11M.)
- **FSORT3,** a very fast sort. Directly sorts RSTS/E files containing up to 16 million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

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

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

CIRCLE 29 ON READER CARD

DEAR RSTS MAN:



Send questions to: DEAR RSTS MAN, P.O. Box 361, Fort Washington, PA 19034-0361.

DEAR RSTS MAN:

I have an 11/44 system which unfortunately came with dual TU58 drives (DECTAPE-II tape cartridges). I have yet to find a use for these little wastes-of-money except for running diagnostics, as they are bootable. If I could put the RSTS initialization code on one, I could boot-up a TU58 (I even have the bootstrap ROM chip) to the Option: level to re-init a pack, do a SAVRES, or whatever. They seem to hold about 500 blocks or so, they should be large enough.

My question is: How can I put a bootable copy of INIT.SYS on a DECTAPE-II cartridge? I've tried to HOOK.SAV it without success.

Bret A. Bailey, Data Proc. Mgr.
Econ. Oppor. Assn., Inc.
Brookville, PA

Dear Bret: *As far as we know, there is no supported (or otherwise) way to do what you want. You can write on it with FIT, but that's the RT11 file structure, and HOOK doesn't know about it. If you could de-assemble the boot block . . .*

DEAR RSTS MAN:

I just received my first VT131 and it doesn't work right. My Terminal Vendor told me that a VT131 was a VT100 with advanced video and a printer port that could do block mode but didn't have to. He reported that they had decided to carry these rather than the VT102 which doesn't do block mode. They thought that the very small price differential between the VT102 and VT131 wouldn't matter.

Why doesn't my software work properly with the VT131? VT

Dear VT: *DEC has done it again! As you know DEC VT terminals return a special escape sequence when queried by escape sequence from a program. The sequence can be used to tell terminals apart (i.e. VT52's, VT50's, VT100's, etc.). Although this terminal will respond to most commands like a VT100 with advanced video, the escape sequence it returns upon a query is different!*

A VT100 returns ESC[?1;nC where n tells you about the terminal attributes (Advanced video, printer port). The VT131 returns ESC[?7C which is quite different from the VT100. Because the 1; is missing from the VT131 you will have to re-program your code to recognize the VT131. A shame indeed that DEC couldn't have used some compatible way to tell us it is a VT131. Worse, some of my code gives bad and unexpected errors when it encounters a VT131. ILLEGAL NUMBER AT LINE xxx. Maybe the VT102 is a better buy.♥

DEC
SYSTEMS
&
COMPONENTS

C.D. SMITH &
ASSOCIATES, INC.
12605 E. Freeway,
Suite 318
Houston, TX 77015
(713) 468-2384

CIRCLE 54 ON READER CARD

"LB:" will be removed and added with the new specification Otherwise a local device assignment for "LB:" will be made in the ATPK command file.

Once the "$PROMPT ALB" command has been executed, additional occurances of this command will be ignored.

This command will define the substitution symbol "LB" with the value of the new logical "LB:".

7.1.3 $PROMPT LB — PROMPT FOR THE LOCATION OF "LB:".

This command prompts for the location of "LB:". It works the same way as does the "$PROMPT ALB" with one exception, if either the Library output disk or the Target system disk is a private disk then this command will be ignored.

Once the "$PROMPT LB" command has been executed, additional occurances of this command will be ignored.

7.1.4 $PROMPT PATCH — PROMPT FOR PATCHING INFORMATION.

This command is used to determinine if patching is to be done during the installation, and if so where the patch files are located.

The "$PROMPT PATCH" command will cause the following prompts to be issued.

```
  Function (Build/Patch, Patch, Build) <Build/Patch> ?
 Patch file input location <SY:[200,200]> ?
 Save patched sources <No> ?
 Write patched sources to <SY:[200,200]> ?
```

7.1.4.1 Function (Build/Patch, Patch, Build) <Build/Patch> ?

If the user enters "BUILD", then the remainder of the prompts are skipped. During the processing of the rest of the BUILD control file, the $DOPAT command will be ignored, but the other patching commands will be executed as normal.

If the user enters "PATCH", then only patching will be done. During the processing of the rest of the BUILD control file everything but "$PROMPT", "$DOPAT", and "$BREAK" commands will be ignored.

If the default response is taken then both patching and the build will be done.

7.1.4.2 Patch file input location <SY:[200,200]> ?

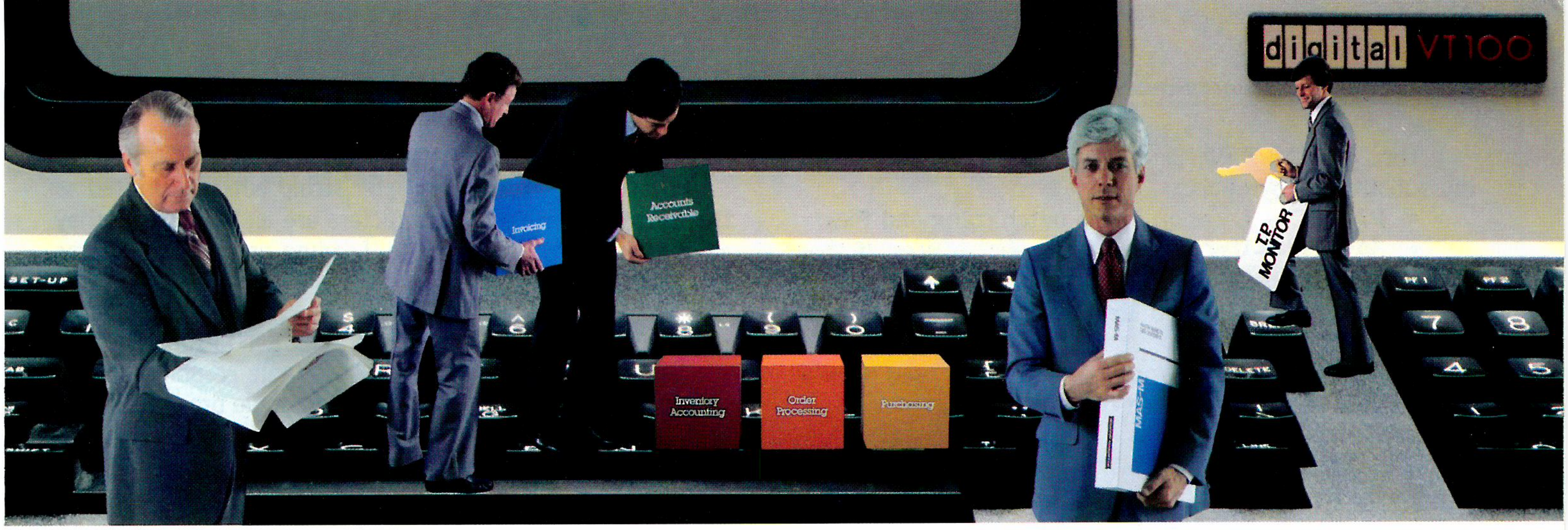
This prompt determines the location of the patch command files. The specified location must be a disk device (use [1,2]PATCPY to copy the files from tape to disk first). The default location is "SY:[200,200]".

7.1.4.3 Save patched sources <No> ?

This prompt determines if you desire to save the patched BASIC-PLUS sources. The only sources which will be saved are the BASIC-PLUS sources which are patched with CPATCH using the $PATCH command.

7.1.4.4 Write patched sources to <SY:[200,200]> ?

If you desire to save these sources then this prompt is issued to determine where the patched sources will be written to. The default response is "SY:[200,200]". This location does not have to be a disk device.



You can get more from your RSTS system with MAS-M.

MAS-M is the application software system from Martin Marietta Data Systems that can help you do more with your DEC hardware. That's because MAS-M is the on-line software system that gives you much more than you'd expect from packaged software.

More Flexibility.

MAS-M's modular design lets you choose from 10 different application systems:

- 🟩 Accounts Receivable
- 🟩 Accounts Payable
- 🟩 General Ledger
- 🟩 Order Processing
- 🟦 Invoicing
- 🟦 Inventory Control
- 🟪 Inventory Accounting
- 🟥 Bill of Materials
- 🟧 Material Requirements Planning
- 🟨 Purchasing

You can implement just the modules you need to satisfy your demands. And no matter which combination you choose, the MAS-M system is always fully integrated.

MAS-M's flexible design also makes it easy to install, and simple for your users to operate. And, since MAS-M is written in BASIC-PLUS-2, and based on the RMS-11 data management system, the software is fully compatible with your current RSTS/E operating system and DEC software.

More Control.

You can count on MAS-M for more comprehensive data accuracy and security, too.

MAS-M's powerful transaction processing MONITOR gives you maximum control over your data—from start to finish. User passwords and menu selections are checked against user security profiles. Data entry validation is also standardized in the MAS-M MONITOR, so any invalid data can be corrected *before* it reaches your application program.

More Productivity.

MONITOR is also an important tool in developing new applications. You can use MONITOR to create input screens and validation rules online. And, MONITOR can help you improve programmer productivity by providing a standard framework for input of code that minimizes the difficulties of user interface and terminal characteristics.

More Support.

You can count on Martin Marietta Data Systems for system development and implementation, comprehensive training, and clear, concise documentation. We can also provide an extensive Maintenance Service to support your MAS-M system.

What it all adds up to is a packaged software system that can give you everything you need to get your jobs done. And more. Write or phone us today, and we'll tell you more about how the MAS-M software system can work for you.

MAS-M

The Software System That Can Help You Do More.

MARTIN MARIETTA
DATA SYSTEMS

Martin Marietta Data Systems
Marketing Services, R/H
6303 Ivy Lane, Greenbelt, MD 20770
(301) 982-6500

MARTIN MARIETTA

CIRCLE 111 ON READER CARD

The "$PROMPT PATCH" command defines the following substitution symbols:

- PATLOC — Location of the patch command files.
- SAVDEV — Location of the saved sources. If they are not being saved to a disk device then this becomes the same as the substitution symbol "SYSTEM".

This command will be ignored after being used once or when BUILD has been chained to from PBUILD.

7.1.5 $PROMPT RTS — PROMPT FOR RTS INFORMATION.

This prompt determines under what run-time system ATPK is to start up the controlled job performing the build.

The prompt from this command is:

```
Run-Time System <RSX > ?
```

The default response is the current system default run-time system. An error will be printed and the prompt issued again if any of the following are true:

- The specified run-time system was not found in "SY:[0,1]" with a file type of ".RTS".
- The specified run-time system has not been added as a keyboard monitor.
- The default file type for executable programs under this run-time system is not ".TSK" or ".BAC".

If the selected run-time system was "RSX" or "BP2COM" then a second prompt is issued to determine whether to use the CSPCOM compiler or not. This prompt is:

```
Use the CUSP compiler 'CSPCOM' <Yes> ?
```

If the selected run-time system was "RSX" then the default for this prompt is "YES", otherwise the default is "NO".

If the run-time system has not been installed, BUILD will ask if you want it installed. Respond with "YES" to have BUILD install it.

Upon completion of the $PROMPT RTS command, the following symbols are defined.

- RTS.NM — The specified run-time system name.
- DEXT — The default RTS file type (".TSK" or ".BAC")
- CSPCOM — "YES" or "NO" depending if CSPCOM will be used or not
- OLB — Either "CSPCOM" if CSPCOM is being used, or the RTS name

Once the "$PROMPT RTS" command has been executed additional occurances of the command are ignored.

7.2 $PROMPT — Format 2

The second format of the $PROMPT command allows the assignment of user defined substitution symbols for use in processing in the remainder of the control file. I will discuss the various arguments for the second format of the $PROMPT command. Then at some common uses of this command. The format is:

```
$PROMPT String-1, String-2, Integer-1, String-3, String-4
```

Where:

- String-1 is the prompt to print
- String-2 is the default to print
- Integer-1 is a bit encoded flag word
- String-3 is the default file specification
- String-4 is the symbol

7.2.1 STRING-1 — THE PROMPT TO PRINT

This argument is a character string which is printed as the pseudo prompt. The characters are all left just as they were found in the control file. Leading and trailing spaces and tabs however are removed.

7.2.2 STRING-2 — THE DEFAULT VALUE TO PRINT

This string is printed as the default response in the prompt. This string will be enclosed in angle brackets ("<", ">".) It remains unaltered by BUILD and is only used as the printed default. No default is printed if this string is null.

7.2.3 INTEGER-1 — BIT ENCODED FLAG WORD

This integer tells BUILD how to process the $PROMPT command. With the various bits of this word set and/or cleared, BUILD can force the input to be a file specification or part of one, a specific response, or just an informational prompt. If this argument is not a valid integer BUILD will abort.

The bit values are:

- 0 (1) — Lookup filename
  If this bit is set, then BUILD will verify that the file or account currently exists. Otherwise, the file or account will not be looked up. If the file was not found, BUILD will re-prompt for a correct filename.

- 1 (2) — Allow wildcards
  If this bit is set wildcards are allowed within the filename (but not the PPN.) If this bit has been cleared, then no wildcards are allowed.

- 2 (4) — Allow/Disallow device specification
  If this bit is set, then a device name is allowed within the entered file specification. Otherwise no device specification is allowed.

- 3 (8) — Allow/Disallow Ppn
  If this bit is set, then an account number is allowed within the file specification. Otherwise no account number is allowed.

- 4 (16) — Expand null Device/PPN to SY: and current account
  If this bit is set, then if the user does not enter a device specification, it will be expanded to SY:. If an account is not specified, then it will be expanded to include the current account.

- 5 (32) — Disallow/Allow filename
  If this bit is set, then a filename will not be permitted. Only a device and/or PPN will be allowed depending on the status of bits 2 and 3. Otherwise, a filename will be allowed.

- 6 (64) — Check input against values
  If this bit is set, then the input will be checked against the values following the $PROMPT command line. The values must be in the following format:
  n

```
Response-1 = Replacement-Value-1
Response-2 = Replacement-Value-2
    :                  :
    :                  :
Response-n = Replacement-Value-n
```

Where "n" is the number of valid responses allowed. If the user response is found in the list of valid responses, then the replacement value opposite it will be used as the replacement value for the substitution symbol specified in the argument list. Otherwise the valid responses will be listed and the prompt will be re-executed.

- 7 (128) — Check for a number in range
  If this bit is set, then BUILD will allow a number to be entered by user. It must fall within a specified range. The range is specified in an argument following the command line, with the format:
  Low-limit >< High-Limit
  The number entered, if valid, will be converted to a character string and be used as the replacement value for the substitution symbol specified in the argument list.

- 8 (256) — Create account if not already there
  If this bit is set, then BUILD will ask the user if he wishes to create the account number entered if it does not exist. For this function to be enabled bits 0 (lookup filename) and 5 (filename not allowed) must be set.

- 9 (512) — Not Used

- 10 (1024) — Not Used

- 11 (2048) — Not Used

- 12 (4096) — Just print prompt
  If this bit is set, then only the prompt is printed, there is no attempt to get any input.

- 13 (8192) — Allow a random string (other than a file name)
  If this bit is set, BUILD will allow the input to be any character string. Otherwise BUILD will assume that the input is part of a file specification and all spaces and tabs will be removed.

- 14 (16384) — Do not input or print anything
  If this bit is set then no prompt is printed, nor is any input requested. The default however will be printed unless it is null or bit 15 is set.

- 15 (-32768) — Do not print the default
  If this bit is set then the default is not printed.

7.2.4 STRING-3 — DEFAULT FILE SPECIFICATION

This field is the default file specification. Any portion of an entered file specification which are missing will be taken from this specification.

7.2.5 STRING-4 — THE SUBSTITUTION SYMBOL

This is the definition for a symbol to be replaced by the value entered at the prompt. If the substitution symbol has already been defined by either a $PROMPT command or by BUILD, it will be replaced by the new definition.

7.3 Using $PROMPT

As you can see the $PROMPT command is very versatile with many uses. We will look at some of the uses for this command, concentrating on the second format. The various uses for the $PROMPT revolve around the bit values of the integer flag word.

(a) Print an informational prompt
Sometimes it is desirable to print a heading before the actual prompts start or to give additional information prior to issuing a prompt. To do this use a value of 4096 for the flag word.

```
$PROMPT ** Software Techniques **,,4096,,JUNQUE
```

This will cause the following message to be printed.

```
** Software Techniques **
```

As you will note, the default to print and the default file specification are missing. This should always be done to avoid the possibility of the command being processed incorrectly. All $PROMPT commands (format 2) must have a substitution symbol specified. The best thing to do is to use the same symbol (JUNQUE in the above example) for all of your informational prompts. If you use a symbol

Two satisfied Amcor customers

Don Chick and Greg Jones, Automation, Inc., Omaha, Nebraska, Service Bureau, using the power of AMBASE in an integrated environment with AMCOR's Accounting and Business Control Application products.

"we write everything with Ambase"

"AMBASE training was well presented and made our use of AMBASE easier.
We are using AMBASE to write custom programs and customize modules within other products. It takes less time and we write everything exclusively with AMBASE.
We are able to put out a report in a matter of hours as opposed to days. We couldn't live without it."

AMBASE is a revolutionary state-of-the-art system for application development and data base management. AMBASE is increasing programming productivity worldwide from 100-900%. In addition to the data management system, AMBASE includes a report generator, query language, screen formatter and automatic code generator.

If you would like to find out more about AMBASE, write to 1900 Plantside Dr., Louisville, KY 40299, U.S.A. or call, 502-491-9820. We will send you free information immediately.

amcor computer corp.
a Kaneb company

Headquarters: Louisville, KY Regional Offices: Nationwide Telex 204122

CIRCLE 3 ON READER CARD

EEC SYSTEMS

LEX-11
WORD AND DATA PROCESSING FOR VAX, PDP-11 AND LSI- 11

- Integrated word, data list processing
- Supports RT-11, TSX-PLUS, RSX-11M, RSTS/E, UNIX, IDRIS, IAS, VMS
- Fast, friendly, flexible
- Calculator, Graphics
- Spelling Dictionary, Forms
- Custom modification easy
- ASCII Formated Files
- Report Writer, Data Entry
- OEM Discounts

EEC Systems
Dept. RST
286 Boston Post Road
Wayland, MA 01778
(617) 358-7781/2
(617) 443-6376

CIRCLE 43 ON READER CARD

which has already been defined, the replacement value of that symbol will be discarded and a null string will take its place.

(b) Require a device name only

```
$PROMPT Enter a device name,SY:,37,SY:,DEVICE
```

This will appear as:

```
Enter a device name <SY:> ?
```

If you don't want the default to be printed, then use a flag word of -32731 (1+4+32+(-32768)).

(c) Enter an existing filename

```
$PROMPT Enter a filename,JUNQUE.IT,17,JUNQUE.IT,FILE
```

This will appear as:

```
Enter a filename <JUNQUE.IT> ?
```

The response to this prompt will cause BUILD to lookup the file entered or JUNQUE.IT if nothing was entered, in the current account on SY:. If the file was not found BUILD will print an error message and re-prompt you.

(d) Enter a specific response

Require the user to enter a specific response to the prompt such as YES or NO.

```
$PROMPT Really continue,No,8256,,YES,NO
2
YES=YES
NO=NO
```

This will appear as:

```
Really continue <No> ?
```

If an invalid response is given, BUILD will display the correct responses and re-prompt as is shown below:

```
Valid options are:
          YES
          NO
Really continue <No> ?
```

(e) Enter a replacement file type

Use BUILD to do a replacement on a file type as input by the user.

```
$PROMPT Enter the file type,B2S,64,.B2S,EXT
2
B2S=.B2S
BAS=.BAS
```

This command when executed will assign the file type which was entered, along with the a preceeding dot to the symbol "EXT". The symbol "~EXT" may then be used in the command file to cause the replacement of all occurances of "~EXT:" with the desired file type. For example:

```
Enter the file type <B2S> ? BAS
```

Then the command:

```
PIP JUNQUE~EXT: = MTO:JUNQUE~EXT:
```

Will appear as:

```
PIP JUNQUE.BAS = MTO:JUNQUE.BAS
```

(f) Logic manipulation in BUILD control files

Through the responses to $PROMPT pseudo prompts, substitution symbols can be used as logicals with true/false values. The definition for logic values is as follows:

Logic value	Replacement value
True	Null
False	$PROMPT ! False

To define the logic values as defined in the above table, we will use the $PROMPT command. For this example we will define the user input as a Yes/No response with Yes = True and No = False.

```
$PROMPT Really continue,No,8256,,YES.NO
2
YES=
NO=$PROMPT ! False
~YES.NO:$PRO Installation continuing,,4096,,JUNQUE
```

As can be seen in this example, the substitution symbol "YES.NO" is set to "TRUE" if the entry was "YES" and to "FALSE" if the entry was "NO". Then the next command, an informational prompt, is prefaced with this symbol. If "YES.NO" is "TRUE" then the command will be executed because "YES.NO" will have a null replacement value. Otherwise the command will be ignored because it is comment entry ($PROMPT!).

The following example shows this command and the result when the user enters "YES".

```
Really continue <No> ? YES
Installation continuing
```

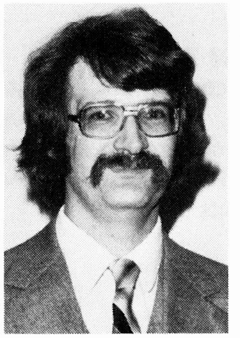
These logic substitution symbols are designed to preface a line to determine if that line is to be processed or ignored. Using this technique, a line may be prefaced with more than one substitution symbol. This will be treated as an "AND" condition. The line will be processed if all the replacement values are "TRUE".

(g) Setting up two opposite conditionals

If we want to set up two substitution symbols with opposite logic values we will use a dummy $PROMPT. In other words a $PROMPT which causes nothing to be printed or input.

```
$PROMPT Really continue,No,8256,,YES.NO
2
YES=YES
NO=NO
$PROMPT *Dummy (yes=true)*,~YES.NO:,-8128,,ANS1
2
YES=
NO=$PROMPT ! False
$PROMPT *Dummy (no=true)*,~YES.NO:,-8128,,ANS2
2
YES=$PROMPT ! False
NO=
~ANS1:$PRO Installation continuing,,4096,,JUNQUE
~ANS2:$PRO Installation being aborted,,4096,,JUNQUE
```

In this example "ANS1" is set "TRUE" if the response was "YES" and "ANS2" is set "TRUE" if the response was "NO". Then only one message will be printed depending upon the user response. The value -8128 (64 + 8192 + 16384 + (-32768)) is used to check the value of "YES.NO" against the defined responses and to define the new replacement symbol without any prompt or default being printed.



and another one

Ken Haarstad, Lead Programmer, Turtle Mountain Community School, Turtle Mountain Indian Reservation, North Dakota.

"Ambase training... a good foundation"

"Everything we have developed over the past year has been under AMBASE; our student registration packages for high school and college, the fund accounting system for the school district, the test scoring program, even the payroll system.

AMBASE training gave me a good foundation. Installation was very easy, and customer service is excellent. I always get a quick response.

If someone were to take AMBASE away from me, I would quit. It speeds things up so much."

AMBASE is a revolutionary state-of-the-art system for application development and data base management. AMBASE is increasing programming productivity worldwide from 100-900%. In addition to the data management system, AMBASE includes a report generator, query language, screen formatter and automatic code generator.

If you would like to find out more about AMBASE, write to 1900 Plantside Dr., Louisville, KY 40299, U.S.A. or call, 502-491-9820. We will send you free information immediately.

amcor computer corp.
a Kaneb company

Headquarters: Louisville, KY Regional Offices: Nationwide Telex 204122

CIRCLE 3 ON READER CARD

The following example shows this command and the result when the user enters "NO".

```
Really continue <No> ? NO
Installation being aborted
```

(h) Aborting a BUILD with $PROMPT

The $PROMPT may also be used to abort BUILD. This is done in two steps, the first is to define the abort message and the second is to do the actual abort. When the abort is executed, BUILD will close the control file and output command file, print the abort message and return you to monitor control. The abort message will appear as:

```
?Program aborted — xxxx
```

Where "xxxx" is the abort message which must be no more than 26 characters long.

The abort message is defined with the following $PROMPT command.

```
$PROMPT *Abort Message*,xxxx,-8192,,ABORTS
```

To define the message for an abort, the substitution symbol must be "ABORTS". The value -8192 (8192 + 16384 + (-32768)) is used to store the abort message (xxxx) without any prompt or default being printed.

The actual abort is done with the following command:

```
$PROMPT *Aborting*,YES,-8192,,ABORT
```

The substitution symbol must be "ABORT" and the replacement value must be "YES" for an abort to take place. In this example we used a dummy prompt and forced the "YES" response.

8.0 PATCHING FROM BUILD

The following commands are used with the patching function of BUILD. Patching, however will not be done unless the "Patch" or "Build/Patch" function was selected with the "$PROMPT PATCH" command.

These commands are:

- $BOOT
- $BREAK
- $DOPAT
- $END
- $FORCE
- $PATCH

8.1 $BOOT n

This command will cause BUILD to be reloaded by ATPK after all of the commands up to the $BOOT have been processed. The $BOOT command will only be processed if the value "n" is greater than the current value of BT.LEV% which is defined at line 1000 of BUILD. Currently this value is zero, so $BOOT 1 will cause BUILD to be booted.

The following actions take place in response to the $BOOT command.

1. The input control file and the output command file are closed.
2. BUILD chains to ATPK.
3. ATPK processes the command file and chains back to BUILD.
4. BUILD will read through the control file ignoring everything except $DOPAT and $PROMPT commands until the $BOOT command is reached. The "$PROMPT PATCH" command is ignored after a boot.
5. BUILD processes the remainder of the control file normally.

8.2 $BREAK

This command is the terminator for a $DOPAT command control block. See section 8.3 for information on the $DOPAT command. If this command is found anywhere but the end of a $DOPAT command control block it is treated as a no-op.

8.3 $DOPAT @Filename

The $DOPAT command will cause the input control file to be changed to the patch command file specified by "Filename". The default file type on the patch command file is ".CMD". The contents of this patch command file are patch commands using the BUILD commands $END, $FORCE, $PATCH, and $PROMPT. See the sections on these commands for more information on their use.

The $DOPAT command will only be used if the "$PROMPT PATCH" command was processed and the user choose to "Patch" or "Build/Patch". If the $DOPAT command is executed, all commands after it are ignored until a $BREAK command is found. If the $DOPAT is not executed then the commands following it are used and the $BREAK is treated as a no-op.

8.4 $END Filename

This command is used to terminate the $FORCE and $PATCH command control blocks. In addition to terminating these control blocks, when terminating the $PATCH command, it will cause the program just patched to be compiled. The "Filename" is the name of the new compiled program. The "Filename" defaults to the file specified with the $PATCH command. There is an optional switch for this command to specify that no compiling is to be done. This switch is "/NC".

8.5 $FORCE

The $FORCE command is used to place special patching commands in the ATPK command file. This command starts a command control block which contains the special patching commands. The control block is terminated with either the $END or the $PATCH commands.

An example of the use of this command would be when a BASIC program is to be patched through the $PATCH command but requires several modules to be appended prior to patching. You would place the necessary append commands in this control block along with a command to save the program to a temporary file for patching.

The $FORCE can also be used to do patching with ONLPAT. All that would have to be done in this case is to place the commands necessary to invoke ONLPAT in the control block.

The $FORCE command causes some special processing to be done on the commands within the control block.

1. All normal BUILD substitutions are performed.
2. The BUILD commands $END, $PATCH, and $PROMPT are properly executed.
3. The following special substitutions are performed:
   - I: is replaced by the input device and the input account (INPUT).
   - S: is replaced by the system device and the input account (SYSDSK).
   - L: is replaced by the library device and the input account (SYSTEM).
   - O: is replaced by the location of the patched sources (SAVDEV).

   These special substitutions are only done when the substitution string is immediately preceeded by one of the following:

   Double quote mark, single quote mark, left bracket, left parenthesis, semi-colon, comma, equal sign, space, or a horizontal tab.
4. The text with all substitutions made is placed in the ATPK command file.

8.6 $PATCH Filename

The $PATCH command is used to set up the commands for patching with CPATCH. The $PATCH command starts a command control block which is terminated with either the $END or the $FORCE commands. The filename specified is the name of the BASIC program to be patched. This filename will have a default file type of ".BAS".

We will look at the result of a sample $PATCH command control block. The run-time system is BASIC, the patch files are in SY:[200,200], and the sources will be saved in SY:[200,201]. The $PATCH command control block is:

```
$PATCH JUNQUE
JUNQUE.PAT
$END JUNQUE
```

The above commands will produce:

```
RUN SY:[1,2]CPATCH
SY:[200,201]JUNQUE.BAS=SY:[1,2]JUNQUE.BAS
SY:[200,200]JUNQUE.PAT
↑Z
↑Z
SCALE 0
OLD SY:[200,201]JUNQUE.BAS
COMPILE SY:[1,2]JUNQUE
```

The filename specified in the $PATCH command must exist, if it does not then BUILD will abort with an error.

9.0 BUILD INDIRECT COMMAND FILES

BUILD will allow references to indirect command files. These indirect command files may be nested 15 deep.

To denote an indirect command file reference, use the commercial at sign (@) before the command file name. This command must be the only thing on the line. For example:

```
@JUNQUE.CMD
```

The default file type for the indirect command file is "CMD".

To place indirect command references for other programs or utilities such as PIP place an underscore (__) before the "@". ATPK will discard the underscore character if it is the first character on the line. For example:

```
RUN ≐RUNLIB:PIP.SAV
__@APBLD1.CMD
__@APBLD2.CMD
__@APBLD3.CMD
↑Z
```

This will cause the following to be sent to the PK by ATPK:

```
RUN SY:[1,2]PIP.SAV
@APBLD1.CMD
@APBLD2.CMD
@APBLD3.CMD
↑Z
```

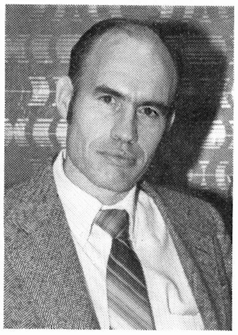
10.0 ATPK COMMAND RECOGNIZED BY BUILD

The only ATPK command which is recognized and processed by BUILD is the "$DETACH" command. When BUILD detects this command it sets up the ATPK command line with the "/DET" switch. This is the same as is done for the "/DETACH" command on the input device prompt. All additional occurances of the command will be ignored. After detecting the "$DETACH" command, BUILD will not prompt for an additional control file when the end of the current control file is reached.

11.0 OLD, APPEND, and COMPILE commands

The commands to the BASIC editor, OLD, APPEND, and COMPILE will be treated differently depending on the responses to the "$PROMPT RTS" prompt. The best way to describe the actions taken by BUILD when encountering these commands is to show the result of a simple command when each of the various options are selected.

and another one



Sam Walden, Vernay Laboratories, Inc., Yellow Springs, Ohio, manufacturer of laboratory gaskets, using AMCOR'S AMBASE and AMFACS systems in an integrated environment.

"Ambase is making our lives easier..."

"AMBASE is making our lives easier through aid messages, and more reliable data. It puts the data in a common area.

Amcor's training facilities are very conducive to learning and the training program is complete. Installation of AMBASE was very, very, very, easy, and Amcor's support is good.

Using AMBASE in an integrated environment with Amcor's application products is advantageous to us because the systems have a number of things built into them that would have taken us a long time to develop. Getting everything under AMBASE completely will bring about programming conformity."

AMBASE is a revolutionary state-of-the-art system for application development and data base management.

AMBASE is increasing programming productivity worldwide from 100-900%. In addition to the data management system, AMBASE includes a report generator, query language, screen formatter and automatic code generator.

If you would like to find out more about AMBASE, write to 1900 Plantside Dr., Louisville, KY 40299, U.S.A. or call, 502-491-9820. We will send you free information immediately.

amcor computer corp.
a Kaneb company

Headquarters: Louisville, KY Regional Offices: Nationwide Telex 204122

CIRCLE 3 ON READER CARD

The following commands have been placed in the BUILD control file.

```
$PROMPT RTS
OLD JUNQUE
APPEND JUNK.APP
COMPILE JUNQUE/TKB
```

These commands can produce the five different results shown depending upon the responses to the "$PROMPT RTS" command. The "/TKB" switch is used to generate the commands to task build the program if necessary under the specified language processor.

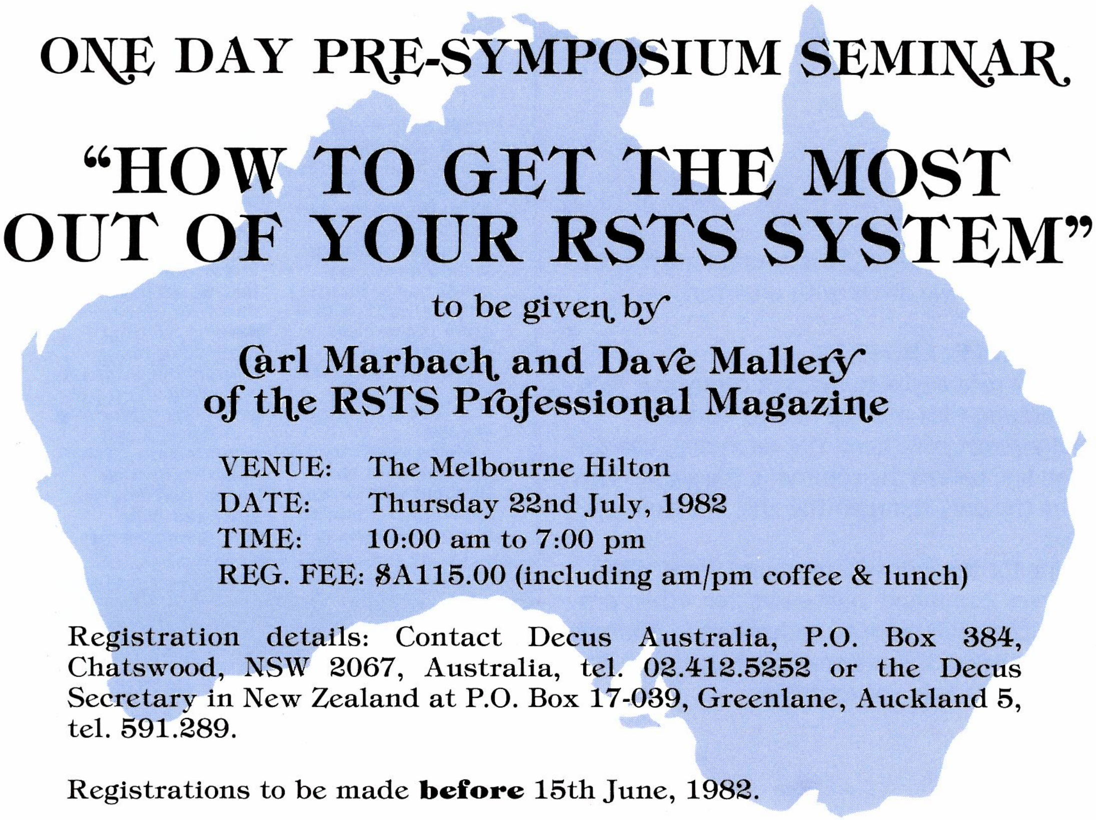
(a) Using the BASIC RTS.

```
SCALE 0
OLD JUNQUE
APPEND JUNK.APP
COMPILE JUNQUE
```

(b) Using the BP2COM RTS and without CSPCOM.

```
SCALE 0
OLD JUNQUE
APPEND JUNK.APP
COMPILE JUNQUE.OBJ/CHA/LIN/NODEB/OBJ
RUN SY:[1,2]TKB.TSK
JUNQUE.TSK/FP=JUNQUE.OBJ,SY:[1,1]BP2COM.OLB/LB
/
HISEG=BP2COM
UNITS=12
ASG=SY:5:6:7:8:9:10:11:12
//
```

1982 DECUS AUSTRALIA SYMPOSIUM



ONE DAY PRE-SYMPOSIUM SEMINAR

"HOW TO GET THE MOST OUT OF YOUR RSTS SYSTEM"

to be given by

Carl Marbach and Dave Mallery
of the RSTS Professional Magazine

VENUE: The Melbourne Hilton
DATE: Thursday 22nd July, 1982
TIME: 10:00 am to 7:00 pm
REG. FEE: $A115.00 (including am/pm coffee & lunch)

Registration details: Contact Decus Australia, P.O. Box 384, Chatswood, NSW 2067, Australia, tel. 02.412.5252 or the Decus Secretary in New Zealand at P.O. Box 17-039, Greenlane, Auckland 5, tel. 591.289.

Registrations to be made **before** 15th June, 1982.

CIRCLE 125 ON READER CARD

RSTS PROFESSIONAL

Box 361 · Ft. Washington, PA 19034-0361 · (215) 542-7008

□ PAYMENT ENCLOSED for one year's subscription (6 issues).
US 3rd class, $35 / Canada & US 1st class, $50 US /
All other countries air mail, $60, payable in US dollars.

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

Please send BACK ISSUES circled:	Vol. 1, #1	Vol. 2, #3	Vol. 3, #2	Vol. 4, #1
□ $10 per issue enclosed.	Vol. 2, #1	Vol. 2, #4	Vol. 3, #3	Vol. 4, #2
□ Bill me for $12.50 per issue.	Vol. 2, #2	Vol. 3, #1	Vol. 3, #4	

□ Send me a RSTS PRO Tee Shirt — $6.95 S M L XL (Adult Sizes Only)

Name ____________________

Address ____________________

____________________ Suite ________

City/State/Zip ____________________

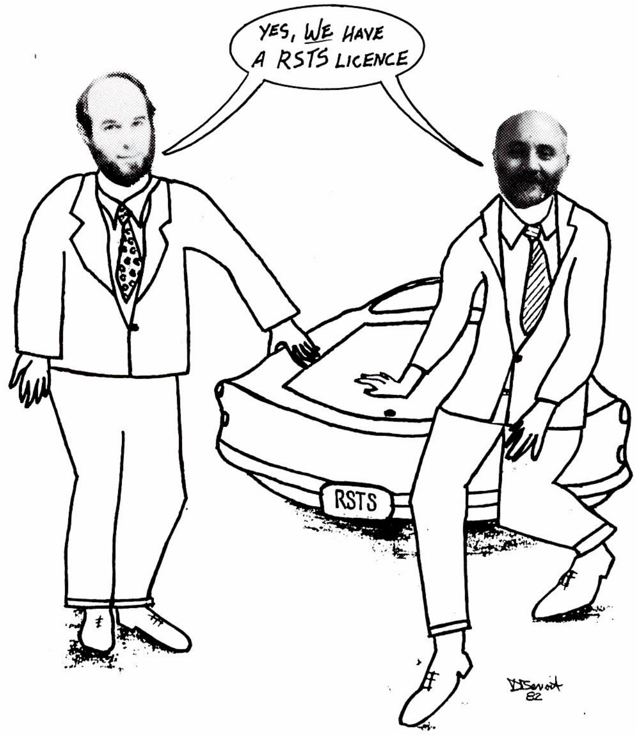
Country ____________________ Phone ( ) ____________

FREE CLASSIFIED AD WITH SUBSCRIPTION!!

Your first 12 words are absolutely FREE, only $1.00 per word thereafter.
Use the space provided below.

____

V4.2



```
        RUN SY:[1,2]PIP.SAV
        JUNQUE.OBJ/DE:NO
        ↑Z
```

(c) Using the BP2COM RTS with CSPCOM.

```
        RUN SY:[1,2]CSPCOM.TSK
        JUNQUE.OBJ/OBJ=JUNQUE,JUNK.APP
        ↑Z
        RUN SY:[1,2]TKB.TSK
        JUNQUE.TSK/FP=JUNQUE.OBJ,SY:[1,1]CSPCOM.OLB/LB
        /
        HISEG=BP2COM
        UNITS=12
        ASG=SY:5:6:7:8:9:10:11:12
        //
        RUN SY:[1,2]PIP.SAV
        JUNQUE.OBJ/DE:NO
        ↑Z
```

(d) Using the RSX RTS without CSPCOM.

```
        SCALE 0
        OLD JUNQUE
        APPEND JUNK.APP
        COMPILE JUNQUE.OBJ/CHA/LIN/NODEB/OBJ
        RUN SY:[1,2]TKB.TSK
        JUNQUE.TSK/FP=JUNQUE.OBJ,SY:[1,1]RSX.OLB/LB
        /
        UNITS=12
        ASG=SY:5:6:7:8:9:10:11:12
        //
        RUN SY:[1,2]PIP.SAV
        JUNQUE.OBJ/DE:NO
        ↑Z
```

If you will note, these commands will fail because the RSX emulator does not know how to deal with the commands SCALE, OLD, APPEND, and COMPILE. Therefore you should always specify CSPCOM when building under RSX or a similar run-time system.

(e) Using the RSX RTS with CSPCOM.

```
        RUN SY:[1,2]CSPCOM.TSK
        JUNQUE.OBJ/OBJ=JUNQUE,JUNK.APP
          .Z
        RUN SY:[1,2]TKB.TSK
        JUNQUE.TSK/FP=JUNQUE.OBJ,SY:[1,1]CSPCOM.OLB/LB
        /
        UNITS=12
        ASG=SY:5:6:7:8:9:10:11:12
        //
        RUN SY:[1,2]PIP.SAV
        JUNQUE.OBJ/DE:NO
        ↑Z              ♥
```

and you could be the next

Amcor Computer Corporation is one of the oldest mini-computer software vendors in the world. We have over a decade of experience in developing software solutions for a multitude of industries.

The people of Amcor are committed to achieving their goal of software excellence. Responsiveness to the requirements of the industries served by Amcor's New Generation Software Products is one of the essential elements that assures our continued success as a leading producer of software.

Amcor offers a full range of Accounting and Business Control application software products, as well as a comprehensive Data Base Management System.

If you would like more information on how you can join our ever-growing list of satisfied customers, just clip the coupon below and drop it in the mail, or give us a call. We will be happy to send you free information immediately.

please send more information on: RSP1

□ Accounts Receivable
□ Accounts Payable
□ Payroll
□ Inventory Control
□ Sales Analysis
□ AMBASE/DBMS
□ AMFACS/G/L Financial Management
□ Order Processing/Billing
□ Amcor's "Bundled System Discount"

company name ____________________

street ____________________ city ____________________

state ____________________ zip ____________ phone number ____________________

contact ____________________

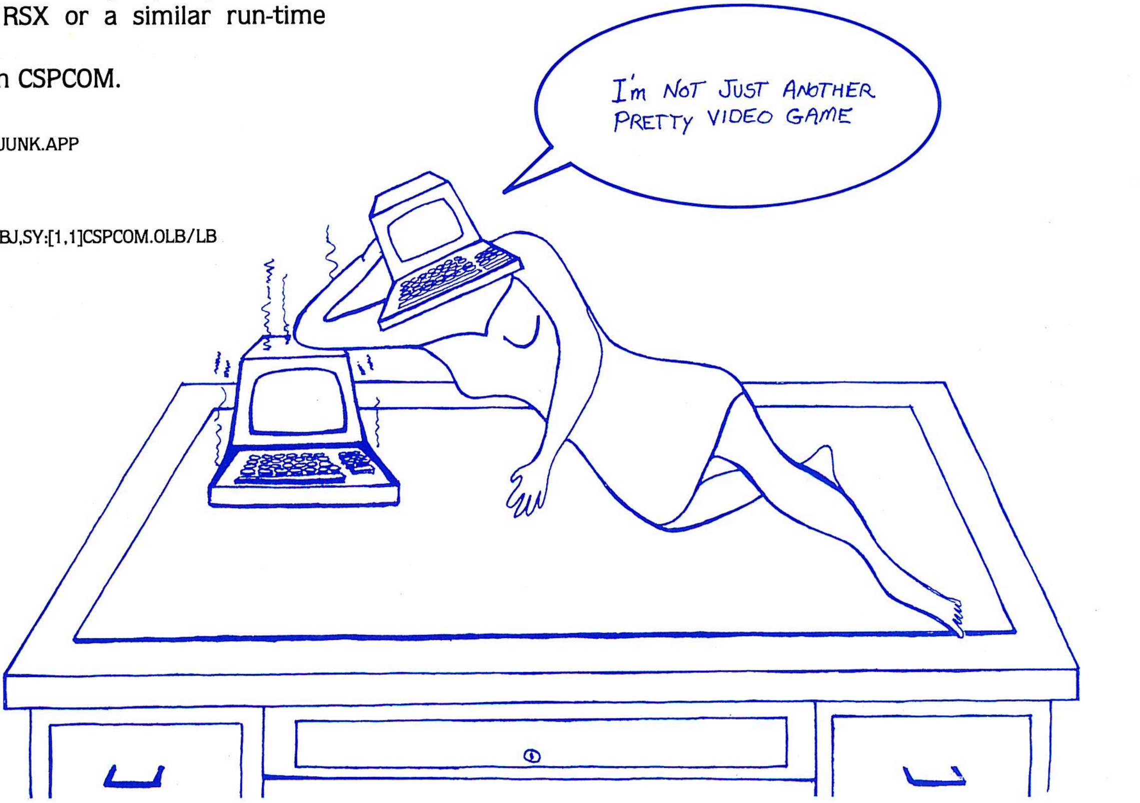
____________________ computer type and model ____________________ language

____________________ operating system

amcor computer corp.
a Kaneb company

Headquarters: Louisville, KY Regional Offices: Nationwide Telex 204122

CIRCLE 3 ON READER CARD



D.Benoit 82

ENABLE COMPATIBILITY WITH NON-DEC PERIPHERALS

By Ken Fleming, Multi-List/McGraw-Hill

In August of 1981, we installed the Able ENABLE "Memory Expander" and one megabyte of Mostec memory on an 11/45 with System Industries' RM05 look-alike drives (S.I. 9400 controller with CDC 9766 drives). We decided to take this approach because (a) we already owned the 11/45, and (b) we are in the process of switching to VAX 11/780's, so we did not wish to buy another PDP 11/70. The 11/45 is a very fast machine, but is limited in memory. We reasoned that with enough memory the problem of job swapping could be reduced to acceptable proportions.

We, the steering committee par excellence, had sold management on the vast improvement in terminal response that users would see (due to less job swapping) when we expanded from 256K DEC memory to 1 megabyte Mostec. The morning after installing Enable, we were forced to report that everything went well, but because of an as yet undefined "Glitch", we were still operating at our original 256K with the Enable installed!

Defining that "Glitch" became the challenge of the day — for too many days. The Enable device ran with RP04's on an RH11 controller with no problem. However, when we substituted the S.I. drives for the RP04's, we could not get past the memory map section of INIT.SYS (no message-system hung). Further investigation revealed that by not using the software patch that turns on the extended memory mapping, the Enable device worked fine with the S.I. drives.

System Industries' only answer was that the problem must be in the "other" device. Able's response was immediate. Les Wellington asked if he could come to our site and try to fix the problem for us. The next night Les, Joe Burdec, and Wayne Needer arrived armed with scopes, logic analyzers, revised boards, soldering guns, and spare parts galore. They worked all night with Bob Kelly (our inhouse electronics wizard) and myself to try and fix the problem. Unfortunately we still had not defined the problem by morning.

The next day I called System Industries again, this time to request an S.I. 9400 controller for Able to test with their Enable device. The response from System Industries was far from adequate. Les Wellington was also pursuing getting an S.I. 9400 controller on a loan basis. Two weeks went by with no response from S.I.

Finally S.I. agreed to send their best technician (not an engineer) to our site to check things out. Up to this time the only person at S.I. who appeared the least bit interested in our problem was Dick Mann. When the technician could not define the problem, we were forced to start calling higher S.I. management in an attempt to get some action. Able was doing everything they could without the S.I. 9400 controller. In fact, Les had discovered that their device would work with various third party controllers. S.I. seemed to be the only problem.

Finally, after applying constant pressure on S.I., Les Wellington was invited to Sunnyvale to work on S.I. equipment at S.I.'s expense. This was an excellent idea and Les agreed at once. However, by now it had taken a month to get S.I. to escalate beyond a "Gee, that's too bad" attitude.

As perserverance and curiosity are our long suit at Multi-List/McGraw-Hill, this author had finally prevailed and solved the impasse in the following manner.

The Enable device may be installed with up to four megabytes of memory, but it cannot address more than 256K bytes without a patch to INIT.SYS and the SIL. The Enable worked just fine on the 256K; but as soon as we patched INIT and the SIL, we could not bring up the system. This would immediately make one suspect the software patches. Joe Burdec assured me that it was not the patches, citing the fact that they were the same patches installed on every other system, and the only problems that they had encountered had been with S.I. equipment. This satisfied me for awhile, but I am responsible for (among other things) Sysgens, installation of new software, and patching.

One of the things that I had done recently was install a special INIT.SYS from S.I. to allow the CDC 9766's to run as large RM03's. This puzzled me — so I did some investigation. By comparing the INIT.SYS V7.0-07 and S.I.'s INIT, I discovered significant differences in DSK, ROOT, COPY,and BOOT. I talked to Dick Mann at S.I. and he assured me that there should be no conflict with the Able software patch because they should be different areas in the code.

By now, weeks had elapsed and I was more and more inclined to look toward software. I compared INIT V7.0-08, INIT V7.0-07, and S.I.'s INIT. The differences between INIT V7.0-08 and INIT V7.0-07 were insignificant. However, the differences between the two standard DEC INITs and S.I.'s INIT were numerous. Then I noticed S.I.'s INIT always asked for cluster size. Somewhere I had read about this being a bug in a very old INIT.SYS.

By now I was convinced that the problem was a conflict between Software Technique patches and S.I.'s patched INIT. So late one night I changed the S.I. drives from RM03 emulation to the RM05 emulation, mounted the new pack with DEC RM05 software with the Software Technique patches, and, lo and behold, everything worked. RSTS recognized all of our megabyte of memory. We have been running now for three months with no problems with the Enable device or S.I. drives.

When I inquired of the S.I. field tech the reason we were running in RM03 emulation, I was told it was because that was the way he was trained to do it. No one at S.I. could tell me the reason for this. The overall impression from dealing with S.I. was lack of field support training, both in software and hardware.

On a more positive note, since resolving this one major "Glitch", we have had no problems with either the S.I. drives or the Enable.

For all you hardware types, the Enable fits in an SPC hex slot. All DMA devices should be in front of the Enable board and the memory goes behind. This means that the Enable will normally be the last device on the bus. One item of interest is that you don't use a bus terminator with the Enable. Be sure you make this clear to your field service tech to avoid grief. Bob Kelly actually put a sign in the expansion box.

Provisions are made for you to piggy-back your present 18 bit address memory behind the Enable and 22 bit addressable memory; however, a separate SPC backplane is required. ABLE says you can go up to 4 meg, but we only have 1¼ megabyte; 1 meg of Mostek 8015 memory and ¼ meg of DEC MS 11-LD.

The S.I. interface also goes in one HEX slot; however, if you buy the 9400 controller instead of the 6100 single board, you will need some rack space. The most important benefit of the 9400 over the 6100 is the dual porting option which, with S.I. switch panel will allow up to four CPU to address up to 32 disk drives. The reader should take great care in deciding which CPU can write to which disk drive, since the disk map on disk and the disk map in memory won't match on all the CPUs at the same time. This feature could be of great value to a shop for backup purposes.

BACmac can do it all!

BAC into RTS / BAC into MAC / BAC into BAS

BACmac is a unique software tool, running under RSTS/E, which provides the following conversions:

■ translation from Basic-Plus "compiled" back to Basic-Plus source code (only the comments will be missing)

■ translation from Basic-Plus into Macro source code, which compiled under RSTS runs faster than Basic-Plus

■ translation from Basic-Plus into Macro source code which may be compiled under RSTS for execution under RT11 — a migration facility

■ translation from Basic-Plus into a RUN-TIME-SYSTEM. Now you can write an RTS in Basic-Plus. The ideal solution to memory thrashing due to "multi-copy" applications programs.

RSTS/E, RT11, Macro-11 and Basic-Plus are trademarks of Digital Equipment Corporation.

Please write for more information TELECOM

Telecom Computer Systems, Inc.
P.O. Box 03285
Portland, Oregon 97203
503/286-5122

CIRCLE 47 ON READER CARD

MORE NOTES ON LITERALS AND STRINGS IN BASIC-PLUS-2

By Brad Smith, Allied Data, Olympia, WA

The author has worked on PDP-11's for 5 years in several languages. He now specializes in the design and optimization of Basic-Plus-2 application systems.

In a previous article (RSTS Professional, December 1981), I explained the basic ways in which space for literals is allocated in Basic-Plus-2. Here is some additional information on ways to reduce the space and time required by a BP2 program.

One feature of the BP2 compiler which can be of importance is that concatenation of string literals is done at compile time. For instance,

```
A$ = "A" + "B"
```

produces the same object code as

```
A$ = "AB".
```

In addition, CHR$ functions with literal arguments are treated as literals: they are evaluated at compile time and can be concatenated with other literals at that time. This can help significantly in reducing the space and time required for printing. To use a simple example,

```
PRINT CHR$ (13%); CHR$ (10%);
```

requires 11 words to store the instructions plus a total of 12 bytes for the two literals. Concatenating them,

```
PRINT CHR$ (13%) + CHR$ (10%);
```

reduces the instruction space to 7 words and the data space to 6 bytes, and also reduces the execution time. Another example of the ways in which this compile-time concatenation can be utilized is in a keyboard input subroutine which returns a different value depending on the delimiter entered by the user. This can be done by writing something like

```
F% = POS(CR + LF + CHR$(27%) + CHR$(4%) + FF, D$, 1%)
```

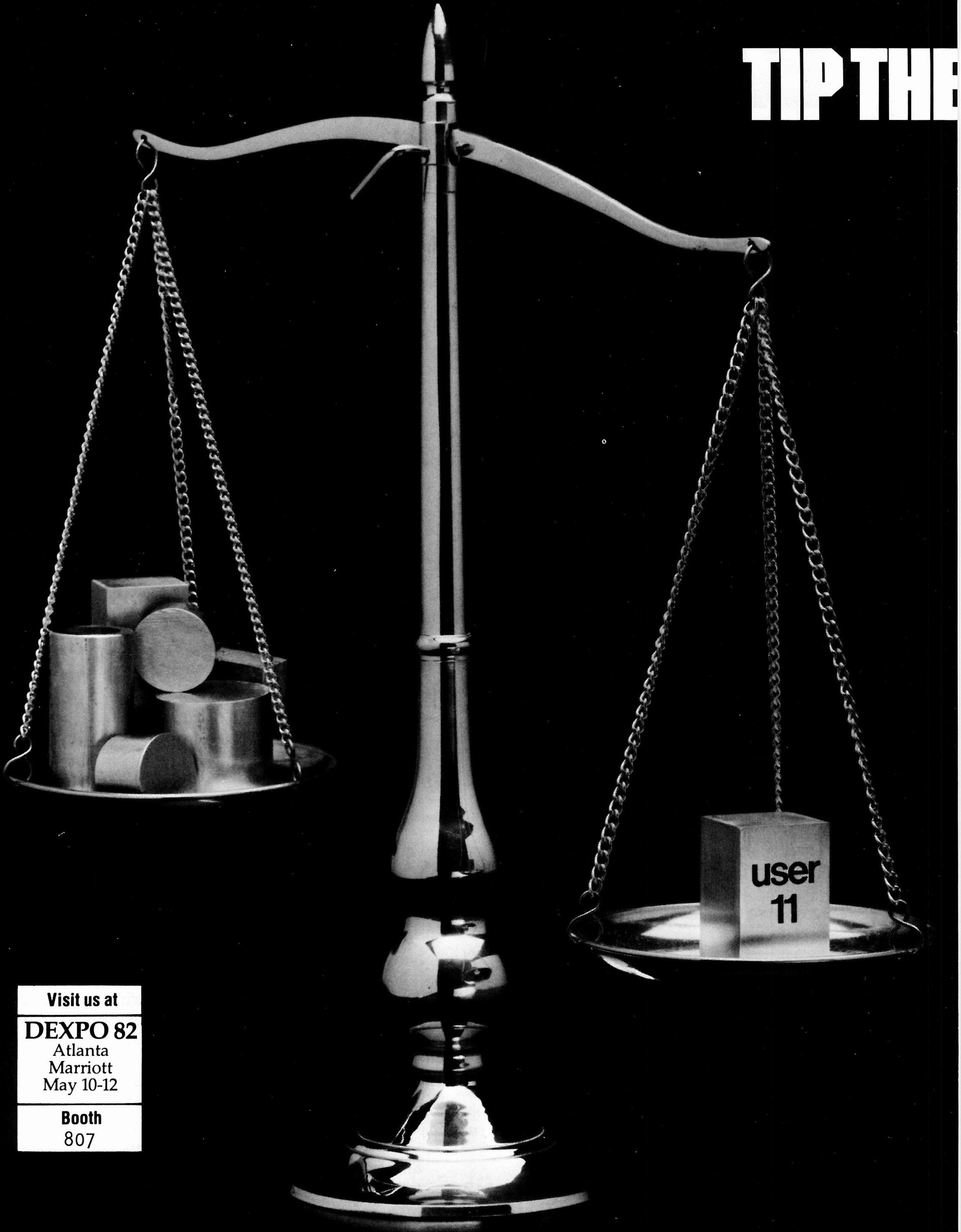
where D$ is the delimiter entered by the user. Being aware of this feature enables the programmer to avoid the "expense" of storing the individual characters as elements in an array or concatenating the characters and storing the result in a variable to be used in the above expression — neither of those approaches is as efficient.

The evaluation of literal expressions applies also to numeric expressions, but only to a limited extent. The compiler has problems with the precedence of operators. In such a case, it will go as far as it can in simplifying the expression. Consider the following examples of integer expressions and how they are expressed in object code:

```
5%*6% / 2%            = 15%
30% / (2%+3%)         =  6%
5%*6% / (2%+3%)       = 30% / 5%
30% / 2% + 3%-2%      = 16%
15% + (3%-2%)         = 16%
```

Enclosing 5%*6% in parentheses has no effect; however, note that the use of parentheses in the next-to-last expression, although not affecting the run-time result of the expression, does increase the space required to store it and the time to evaluate it.

TIP THE



Visit us at

DEXPO 82
Atlanta
Marriott
May 10-12

Booth
807

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

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

PRODUCTIVITY... A MATTER OF TIME.

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

PERFORMANCE... SIMPLY INCREDIBLE.

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

RELIABILITY... A PROVEN FACT.

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

SECURITY... MORE THAN RSTS.

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

STANDARDIZATION... A BYPRODUCT.

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

ADAPTABILITY... NO PROBLEM.

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

DOCUMENTATION... GOOD AND PLENTIFUL.

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

TRAINING...ALL KINDS.

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

FEATURES...ON AND ON.

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

NCCS
USER
•11•

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

DEC and RSTS are registered trademarks of Digital Equipment Corporation.

© Copyright NCCS

CIRCLE 30 ON READER CARD

VERSION 2.1 NOW AVAILABLE:

QUE-11 — V2.1

ONE JOB SPOOLER FOR RSTS/E CONTROLS ALL SPOOLING

REPLACES:

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

QUE-11:

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

For more information contact:

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

CIRCLE 11 ON READER CARD

One use of this feature is to make a program more readable — instead of

```
I% AND 53%
```

a programmer can write

```
I% AND (1% + 4% + 16% + 32%)
```

or `I% AND (1% OR 4% OR 16% OR 32%).`

Lest you hope for too much from the compiler, I should point out that the SPACE$, STRING$, ASCII, LEN, LEFT$, RIGHT$, MID$ and SEG$ functions with literal arguments are evaluated at run time.

A couple of further notes on strings:

In order to examine the first character of a string, it is slightly faster (at least, on an 11/70 with FPP — this could vary in a different environment) and occupies the same space to say

```
CHR$ (ASCII(A1))
```

instead of

```
LEFT$ (A$,1%).
```

When comparing the first character of a string to a literal, it occupies less space (7 words instead of 8) to say

```
IF ASCII(A$) = ASCII ("A")
```

instead of

```
IF LEFT$(A1,1%) = "A".
```

It is still better (only 5 words of instructions, and 6 bytes less data due to the elimination of the string literal) to say

```
IF ASCII(A$) = 65%.
```

It may appear that use of these techniques could not effect significant results, but when you need another 5 miles per hour out of a program that appears to be at its speed limit, a few seemingly minor changes can add up to a noticeable improvement. ♥

IT'S 3:15 PM MONDAY

Tired of writing depreciation journals in 3,5;GL, your third assistant bookeeper just discovered the joys of 4,0;PAY.

He's on his way from the bank to the airport.

LOTS OF LUCK!

LOCK-11

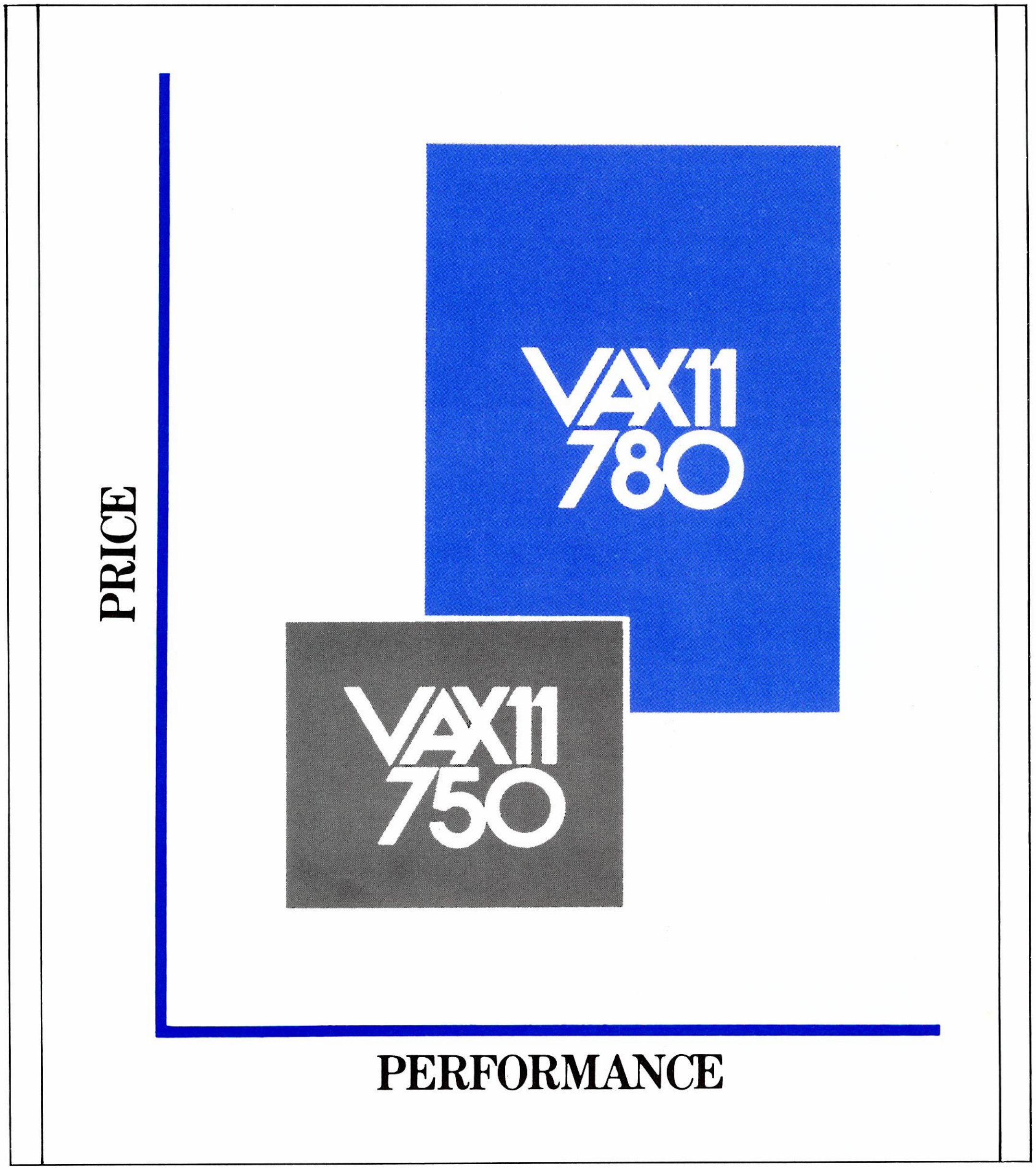
©NDD

SPD on Page 60

CIRCLE 80 ON READER CARD

The VAX-SCENE

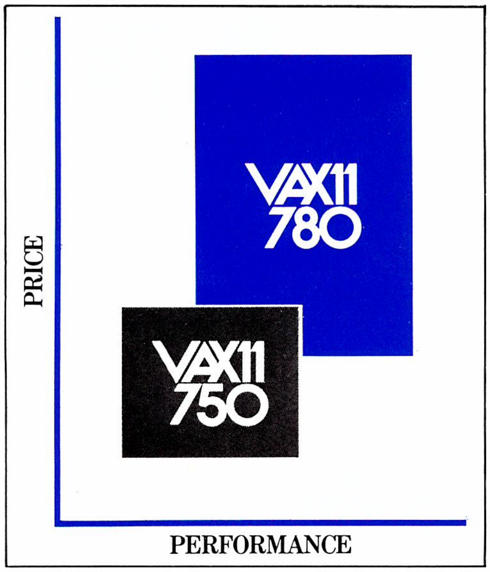
Number 7 (RSTS PROFESSIONAL, Vol. 4, No. 2) April 1982

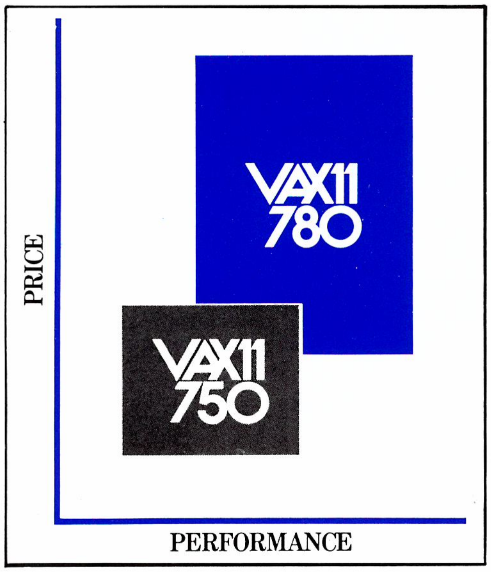


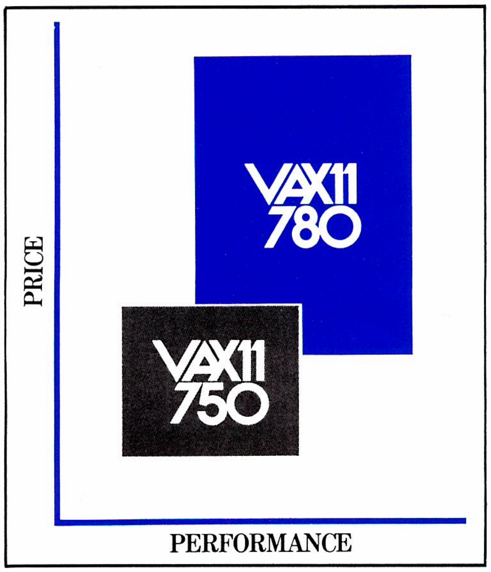
INSIDE:

□ Learning VAX Macro for Fun & Profit

□ Replacing RSTS SYS Calls with VAX/VMS System Services







LEARNING VAX MACRO FOR FUN & PROFIT

By Bob "MACRO MAN" Meyer

I had been doing some RSTS macro consulting for a small firm in New England (IE Systems, Newmarket, NH.). The project seemed to go quite well, and a few phone calls later I was asked if I'd like to get involved in a VAX project. 'A VAX project? Me?' I asked. 'Well, I'm willing to learn' I told them. That combined with a reduction in price landed my first VAX gig. It's been going on for almost a month now, with most of my time spent learning the ways of DCL, the assembler & linker, the instruction set, and monitor calls. The project so far has been rather interesting, for a guy that knows a fair amount of RSTS, and has done some Macro work under RSX-11M, so I though I'd share some of the adventure of converting a RSTS Macro program over to VAX land. In the next few issues I'll touch on some of the basic I/O calls to VMS, later pointing out some of the more interesting ones.

Please remember, I'm not a VAX-man (yet). These articles are for the purpose of showing others how to do some simple things under VMS. Please forgive any errors found; I'll try to be as accurate as possible.

Of all the things that impressed me most, I must first stand and RAVE about the Help command. The help system is so elaborate, that in most cases where a question arose, about ANY area of the VAX, I could usually get some direction, if not the complete answer, by using the help command.

Well done, DEC.

Assembling and linking the small test programs I was using was quite fast unless you tried to use RMS. Small programs that assemble in around 13 seconds would jump to about 1:20 if you-know-who was called in. . . too bad.

The command file processor is also outragous; it's an interperter in inteself, and lends a very helpful hand with a minimum of effort to learn the basics of it's use.

Next we'll talk about some of the simple I/O calls.

The basic I/O interface (at least from MY point of view) is VERY similar to that of RSX-11M. A channel is assigned to a device or file, and I/O requests are Queued to that channel. As in 11M, control can be returned to the user program as soon as the request is queued, and the program interupted when it completes, making for some pretty clever programming if desired. However, being quite new at all this, I opted to take the more conventional route, and wait for my I/O to finish before doing anything else. The following directive can be used for most I/O needs:

```
$QIOW_S  CHAN=TTCHAN,FUNC=#IO$_WRITEVBLK,-
             P1=BUF,P2=SIZ
```

(note that parameters to macros can be passed in any order)

Where:

$QIOW__S is the call 'Que I/O request & wait for completion'

CHAN is the channel # to do the I/O to as returned by the $ASSIGN directive (that's next)

FUNC is the function we're interested in; in this case 'write virtual block'

P1 parameter 1 is the address of the buffer to be written,

P2 P2 is the length of the buffer

This call will Que the I/O request, and return control to the user program when the I/O is complete.

Before this call can be executed successfully, the $ASSIGN directive must be issued to connect a path to the current terminal:

$ASSIGN__S DEVNAM = TT,CHAN = TTCHAN

Where TT is the text descriptor of the device to be opened (see example), and TTCHAN is a word which will hold the channel number returned by $ASSIGN.

A few other things in the example which may need explaning are:

The .ASCID directive puts the specified bytes in memory (like .ASCII in Macro-11), but preceedes the data with a string descriptor consisting of the string length, some descriptor information, and a position-independant pointer to the string. This is required by the $ASSIGN call to access the user terminal.

The .ENTRY directive. This is an assembler directive which sets up the entry point for the program, and a mask to save specified registers on program startup.

The instruction:

BLBS r0,<label>

can be used to watch for errors after executing monitor calls. This is because VMS places a STATUS code in R0 after each call. The instruction reads 'branch if low bit set (in R0) to <label>'. The branch will occur if the previous call completed successfully. If not, the RETurn instruction will be executed, bouncing control back to VMS, who, upon seeing a bad STATUS code, will print the respective error message on the user terminal.

If all is well and we're at the end of the program, the RETURN instruction will act as an EXIT directive, and just return control to VMS.

The sample program should work if keyed in. If time permits, I'll try to show some more calls next issue.

Bye for now.

```
.title      bob
.ident      /1.0/

tt:         .ascid   /_TTB5:/                   ;my terminal number
ttchan:     .word                               ;place for tt channel as returned
                                                ;by $ASSIGN
msg:        .ascii   /Hello from VAX-land!/<10><13>
msglen      =.-msg                              ;length of the message

.entry      bob,^M<r2>                          ;entry point of program
            $assign_s
                     devnam=tt,chan=ttchan      ;assign a channel to the current terminal
            blbs     r0,10$                     ;assign worked
            ret                                 ;assign failed; return to monitor

10$:        $qiow_s  chan=ttchan,-
                     func=#io$_writevblk,-
                     p1=msg,p2=#msglen          ;write the message to the terminal
            ret                                 ;return to the monitor

            .end     bob
```

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

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

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

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

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

ROSS/V is available from:

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

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

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

CIRCLE 67 ON READER CARD

REPLACING RSTS SYS CALLS WITH VAX/VMS SYSTEM SERVICES

A Few VMS Conversion Notes

By Bob Stanley, Computer Methods Corporation

INTRODUCTION

"So, you're thinking of converting from RSTS to a VAX? Well, I've heard the VAX is a nice machine; big, powerful, fast. But what about all of those RSTS dependent features that I've heard the VAX can't emulate?"

"How about things like direct CRT cursor addressing? Or echo control mode? Or programmable wildcard directory lookups? The VAX just can't handle those types of business application features that RSTS performs so well."

Does that conversation sound familiar? Have those types of questions and concerns turned you off to the VAX? Well, to the surprise and delight of many, there are solutions to these problems. This article takes a first hand look at how to make your brand new 32-bit supermini look just like RSTS. By the way, the rumor that the original title of this article was "Turning Unbearable Pain Into Extra Income" is just not true!

The conversion factors described below are from an actual RSTS to VAX conversion done for a client of Computer Methods Corporation that is currently running a 50 job RSTS system that tracks and manipulates export orders. As is typical of most installations , many programs were written that take advantage of RSTS dependent features and are, therefore, not easily convertible. Several external functions were written and placed in an object library that provided the programmers with substitute methods of performing these RSTS dependent functions. The basic building blocks for all of the functions that I will be discussing are the VMS system services.

SYSTEM SERVICES

System services are the VAX version of RSTS sys calls. While sys calls are cryptic, unwieldly, difficult to understand and even more difficult to use, system services are all of this and more! Actually, system services are more straightforward and easier to use because they follow the standard VMS calling procedures. They are invoked similar to a user defined function (E% = SYS$ASSIGN), they take a list of parameters, and they return a status code as their value.

VMS maintains a very long list of internal integer status codes that can be referenced within a program via the EXTERNAL INTEGER CONSTANT statement. These codes range from VAX BASIC error codes (BAS$__CANFINFIL meaning can't find file or account) to RMS status codes (RMS$__FNF meaning file not found) to system service status codes (SS$__NOPRIV meaning insufficient privilege). Any system service return status can be tested against these status codes (IF E% = SS$__NOPRIV in the above example) to test for expected errors or a normal successful status (SS$__NORMAL).

A program that is going to call a system service must first declare the system service and any external constants (status codes) via the EXTERNAL statement. Example 1 is an example program that calls the system service SYS$BRDCST which broadcasts a message to a specified terminal. This and all of the other system services are described in detail in the SYSTEM SERVICES REFERENCE MANUAL.

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

10       ! SYSTEM SERVICE EXAMPLE PROGRAM &

20       EXTERNAL INTEGER FUNCTION SYS$BRDCST &
\        EXTERNAL INTEGER CONSTANT SS$_NORMAL &

30       BRD.MESS$ = 'THIS IS A TEST MESSAGE' &
\        RECEIVING.TERMINAL$ = 'TTA6:' &
\        E% = SYS$BRDCST (BRD.MESS$,RECEIVING.TERMINAL$) &
\        PRINT 'ERROR IN MESSAGE SEND' &
                  IF E% <> SS$_NORMAL &

40       END &


                  Example 1

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

ECHO CONTROL

VMS does not handle opening a terminal in mode 8 (echo control mode). This mode is used to define specific fields (with specific lengths) that should be input from and displayed at specific positions on the terminal screen.

A typical application of this type would be the need to perform a data entry function via a predefined input screen format or to display control information while allowing an operator to move about the screen and enter selected fields of data.

While VMS does not perform echo control mode in the same fashion as RSTS, it does allow a program to do direct QIO's to any physical device including the keyboard. A special form of a QIO called 'read with prompt' enables a program to effectively perform controlled field input.

TERMINAL QIO'S

The first step in performing QIO's to any device is to assign that device to a specific channel (this is different from opening a file on a channel). This is done via the system service SYS$ASSIGN. Example 2 shows an external integer function that accepts a keyboard specification (TT on the VAX rather than KB:) and returns both an assigned channel number and a terminal type (VT52, VT100, etc.). A user supplied external function TERM__TYPE is called to provide the terminal type (this uses the system service SYS$GETDEV).

Once a channel has been assigned to the keyboard, the system service SYS$QIOW can be used to perform I/O to the terminal. A QIOW is an I/O with a wait for the device to respond. Several different functions can be performed via

SYS$QIOW. The one we are interested in is read with prompt. This is specified by passing IO$__READPROMPT (another external constant) to the system service.

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

10      FUNCTION INTEGER TERM_ASSIGN (TERMINAL.ID$, &
                                      TERMINAL.CHANNEL%, &
                                      TERMINAL.TYPE$) &

20      EXTERNAL INTEGER FUNCTION SYS$ASSIGN, &
                                  TERM_TYPE &
\       E% = SYS$ASSIGN (TERMINAL.ID$,TERMINAL.CHANNEL%) &
\       E% = TERM_TYPE (TERMINAL.ID$,TERMINAL.TYPE$) &

32767   TERM_ASSIGN = E% &
\       FUNCTIONEND &

                    Example 2

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

Example 3 shows an example of a QIOW using the IO$__READPROMPT function. The argument list allows you to specify the channel number assigned to the terminal, the function to be performed, a string to receive the data read, the length of that string, a string that determines which characters should be terminators (we'll talk about that in a minute), a prompt string, and the length of that prompt string. Other parameters are allowed and can be found in the SYSTEM SERVICES REFERENCE MANUAL and the I/O USERS GUIDE.

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

10      ! SAMPLE QIOW WITH READPROMPT &
        EXTERNAL INTEGER FUNCTION SYS$QIOW &
\       EXTERNAL INTEGER CONSTANT SS$_NORMAL, &
                                  IO$_READPROMPT &
\       READ.FUNCTION% = IO$_READPROMPT &
\       INPUT.BUF.LEN% = 50% &
\       READ.PROMPT$ = 'Enter field - ' &
\       PROMPT.LEN% = LEN(READ.PROMPT$) &

20      E% = SYS$QIOW (TERM.CHANNEL%,   ! FROM SYS$ASSIGN &
                       READ.FUNCTION% BY VALUE,,,, &
                       INPUT.FIELD$ BY REF, &
                       INPUT.BUF.LEN% BY VALUE,, &
                       TERMINATOR.MASK BY REF, &
                       READ.PROMPT$ BY REF, &
                       PROMPT.LEN% BY VALUE) &
\       PRINT 'ERROR IN SYSTEM SERVICE' &
             IF E% <> SS$_NORMAL &

32767   END &

                  Example 3

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

By passing these parameters to the system service, you can define a field of a specific length to be input from the terminal. The field will be returned to the program either when a terminator is typed or when the field is full. By combining this with a function to position the cursor at a specfic location (by printing the proper escape sequences just as under RSTS), controlled, esthetic data entry can be performed.

TERMINATORS

A program using the SYS$QIOW system service can specify its own set of terminator characters. This is done by turning on any of the low order 32 bits of a 64 bit quadword. Each of the bits 0—31 represent the ascii characters 0—31 (bit 3 is control C, 26 is control Z, etc.). Thus any of the ascii characters 0—31 can be specified as a terminator by turning on the appropriate bit.

The easiest way to do this is to start with a 32 bit longword set to zero and "OR" it with the proper power of 2 to turn that bit on. Thus, if A% is a long word with a value of zero, A% = A% OR 2**26% would cause a control Z to become a terminator. A% = A% OR 2**I% FOR I% = 0% TO 31% would cause all of the ascii characters 0—31 to become terminators.

As mentioned above, the low order 32 bits of a 64 bit quadword need to be set to determine the terminators. The simplest way to set up the quadword would be to define a map as follows:

MAP (TERMMASK) LONG TERMINATOR.MASK, LOW.ORDER.BITS

This defines two successive long words (a quadword). The variable LOW.ORDER.BITS can then be used as A% was in the above examples to set up the terminators.

The example program at the end of this article shows an external function that can be called to execute controlled terminal I/O. It was originally designed to facilitate the conversion of data entry screen formats but can be used by any application that needs to control the input of data by an operator.

WILDCARDS

A second function that does not lend itself easily to VMS is in-program wildcard directory lookups. Several conversion applications needed to send individual messages to a receiving program containing the names of each file in a specified directory. RSTS handled this problem via the wildcard directory sys call. VMS has no simple system service that will return file names given a wildcard specification. In fact, the VMS documentation's only reference to this function is in the back of the RMS REFERENCE GUIDE (chapter 13).

RMS does provide two system services that, with some considerable effort, perform wildcard lookups. (This RMS-32 facility, unlike its RSTS counterpart, allows wildcard characters in the directory specification as well as the file name.) In order to do this, however, one needs to understand and manipulate internal RMS file information structures; namely the FAB and NAM blocks.

FAB AND NAM BLOCKS

The FAB is an internal block of data that describes a particular file. The fields of the FAB contain information about the file such as the name of the file, the file's organization, its record format, space allocation, etc. The RMS REFERENCE GUIDE describes the FAB and gives a list of all the fields contained in the FAB. A map or common area can then be set up to define the fields of a FAB in your program.

The first word of caution, which is very important if you attempt to use the FAB block, is that the table in the manual that describes each field in the FAB is in alphabetical order. If you set up a map with the fields in the order listed in the table, your program will provide some interesting but highly inaccurate results. The second word of caution is that a field, right in the middle of the FAB block, is not listed in the table!

This little bit of information was discovered by expanding the FAB MACRO definition and looking at the offsets (listed in the table) and field lengths. Example 4 shows how to obtain the MACRO expansion listing.

The expanded MACRO listing contains information about the internal variables used by the MACRO. The FAB and NAM block offsets and the lengths of each of the fields appears on the first two pages. If reading an expanded MACRO listing is not your cup of tea, the examples below show how to incorporate the FAB and NAM blocks into your program.

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

  $ MAC/LIST=FABTEST TT
          .LIST BINARY
          $FAB
          $NAM
          .END
  ^Z
  $ PRINT FABTEST.LIS

             Example 4

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

Example 5 offers a Vax Basic callable function which performs wildcard directory lookups. It shows a map for the FAB block and the NAM block (described below) that have the fields in the proper order and with the proper lengths.

The NAM block contains supplementary information about a file such as device and directory information, expanded file name strings and wild card character context. Again, the manual does not provide enough information to accurately set up a NAM block map. Example 5 contains the complete NAM block layout.

SYS$PARSE

The first step in performing a wildcard directory lookup is the SYS$PARSE system service (described only in the RMS REFERENCE GUIDE). This service takes information provided in the FAB block, parses it, and allocates fields in the NAM block to store the wild card character context for subsequent searches. This service need only be called once in the case of iterative directory lookups.

In order to use SYS$PARSE, certain fields in the FAB and NAM blocks must be initialized. The external function in Example 5 performs the SYS$PARSE system service in the function FNSET.UP%. The FAB$C__BID and NAM$C__BID external integers must be provided to identify the FAB and NAM blocks. Also, the length of the FAB and NAM blocks must be placed in the FAB.BLN and NAM.BLN fields. The external constants FAB$C__BLN and NAM$C__BLN can be used for this purpose.

The remainder of the fields that need to be initialized can be found in the RMS REFERENCE GUIDE, chapter 13, pages 13-4 and 13-5.

SYS$SEARCH

Once the wildcard specification has been parsed, the directory specified in the NAM block can be searched via the SYS$SEARCH system service. SYS$SEARCH will return one file name at a time and can be called iteratively until the status code RMS$__NMF (nor more files) is returned. The service maintains its own internal wildcard count (in NAM.WCC) so that it never gets lost in the middle of the directory.

The sample external function in Example 5 shows the SYS$SEARCH system service being called after the SYS$PARSE service. The parameter flag FIRST.TIME.FLAG% is used to determine wether or not the SYS$PARSE system service should be performed. If it is performed each time the function is called, only the first file name in the directory would ever be returned.

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

10        EXTERNAL INTEGER FUNCTION WILD_LOOK (WILDCARD.SPEC$, &
                                               FILE.NAME$, &
                                               FIRST.TIME.FLAG$) &
\         EXTERNAL INTEGER CONSTANT FAB$C_BID, NAM$C_BID, &
                                    FAB$C_BLN, NAM$C_BLN, &
                                    FAB$V_OFP, RMS$_NORMAL &
\         EXTERNAL INTEGER FUNCTION SYS$PARSE, SYS$SEARCH &

20        MAP (FAB)    STRING FAB.BID = 1%, STRING FAB.BLN = 1%, &
                       WORD   FAB.IFI,      LONG   FAB.FOP, &
                       LONG   FAB.STS,      LONG   FAB.STV, &
                       LONG   FAB.ALQ,      WORD   FAB.DEQ, &
                       STRING FAB.FAC = 1%, STRING FAB.SHR = 1%, &
                       LONG   FAB.CTX,      STRING FAB.RTV = 1%, &
                       STRING FAB.ORG = 1%, STRING FAB.RAT = 1%, &
                       STRING FAB.RFM = 1%, LONG   FAB.JNL, &
                       LONG   FAB.XAB,      LONG   FAB.NAM, &
                       LONG   FAB.FNA,      LONG   FAB.DNA, &
                       STRING FAB.FNS = 1%, STRING FAB.DNS = 1%, &
                       WORD   FAB.MRS,      LONG   FAB.MRN, &
                       WORD   FAB.BLS,      STRING FAB.BKS = 1%, &
                       STRING FAB.FSZ = 1%, LONG   FAB.DEV, &
                       LONG   FAB.SDC,      LONG   FAB.SDC1, &
                       LONG   FAB.SDC2 &
\         MAP (FAB)    FAB.BLOCK = 80% &

30        MAP (NAM)    STRING NAM.BID = 1%, STRING NAM.BLN = 1%, &
                       STRING NAM.RSS = 1%, STRING NAM.RSL = 1%, &
                       LONG   NAM.RSA,      WORD   NAM.RSA1, &
                       STRING NAM.ESS = 1%, STRING NAM.ESL = 1%, &
                       LONG   NAM.ESA,      LONG   NAM.RLF, &
                       STRING NAM.DVI =16%, WORD   NAM.FID, &
                       WORD   NAM.FID1,     WORD   NAM.FID2, &
                       WORD   NAM.DID,      WORD   NAM,DID1, &
                       WORD   NAM.DID2,     LONG   NAM.WCC, &
                       LONG   NAM.FNB &

40        MAP (WILDLOOK_FILENAME)  ORIG.FILE.NAME$ = 63%, &
                                   NAM.FILE.NAME$  = 63%, &
                                   RESULT.FILE.NAME$ = 63% &

50        EXEC% = FNSET.UP% IF FIRST.TIME.FLAG% &
\         GOTO 32767 IF E% <> RMS$_NORMAL AND &
               FIRST.TIME.FLAG% &
\         E% = SYS$SEARCH (FAB.BLOCK BY REF,,) &
\         FILE.NAME$ = LEFT(RESULT.FILE.NAME$,ASCII(NAM.RSL)) &
               IF E% = RMS$_NORMAL &
\         GOTO 32767 &

100       DEF FNSET.UP% &
\         NAM.FILE.NAME$, RESULT.FILE.NAME$ = SPACE$(63%) &
\         FAB.BID = CHR$(FAB$C_BID) &
\         NAM.BID = CHR$(NAM$C_BID) &
\         ORIG.FILE.NAME$ = WILDCARD.SPEC$ &
\         FAB.FNA = LOC(ORIG.FILE.NAME$) &
\         FAB.FNS = CHR$(LEN(WILDCARD.SPEC$)) &
\         FAB.IFI = 0% &
\         FAB.NAM = LOC(NAM.BID) &
\         FAB.BLN = CHR$(FAB$C_BLN) &
\         NAM.BLN = CHR$(NAM$C_BLN) &
\         NAM.ESA = LOC(NAM.FILE.NAME$) &
\         NAM.ESS = CHR$(LEN(NAM.FILE.NAME$)) &
\         NAM.RSA = LOC(RESULT.FILE.NAME$)) &
\         NAM.RSS = CHR$(LEN(RESULT.FILE.NAME$)) &

110       E% = SYS$PARSE(FAB.BLOCK BY REF,,) &
\         FNEND &

32767     WILD_LOOK = E% &
\         FUNCTIONEND &

                    Example 5

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

CONCLUSION

While some of the functions that many of our programs have come to depend on under RSTS do not exist under VMS, there are ways to emulate these functions on the VAX. As a whole, VMS provides many more functions that make writing those programs that use tricky system techniques much easier.

```
1000    !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! &
        !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! &
        !                                                                &
        !       PHO_KEYIN - DuPont Photo Products Key Input Function     &
        !                                                                &
        !                                                                &
        !       AUTHOR : R.S.STANLEY (Computer Methods Corp.)            &
        !       DATE   : 10-DEC-81                                       &
        !                                                                &
        !                                                                &
        !       This function provides any program written in a VAX     &
        ! native mode language with capabilities similar to RSTS cursor &
        ! and echo control modes. It allows the program to specify      &
        ! cursor positioning, prompting text, field length, etc. and    &
        ! returns to the program the operator entered string. Its       &
        ! original purpose was to provide input capabilities similar to &
        ! the Dupont sponsered TAM data entry package. It can, however, &
        ! be used by any program that needs to retrieve data from the   &
        ! operator in a controlled, esthetic fashion.                   &
        !                                                               &
        !       The function parameters are:                            &
        !                                                               &
        !               PROMPT.TEXT$        - Any prompting string      &
        !                                     that should appear with   &
        !                                     the input field           &
        !                                                               &
        !               PAINT.CHARACTER$    - The character that will   &
        !                                     define the data position  &
        !                                     prior to data entry       &
        !                                                               &
        !               DEFAULT.ANSWER$     - A string that should be   &
        !                                     returned in place of a    &
        !                                     blank field               &
        !                                                               &
        !               PROMPT.COLUMN%      - Starting screen column    &
        !                                     for the prompting text    &
        !                                                               &
        !               PROMPT.LINE%        - Starting screen line for  &
        !                                     the prompting text        &
        !                                                               &
        !               INPUT.COLUMN%       - Starting screen column    &
        !                                     for the input area        &
        !                                                               &
        !               INPUT.LINE%         - Starting screen line for  &
        !                                     the input area            &
        !                                                               &
        !               FIELD.LENGTH%       - Predefined length of the  &
        !                                     input field. Either a     &
        !                                     terminator or a full      &
        !                                     field will end data       &
        !                                     entry for this field      &
        !                                                               &
        !               TERMINAL.CHANNEL%   - The keyboard ('TT') from  &
        !                                     which data entry can be   &
        !                                     be expected.              &
        !                                                               &
        !               TERMINAL.TYPE$      - The type of terminal to   &
        !                                     be used (VT52, FT1, etc)  &
        !                                                               &
        !               DONT.ECHO%          - Flag to indicate whether  &
        !                                     or not the data entered   &
        !                                     should be echoed (true    &
        !                                     or false only)            &
        !                                                               &
        !               CLEAR.SCREEN%       - Flag to indicate whether  &
        !                                     or not the screen should  &
        !                                     be cleared prior to data  &
        !                                     entry (true or false)     &
        !                                                               &
        !               KEYPUNCH.MODE%      - Flag to indicate whether  &
        !                                     to break on field full    &
        !                                     or insist on an operator  &
        !                                     typed terminator.         &
        !                                                               &
        !               INPUT.DATA$         - The operator entered      &
        !                                     data being returned to    &
        !                                     the calling program.      &
        !                                                               &
        !                                                               &
        !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! &
        !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! &
&
&
        FUNCTION INTEGER PHO_KEYIN ( PROMPT.TEXT$, &
                                     PAINT.CHARACTER$, &
                                     DEFAULT.ANSWER$, &
                                     PROMPT.COLUMN%, &
                                     PROMPT.LINE%, &
                                     INPUT.COLUMN%, &
                                     INPUT.LINE%, &
                                     FIELD.LENGTH%, &
                                     TERMINAL.CHANNEL%, &
                                     TERMINAL.TYPE$, &
                                     DONT.ECHO%, &
                                     CLEAR.SCREEN%,  &
                                     KEYPUNCH.MODE%, &
                                     INPUT.DATA$ ) &

1010    EXTERNAL INTEGER FUNCTION    SYS$QIOW, &
                                     PHO_DISPLAY, &
                                     PHO_HOMEUP, &
                                     PHO_CLRSCN, &
                                     PHO_CURPOS &
\       EXTERNAL INTEGER CONSTANT    IO$_READPROMPT, &
                                     IO$M_TRMNOECHO, &
                                     IO$M_NOECHO, &
                                     SS$_NORMAL &
\       MAP (TRMASK) LONG            TERMINATOR.MASK, &
                                     LOW.ORDER.MASK &

1020    ON ERROR GOTO 19000 &
\       E% = CTRLC &
\       TRUE% = (1% = 1%) &
\       FALSE% = (1% = 0%) &
\       E% = PHO_HOMEUP (TERMINAL.TYPE$,HOME.UP$) &
\       E% = PHO_CLRSCN (TERMINAL.TYPE$,CLEAR.SCREEN$) &
\       PRINT HOME.UP$ + CLEAR.SCREEN$ IF CLEAR.SCREEN% &
\       E% = PHO_CURPOS (TERMINAL.TYPE$,INPUT.COLUMN%,INPUT.LINE%,INPUT.POS$) &
\       E% = PHO_CURPOS (TERMINAL.TYPE$,PROMPT.COLUMN%,PROMPT.LINE%, &
                PROMPT.POS$) &
\       LF$ = CHR$(10%) &
\       CR$ = CHR$(13%) &
\       CTLZ$ = CHR$(26%) &
\       CTLC$ = CHR$(3%) &
\       ESC$ = CHR$(27%) &
\       LOW.ORDER.MASK = 0% &
\       LOW.ORDER.MASK = LOW.ORDER.MASK OR 2%**I% FOR I% = 0% TO 30% &
\       LOW.ORDER.MASK = LOW.ORDER.MASK XOR 2%**2I% &
\       READ.FUNCTION% = IO$_READPROMPT OR IO$M_TRMNOECHO &
\       READ.FUNCTION% = READ.FUNCTION% OR IO$M_NOECHO &
                IF DONT.ECHO% &
\       INPUT.FIELD$ = SPACE$(FIELD.LENGTH%) &
\       INPUT.BUF.LEN% = LEN(INPUT.FIELD$) &
\       PAINT.STRING$ = STRING$(FIELD.LENGTH%,ASCII(PAINT.CHARACTER$)) &
\       PAINT.STRING$ = STRING$(FIELD.LENGTH%,ASCII(' ')) &
                IF PAINT.CHARACTER$ = '' &
\       BACK.SPACES$ = STRING$(FIELD.LENGTH%,8%) &
\       READ.PROMPT$ = PAINT.STRING$ + BACK.SPACES$ &
\       READ.PROMPT$ = INPUT.POS$ + READ.PROMPT$ &
                UNLESS INPUT.COLUMN% = 0% AND INPUT.LINE% = 0% &
\       READ.PROMPT$ = PROMPT.TEXT$ + READ.PROMPT$ &
                UNLESS PROMPT.TEXT$ = '' &
\       READ.PROMPT$ = PROMPT.POS$ + READ.PROMPT$ &
                UNLESS PROMPT.LINE% = 0% AND PROMPT.COLUMN% = 0% &
\       PROMPT.LEN% = LEN(READ.PROMPT$) &

1030    ERROR% = SYS$QIOW ( , TERMINAL.CHANNEL% BY VALUE, &
                              READ.FUNCTION% BY VALUE,,,, &
                              INPUT.FIELD$ BY REF, &
                              INPUT.BUF.LEN% BY VALUE,, &
                              TERMINATOR.MASK BY REF, &
                              READ.PROMPT$ BY REF, &
                              PROMPT.LEN% BY VALUE) &

1040    GOTO 32767 IF ERROR% <> SS$_NORMAL &
\       ESC.LOC% = INSTR(1%,INPUT.FIELD$,ESC$) &
\       LF.LOC% = INSTR(1%,INPUT.FIELD$,LF$) &
\       CTLZ.LOC% = INSTR(1%,INPUT.FIELD$,CTLZ$) &
\       FOUND.ONE% = FALSE% &
\       FOR I% = 1% UNTIL I% = 31% OR FOUND.ONE% &
\               TERMINATOR.LOC% = INSTR(1%,INPUT.FIELD$,CHR$(I%)) &
\               FOUND.ONE% = TRUE% IF TERMINATOR.LOC% <> 0% &
\       NEXT I% &
\       IF ESC.LOC% <> 0% THEN &
            INPUT.FIELD$ = CHR$(27%) + SPACE$(FIELD.LENGTH%-1%) &
\           GOTO 1070 &
        ELSE &
            IF LF.LOC% <> 0% THEN &
                INPUT.FIELD$ = LF$ + SPACE$(FIELD.LENGTH%-1%) &
\               GOTO 1070 &
            ELSE &
                IF CTLZ.LOC% <> 0% THEN &
                    INPUT.FIELD$ = CTLZ$ + SPACE$(FIELD.LENGTH%-1%) &
\                   GOTO 1070 &
                ELSE &
                    IF TERMINATOR.LOC% <> 0% THEN &
                        INPUT.FIELD$ = LEFT(INPUT.FIELD$, &
                            TERMINATOR.LOC%-1%) + ' ' + &
                            RIGHT(INPUT.FIELD$,TERMINATOR.LOC%+1%) &

1050    IF INPUT.FIELD$ = '' THEN &
                INPUT.FIELD$ = DEFAULT.ANSWER$ &
\               E% = PHO_DISPLAY (TERMINAL.TYPE$, &
                                  TERMINAL.CHANNEL%, &
                                  INPUT.COLUMN%, &
                                  INPUT.LINE%, &
                                  INPUT.FIELD$) &

1060    INPUT.FIELD$ = LEFT(INPUT.FIELD$,FIELD.LENGTH%) &
\       INPUT.FIELD$ = INPUT.FIELD$ + ' ' &
                UNTIL LEN(INPUT.FIELD$) = FIELD.LENGTH% &

1070    INPUT.DATA$ = INPUT.FIELD$ &
\       GOTO 32767 &

19000   IF ERR = 28 THEN &
                INPUT.DATA$ = CTLC$ + SPACE$(FIELD.LENGTH%-1%) &
\               RESUME 32767 &

32767   PHO_KEYIN = ERROR% &
\       FUNCTIONEND &
```

IT'S 2:28 AM

The kid with his auto-dial MODEM
just found your "new" dial-in number
555-0112 on the 112th try.
He's in and you are out.

LOTS OF LUCK!

LOCK-11
©NDD 1

SPD on Page 60

CIRCLE 80 ON READER CARD

THE BASICS OF NETWORKING AND DIGITAL COMMUNICATION FOR THE SYSTEM MANAGER

By Michael H. Koplitz

Digital communication is used in all aspects of computing, from the asynchronous terminal to synchronous communication between CPUs. Networking involves the use of digital communication between several devices and CPUs. The objective of this article is to acquaint the RSTS/E System Manager to the methods and terminology of digital communication.

BASIC ELEMENTS OF COMMUNICATION

1. **Message** — a sequence of characters used to convey information or data.
2. **Transmission** — the act of sending a message between the sender and receiver.
3. **Sender** (transmitter) — a device which has a message to communicate.
4. **Receiver** — a device capable of receiving or accepting a message.
5. **Medium** (of transmission) — the way of getting the message from the sender to the receiver.
6. **Noise** — anything that interferes with the process of communication.
7. **Efficiency** — effective use of the communication channel.

TYPES OF TRANSMISSION

Parallel transmission — the medium of parallel transmission consists of one wire for each bit in a character plus an additional wire for a clock or strobe signal. The clock or strobe tells the receiver to read the character which is on the other wires. This type of transmission is good for high speed data transmission.

Serial transmission — the medium of serial transmission consists of a pair of wires, one wire to transmit data and one wire to act as a common signal ground. Bits are transmitted serially, one after the other. Most serial transmissions can be sent over telephone lines by using a modem. A modem is a device which converts a binary (digital) signal into an analog signal by modulation at the transmitter's end. The modem at the receiver's end demodulates the analog signal into a binary signal.

20 mA transmission — a technique used to transmit binary data along serial lines. This method transmits the binary data by turning a 20 mA (milli-amp) current on and off. The flow of current indicates a "1" bit and a "0" bit is indicated by stoping the flow of current. 20 mA transmissions can not use modems.

EIA transmission — a second technique used to transmit binary data along serial lines. This method transmits data by reversing the polarity of the voltage on a dc serial line. A positive voltage on the line communicates a "0" bit and a negative voltage communicates a "1" bit.

Voltage varying systems are more susceptible to noise. The EIA system is based on standards prepared by the Electronics Industry Association and includes the definition of modem control signals. Most modems manufactured in the United States are compatible with the EIA standard RS-232C.

CCITT transmission — a third technique used to transmit binary data along serial lines. CCITT is a voltage varying system based on standards prepared by the International Consultative Committee on Telephony under the auspices of the United Nations.

MODES OF TRANSMISSION

Simplex — communication can only occur in one direction on the wire pair.

Half-duplex (HDX) — communication can occur in either direction on the wire pair but only in one direction at a time.

Full-Duplex (FDX) — communication can occur in either direction on the wire pair at the same time.

ASYNCHRONOUS SERIAL TRANSMISSION

In asynchronous serial transmission, the sender transmits a character whenever a character is ready to be transmitted. Sometimes this type of transmission is called "Start/Stop" transmission. This is because a start bit is transmitted first, then the character, followed by a stop bit(s).

A line is said to be idle when no characters are being communicated. As soon as the receiver senses the start bit, the receiver starts a clock which measures bit times. The receiver then samples the next eight bits and places them into a register for transfer to memory. The next bit(s) is the stop bit, which must be a "1" bit. A stream of stop bits will indicate that the line is idle. Whenever a "0" (start) bit comes down the line the receiver would then start the clock.

This is not a very efficient way to communicate because at least two out of every ten bits serve as start and stop bits, which do not communicate data.

SYNCHRONOUS SERIAL TRANSMISSION

In synchronous communication and entire block of characters is sent at a time. Special synchronous characters are sent before and after each block to coordinate or synchronize both the sender and the receiver. There is not any need for start and stop bits since the entire block of characters is synchronized. Therefore the synchronous technique uses the line more efficiently than the asynchronous serial transmission.

SYNCHRONOUS PROTOCOL

Every protocol has the following functions: controlling data transfers, error checking and recovery, information coding, information transparency, line utilization, synchronization, communications facility transparency, and bootstrapping.

Controlling data transfers — there are three elements involved, formatting, control information and "handshaking" procedures. Control data and error checking information are contained in one block:

```
-----------------------------------------------------
!header or     !body or text field    !trailer or error!
!control field!                       !checking field  !
-----------------------------------------------------
```

The header contains addressing, block sequencing, control flags and acknowledgement information. The addressing is used for determining the destination of the data. Block sequencing ensures that no transmission is lost or duplicated. The control flags are used to indicate whether the transmission is data, control-only, first, intermediate or last block of the message. Control messages are used to determine who transmitted and who received the data. It is also used for the receiver to acknowledge the receipt of data and whether it is a good or bad transmission. This procedure of acknowledgement is refered to as handshaking.

Error checking and recovery — assures correct reception of data. Check bits are transmitted with the message which are used in verifying that the data transmitted is correct. The check bits are commonly called block check characters (BCC) which make up the trailer field of the transmission block.

Methods to check errors:

Vertical Redundancy Checking (VRC) — parity checking is done on each character as the data is received. The parity can be even or odd.

Longitudinal Redundancy Checking (LRC) — uses exclusive OR logic to check the entire block for errors after the block is received. After each transmission the receiving station normally replies with a positive (ACK) acknowledgement or with a negative (NAK) acknowledgement.

Cyclic Redundancy Checking (CRC) — also checks the block after the entire block has been received. This method uses polynomial division of the data stream by a CRC polynomial to check for errors. This is a very complex method to check for errors.

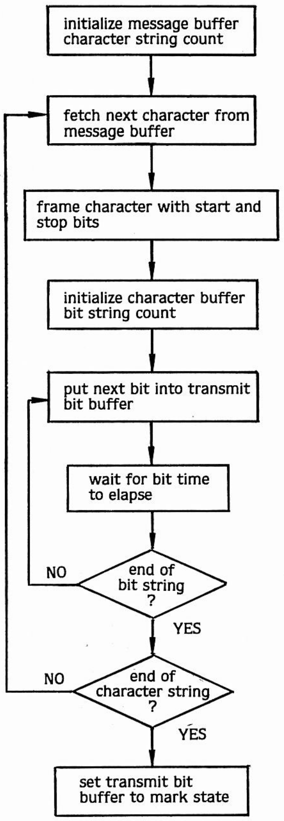
Information Coding

ASCII (American Standard Code for Information Interchange) — this code was introduced by the USA Standards Institute. It is a seven bit-plus-parity code. There are several codes in the scheme which have been set aside for communication control. The parity can be either even or odd.

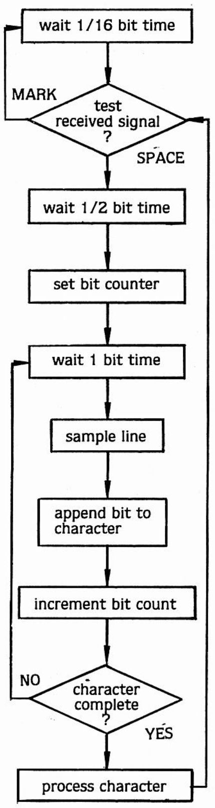
Data Interchange Code — is a variation of the ASCII code, some of the printing characters of the ASCII code have been replaced by non-printing control characters. The parity must be odd.

Other types of code are **Extended Binary Coded Decimal Interchange Code** (EBCDIC), **Baudot Code** (a five bit code used on the old teleprinters), **Four of Eight Code, IBM Punch Card Hollerith Code, Binary Coded Decimal Code** (BCD) and the six bit **Transcode.**

ASYCHRONOUS TRANSMISSION LOGIC
(figure A)


initialize message buffer character string count
fetch next character from message buffer
frame character with start and stop bits
initialize character buffer bit string count
put next bit into transmit bit buffer
wait for bit time to elapse
NO
end of bit string ?
YES
NO
end of character string ?
YES
set transmit bit buffer to mark state

ASYCHRONOUS RECEPTION LOGIC
(figure B)



RSTS SITE MANAGEMENT
AND APPLICATION
DEVELOPMENT TOOLS

□ APC
an automatic password changer that creates meaningful six-character passwords and updates the ACCT.SYS file, allows selective changing of passwords and produces three informative reports.

□ ENCRYPTION ROUTINES
a site security feature which encodes ASCII characters and can be incorporated into any application where sensitive data is processed. Also exists as a stand alone program for encoding and decoding entire files.

□ M/APS
a menu/authorization processor and application security system that controls user access to menus and applications programs. Uses DEC's VT series CRTs.

□ STANDARD SUBROUTINE LIBRARY
callable macro-11 routines that perform screen and terminal I/O, cursor positioning and many other necessary program functions, including data conversions.

□ SOURCE/FILE CROSS-REFERENCE (XREF)
XREF provides cross-reference listings which detail the relationship between source files, callable routines, data files and task images.

□ VT100 ACCOUNTING CALCULATOR
a multi-function calculator designed for user's of DEC's VT100 CRTs. Options and features beyond the capabilities of the normal Accountant's calculator.

McHUGH, FREEMAN & ASSOC., INC.
1135 Legion Drive
Elm Grove, Wisconsin 53122
(414) 784-8250

CIRCLE 57 ON READER CARD

Software Product Description

Product Name: LOCK-11 Version 2.2

Description:

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

- Absolute control of system access by keyboard. Manager may limit any keyboard to certain accounts or groups of accounts and control time as well as day of week access.
- Password knowledge is no longer carte blanche system access. System detects unauthorized use of passwords. Privileged passwords don't work on non-privileged keyboards. Non-privileged passwords work only on specified keyboards.
- Real time system surveillance. Manager specifies a list of alarm keyboards which log all infractions and probes as they happen. Opser is not required.
- Auto-login (with or without password) and chain with specified core common contents by KB.
- Manager may establish special priority/burst settings by KB. Manager may establish default output protection code, @ assignment and up to three specific user logicals for each KB. Default RTS is also selectable. All assignments are made at log-in.
- Manager specifies a list of console keyboards from which security file editor may operate.
- Manager may define a KB-specific access-denied message.
- Manager may specify number of retries before access-denied and number of access-denied messages before line disable. Hangup on access denied is optional. All above may be specified on a per-kb basis.
- A macro DYNPRI program is included which performs the following functions:
  - Users may be dispatched into ten separate priority queues, separately tunable on-line. Each queue has ten levels. Queues are selectable by KB.
  - Program detects hibernating jobs and announces the fact on ALARM keyboards. Privileged jobs hibernating cause extra loud and long alarms.
  - The program produces almost no load in operation and runs in 5K words.
  - Program will hold up to fourteen files open for performance purposes.

Minimum Hardware/Software Required:

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

Support: See License Agreement

Installation: User Installed

Ordering Information:

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

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

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

License Fee:

Single CPU license: $950.00. Annual maintenance at 12% of current list price.

Contact:

Dave Mallery
Nationwide Data Dialog
215—364-2800

CIRCLE 12 ON READER CARD

Information Transparency — It is important that a protocol be able to send binary data, floating point numbers, packed data, and machine language computer programs in the same format. Different methods are used to accomplish this and each different protocol does it in a different way.

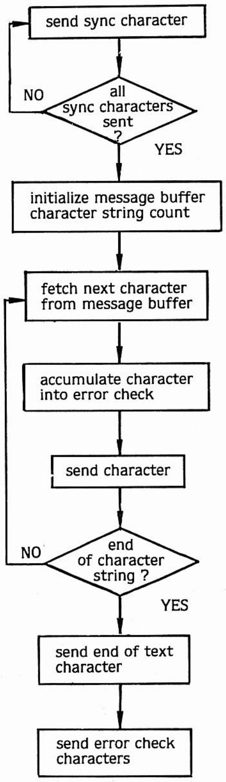
Line Utilization — is the attempt to make protocol utilize the communication channel to its fullest.

Synchronization — when transmitting synchronous data the sender and receiver must be synchronized for proper reception of the transmission. To get the receiver in phase a unique group of bits called a sychronization sequence precedes the transmission. The synchronization sequence should be such that the data stream can not reproduce it.

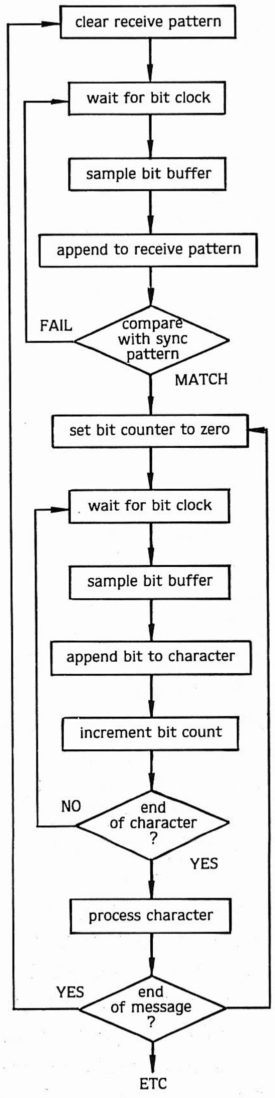
Communication facility transparency — the idea is to have any protocol run on any facility. This would be ideal but as of yet has not been done.

Bootstrapping — ability of down line loading a computer.

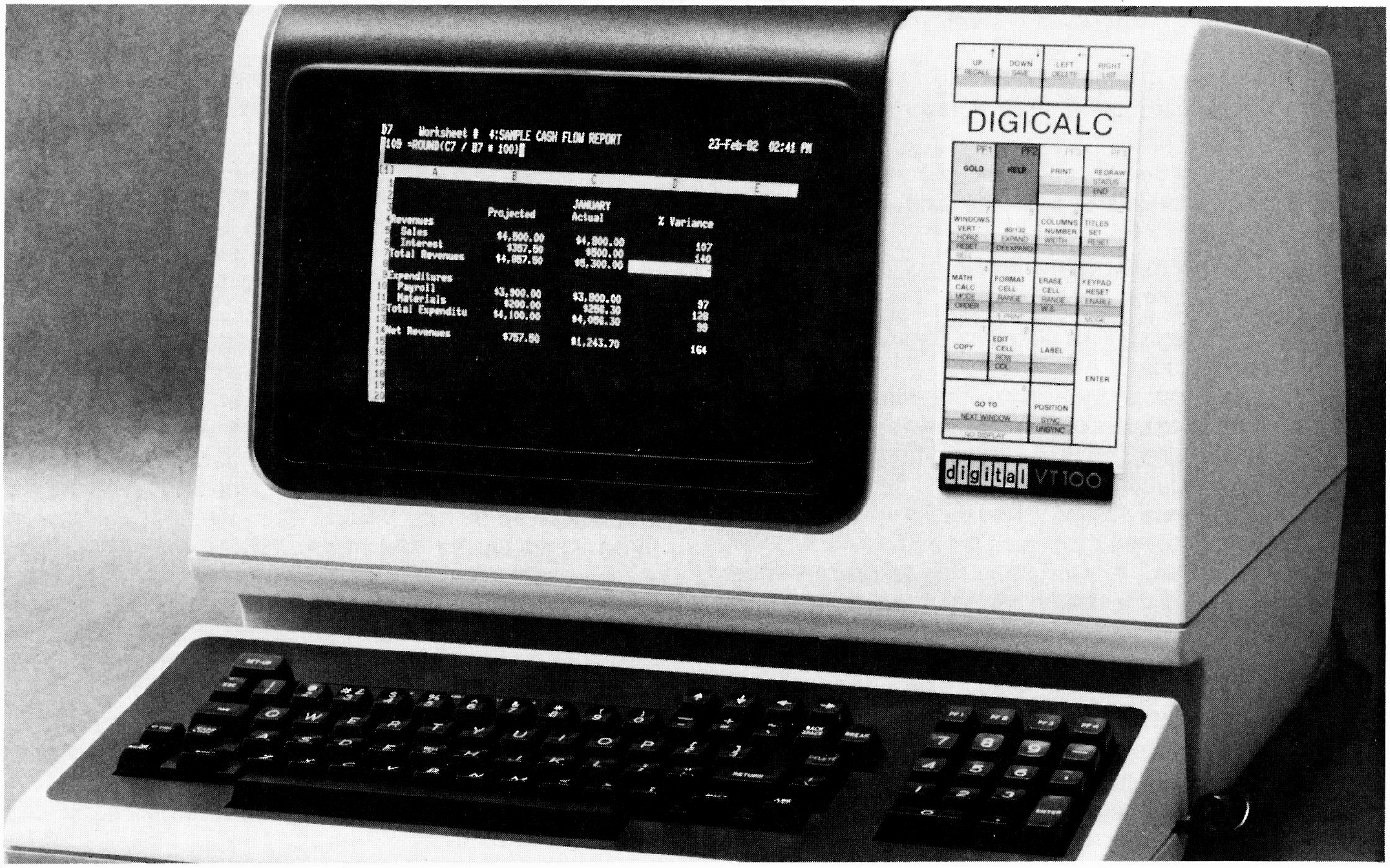
SYCHRONOUS TRANSMISSION LOGIC
(figure C)


send sync character
NO
all sync characters sent ?
YES
initialize message buffer character string count
fetch next character from message buffer
accumulate character into error check
send character
NO
end of character string ?
YES
send end of text character
send error check characters

SYCHRONOUS RECEPTION LOGIC
(figure D)



DIGICALC™ The total electronic spread sheet for DEC™ computer systems.



```
B7     Worksheet #  4:SAMPLE CASH FLOW REPORT
D109 =ROUND(C7 / B7 * 100)                                  23-Feb-82  02:41 PM
[1]        A              B              C              D              E
 1
 2
 3                                    JANUARY
 4Revenues               Projected    Actual
 5  Sales                                            % Variance
 6  Interest              $4,800.00    $4,800.00
 7Total Revenues            $357.50      $500.00           107
 8                        $4,857.50    $5,300.00           140
 9Expenditures
10  Payroll
11  Materials             $3,900.00    $3,800.00
12Total Expenditu           $200.00      $256.30            97
13                        $4,100.00    $4,056.30           128
14Net Revenues                                              98
15                          $757.50    $1,243.70
16                                                         164
17
18
19
20
```

WHY SYSTEMS has developed the "big machine" software package for Executives, Accountants and Professionals who need their numbers fast and accurate.

DIGICALC®, designed exclusively for DEC equipment, with an amazing range of applications and one of the finest built-in automatic training procedures on the market. Prepare everything from budgets to multi-year forecasts, change one value and DIGICALC will re-compute all of your results and give you a hardcopy too. The tabular worksheet display on the wide screen can be as simple or complex as you wish and has the capability to call on numerous mathematical functions. For an amazing display of DIGICALC's ease of operation and virtually limitless capabilities call WHY SYSTEMS. Runs on RSTS/E, VMS, and Tops – 10/20.

- ON-LINE HELP AND SELF TEACHING MODE
- TEN KEY NUMERIC DATA ENTRY
- EXTERNAL FILE INTERFACE
- WORKSHEET CONSOLIDATION
- "BOARDROOM QUALITY" REPORTS
- EXTENSIVE MATH FUNCTIONS
  ALGEBRAIC
  LOGICAL
  FUNCTIONAL
  SCIENTIFIC
  USER DEFINED FUNCTIONS
- SAVES AND RECALLS WORKSHEETS.

WHY

WHY SYSTEMS INCORPORATED
17130 Avondale Way, N.E.
Redmond, WA 98052

CALL TODAY (206) 881-2331 FOR FREE BROCHURE AND DIAL-UP DEMONSTRATION

CIRCLE 72 ON READER CARD

A WORD ABOUT THE AUTHOR . . .

Rudy Bazelmans is a Software Analyst at Sykes Datatronics Inc., where he designs and codes Language Processors.

THE ULTIMATE PUSH/PULL MACROS

By Rudy Bazelmans, Sykes Datatronics, Inc.

ABSTRACT

In Assembly Language Programming it is very common to utilize the stack for temporarily storing groups of variables. This paper presents a set of macros for easily manipulating the stack on a PDP-11. Some of the richness and power of the MACRO-11 assembly language is also demonstrated.

INTRODUCTION

When manipulating the stack in Macro-11 there are a number of inconveniences:

1. The instruction to push and pull items from the stack is awkward to write and a nuisance to remember.

```
MOV VALUE,-(SP)   ;PUSH
MOV (SP)+,VALUE   ;PULL
```

2. Only one item may be placed on the stack in each line of source code.
3. If you push a byte onto the stack you must remember to pull a byte off, otherwise you will pull a word off and you may unintentionally change a memory location.
4. After you have pushed values on the stack, you must remember to pull them back off in the reverse order.
5. Before exiting a subroutine you must remember which items are still on the stack so you can take them off.

An approach to solving these problems is through the use of macros. To my knowledge, macros have been used to solve items 1, 2, and 3 above. I am not aware of an existing solution to items 4 and 5.

The following is a group of macros which I have written to solve all five items most notably items 4 and 5. The explanation of how these macros work is broken into two parts. The first part will center around the concept of solving a subset of the problems mentioned above. The second part will describe the complete solution, which includes more features and error checking than the first part.

For those of you who are interested in using a set of macros with the above properties and are not concerned about the details of how they work, you can simply use the macros in figures 3B and 4B. All the information required to use these macros is included in figures 3A and 4A.

THE CONCEPT

The easiest way to simulate the action of a stack is through the use of another stack. That is my basic approach to solving these problems.

The first set of macros is shown in Figure 1A and 1B. You should take a moment and read the description included with them. These macros (along with the examples in figure 2A) are quite limited, but they do implement the basic idea of assembly time stacks.

There is a stack pointer in these macros called PSHCT$ which begins at zero and keeps a count of the number of items PUSHed on the stack. Remember, stacks are LIFOs, the Last item In is the First item Out. The initialization of this counter is in the user's program at line 2 of figure 2A. The counter is incremented whenever a new item is placed on the stack (line 32 of figure 1A). The counter is decremented again when the macro for that item is expanded (line 46 of figure 1A).

In order to place an item on the stack, you must first call the PUSH macro. Each argument in the group of arguments to PUSH is isolated one at a time (line 30 of figure 1A). Each argument is then moved onto the stack (line 31) and PSHCT$ is incremented to show that another value has been placed on the stack (line 32).

Lines 33-38 is where the items PUSHed are remembered for the PULL macro. PSHFL$ is used to indicate if the current argument is the first argument to the PUSH macro. If it is the first argument, PSHFL$=0 (line 29 of figure 1A). If it is not the first argument, PSHFL$=1. The setting of PSHFL$ is important to the PUSH$ macro and its significance will be discussed below.

There are three parameters passed to the PUSH$ macro: the name of the current argument being pushed on the stack, the ASCII equivalent of PSHCT$, and (if the current argument is not the first argument to PUSH) the ASCII equivalent of PSHCT$-1.

The PUSH$ macro (lines 42-50) defines a macro (lines 46-48) of the name PSH**name**$ where **name** is the current value of PSHCT$. The macro definition consists of three lines. The first line restores the value of the argument from the stack (line 45). PSHCT$ is decremented in the second line in order to indicate a change in the nesting level. In the third line, a check is made to see if the argument which was passed to PUSH$ is the first argument to the PUSH macro. If it is the first argument, then we have restored all the arguments in the group. Remember that when we restore the values from the stack we have to do it in the reverse order of the way we stored them on the stack. If the current argument to PUSH$ is not the first argument to the PUSH macro, then we should call the macro that is necessary to restore the next argument of the group (line 47).

At this point, we are only defining a macro to restore the arguments from the stack, we are not actually restoring them. The actual restoration will occur when the PULL macro calls the macro which we just defined. If the user calls the PUSH macro again this entire process will be repeated.

The next step is the retrieval of the data which was placed on the stack. The user calls the macro PULL which will pull the last group of values from the stack. This is extremely simple. First the current value of PSHCT$ is converted to it's ASCII equivalent (line 28 of figure 1B) in **name** and the macro PSH**name**$ is called (line 29). Remember that PSHCT$ has the value of the last argument placed on the stack.

A sample execution with the expanded code is shown in figure 2A. Figure 2B shows the state of the symbol table at line 5 of the sample execution.

```
 1         .SBTTL  PUSH    SIMPLIFIED PUSH MACRO
 2 ;******************************************************************
 3 ;*                                                                *
 4 ;* NAME:          PUSH    - PUSH a group of values on the stack   *
 5 ;*                                                                *
 6 ;* DESCRIPTION: This macro PUSHes groups of registers, vari-      *
 7 ;*                ables, locations and constants on the stack.    *
 8 ;*                They  are pushed left to right and retrieved    *
 9 ;*                in reverse order using the PULL macro.          *
10 ;*                                                                *
11 ;* CALL SEQ:      PUSH STATEMENT  ::=     PUSH <group>            *
12 ;*                group           ::=     arg | arg,group         *
13 ;*                                                                *
14 ;* INPUT:         arg     - A register,  variable,  location or   *
15 ;*                constant to be placed on the stack.   Only 16   *
16 ;*                bit values may be PUSHed.                       *
17 ;*                                                                *
18 ;* OUTPUT:        None                                            *
19 ;*                                                                *
20 ;* SIDE EFFECTS:The variables PSHFL$  and PSHCT$  are used as     *
21 ;*                well as macro names of the form PSHxx$ where    *
22 ;*                xx is a number between 0 and 77 octal.          *
23 ;*                                                                *
24 ;* AUTHOR:        RUDY BAZELMANS                 VERSION A        *
25 ;*                                                                *
26 ;******************************************************************
27
28         .MACRO   PUSH   ARGS
29            PSHFL$=0
30            .IRP   ARG,<ARGS>
31               MOV   ARG,-(SP)
32               PSHCT$=PSHCT$+1
33               .IF   EQ   PSHFL$
34                  PUSH$   ARG,\PSHCT$
35                  PSHFL$=1
36               .IFF
37                  PUSH$   ARG,\PSHCT$,\<PSHCT$-1>
38               .ENDC
39            .ENDR
40         .ENDM    PUSH
41
42         .MACRO   PUSH$   ARG,NAME,NEXT
43
44            .MACRO   PSH'NAME'$
45               MOV   (SP)+,ARG
46               PSHCT$=PSHCT$-1
47               .IIF   NB   <NEXT>,   PSH'NEXT'$
48            .ENDM    PSH'NAME'$
49
50         .ENDM   PUSH$
```

FIGURE 1A. Macros can be used to simulate assembly time stacks. This macro pushes items onto the stack.

THE ULTIMATE PUSH/PULL MACROS

The second group of macros (figures 3 and 4) show the PUSH and PULL macros in their final state. This version is more flexible than the first version and does more error checking. Read figures 3A and 4A for a complete description of how these macros should be used. Some of the differences between the first and second set of macros are described below:

There is more error checking being done in this new version. Line 2 of figure 3B shows a check to see that the user specified at least one argument in the call to the PUSH macro. Lines 5-7 show a check for a stack overflow. The limit is 63, this should be more than adequate. If you are PUSHing more than 63 levels deep on the stack, you're doing something wrong. Lines 3 and 4 of figure 4B show a check for an empty stack. If you want to PULL a value and you have never PUSHed anything on the stack, you should be specifying a destination on the PULL.

The manipulation of words and bytes is also supported in this improved set of macros. The default argument size is one word but if a byte is to be PUSHed, an apostrophe (') should be placed before the byte argument in the parameter list of the PUSH or PULL (line 3 of figure 5). The code to do this checking is in lines 9 and 30 of figure 3B and in line 13 of figure 4B. The unusual construct in lines 10 and 31 of figure 3B and line 14 of figure 4B is executed whenever the

```
 1         .SBTTL  PULL    SIMPLIFIED PULL MACRO
 2 ;*********************************************************************
 3 ;*                                                                   *
 4 ;* NAME:          PULL    - PULL a group of values from the stack.   *
 5 ;*                                                                   *
 6 ;* DESCRIPTION: This macro PULLs groups ot values from the stack.    *
 7 ;*                                                                   *
 8 ;*                All the elements of the last group of values PUSHed *
 9 ;*                on the stack are now PULLed off the stack.  All    *
10 ;*                the values are PULLed off in the reverse order of  *
11 ;*                the way they were PUSHed on.                       *
12 ;*                                                                   *
13 ;* CALL SEQ:      PULL                                               *
14 ;*                                                                   *
15 ;* INPUT:         None                                               *
16 ;*                                                                   *
17 ;* OUTPUT:        The last group of values are PULLed from the stack. *
18 ;*                                                                   *
19 ;* SIDE EFFECTS:The variables PSHFL$  and PSHCT$  are used as        *
20 ;*                well as macro names of the form PSHxx$ where       *
21 ;*                xx is a number between 0 and 77 octal.             *
22 ;*                                                                   *
23 ;* AUTHOR:        RUDY BAZELMANS                     VERSION A       *
24 ;*                                                                   *
25 ;*********************************************************************
26
27         .MACRO   PULL
28            .IRP   NAME,\PSHCT$
29               PSH'NAME'$
30            .ENDR
31         .ENDM    PULL
```

FIGURE 1B. This simple macro pulls items off the assembly time stack.

```
1         .SBTTL EXAMPLES
2         PSHCT$=0                            ;INIT STK PTR
3         PUSH     <ABC,DEF,GHI>
             MOV    ABC,-(SP)
             MOV    DEF,-(SP)
             MOV    GHI,-(SP)
4         PUSH     <JKL,R0>
             MOV    JKL,-(SP)
             MOV    R0,-(SP)
5
6         PULL
             MOV    (SP)+,R0
             MOV    (SP)+,JKL
7         PULL
             MOV    (SP)+,GHI
             MOV    (SP)+,DEF
             MOV    (SP)+,ABC
```

FIGURE 2A. This shows a sample execution of the simplified macros.

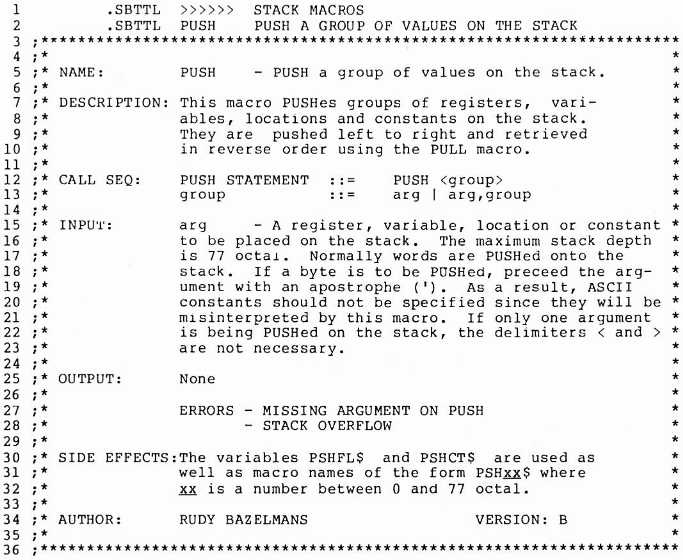
```
                               MACRO PSH1$
                        ____________________________
                        |                          |
                        |   MOV (SP)+,ABC          |<---
                        | PSHCT$=PSHCT$-1          |   |
                        |__________________________|   |
                                                       |
                               MACRO PSH2$             |
                        ____________________________   |
                        |                          |   |
                        |   MOVB (SP)+,DEF         |   |
                        | PSHCT$=PSHCT$-1          |<--|---
                        |      PSH1$               |----   |
                        |__________________________|       |
                                                           |
                               MACRO PSH3$                 |
                        ____________________________       |
                        |                          |       |
                        |   MOV (SP)+,GHI          |       |
                        | PSHCT$=PSHCT$-1          |       |
                        |      PSH2$               |-------|
                        |__________________________|

                               MACRO PSH4$
                        ____________________________
                        |                          |
                        |   MOVB (SP)+,JKL         |<---
                        | PSHCT$=PSHCT$-1          |   |
                        |__________________________|   |
                                                       |
                               MACRO PSH5$             |
  VARIABLE PSHCT$       ____________________________   |
 _________________      |                          |   |
 |               |      |   MOV (SP)+,R0           |   |
 |       5       |--->  | PSHCT$=PSHCT$-1          |   |
 |_______________|      |      PSH4$               |----
                        |__________________________|
```

FIGURE 2B. The state of the macros and variables at line 5 of figure 2A are shown above.

user specifies a byte argument. The byte arguments have a preceeding apostrophe which must be removed before the instruction is generated. This is accomplished by using the universal unary operator (^) with an apostrophe appended to the end of the argument. As a result, the argument (less the leading apostrophe) is passed to the .IRP loop.

The final macros contain several .MEXIT statements lines 16 and 45 of figure 3B and line 20 of figure 4B. These exist because the .IRPC loop which surrounds them only needs to be executed once to check for a leading apostrophe on each argument.

Figure 5 shows an enhanced sample execution for the final version of the PUSH/PULL macros.



```
 1         .SBTTL  >>>>>>  STACK MACROS
 2         .SBTTL  PUSH    PUSH A GROUP OF VALUES ON THE STACK
 3 ;*********************************************************************
 4 ;*                                                                   *
 5 ;* NAME:        PUSH    - PUSH a group of values on the stack.       *
 6 ;*                                                                   *
 7 ;* DESCRIPTION: This macro PUSHes groups of registers,  vari-        *
 8 ;*              ables, locations and constants on the stack.         *
 9 ;*              They are  pushed left to right and retrieved         *
10 ;*              in reverse order using the PULL macro.               *
11 ;*                                                                   *
12 ;* CALL SEQ:    PUSH STATEMENT  ::=    PUSH <group>                  *
13 ;*              group           ::=    arg | arg,group               *
14 ;*                                                                   *
15 ;* INPUT:       arg     - A register, variable, location or constant *
16 ;*              to be placed on the stack.  The maximum stack depth  *
17 ;*              is 77 octal.  Normally words are PUSHed onto the     *
18 ;*              stack.  If a byte is to be PUSHed, preceed the arg-  *
19 ;*              ument with an apostrophe (').  As a result, ASCII    *
20 ;*              constants should not be specified since they will be *
21 ;*              misinterpreted by this macro.  If only one argument  *
22 ;*              is being PUSHed on the stack, the delimiters < and > *
23 ;*              are not necessary.                                   *
24 ;*                                                                   *
25 ;* OUTPUT:      None                                                 *
26 ;*                                                                   *
27 ;*              ERRORS - MISSING ARGUMENT ON PUSH                    *
28 ;*                     - STACK OVERFLOW                              *
29 ;*                                                                   *
30 ;* SIDE EFFECTS:The variables PSHFL$  and PSHCT$  are used as        *
31 ;*              well as macro names of the form PSHxx$ where         *
32 ;*              xx is a number between 0 and 77 octal.               *
33 ;*                                                                   *
34 ;* AUTHOR:      RUDY BAZELMANS                  VERSION: B           *
35 ;*                                                                   *
36 ;*********************************************************************
```

FIGURE 3A. This documentation describes the PUSH macro.

DEC
RSTS/E USERS

From one of the pioneers in commercial data processing using RSTS. Off the shelf software ready for immediate delivery. Completely interactive. Extensively documented. Fully supported. Ideal for OEM's, service bureaus or end users. Cost effective solutions including:

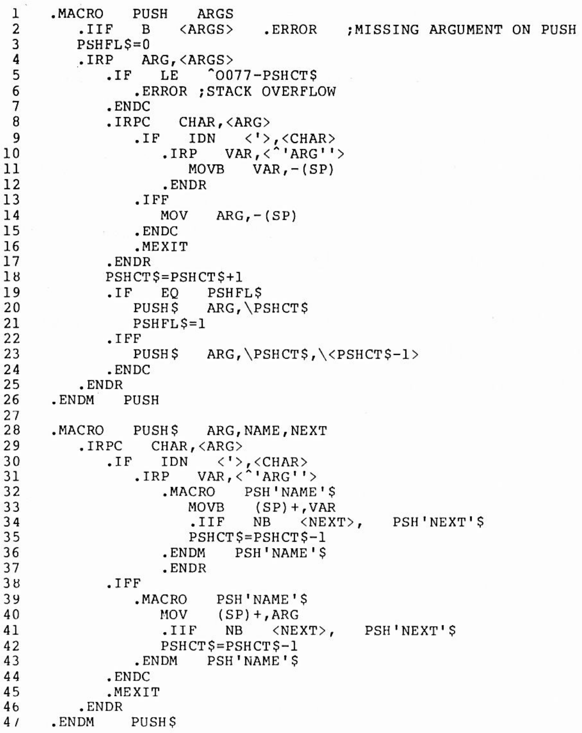
- ACCOUNTS PAYABLE
- GENERAL LEDGER
- FINANCIAL REPORTING
- ACCOUNTS RECEIVABLE
- PAYROLL
- FIXED ASSETS

For complete details, contact us at:

Plycom services, inc.

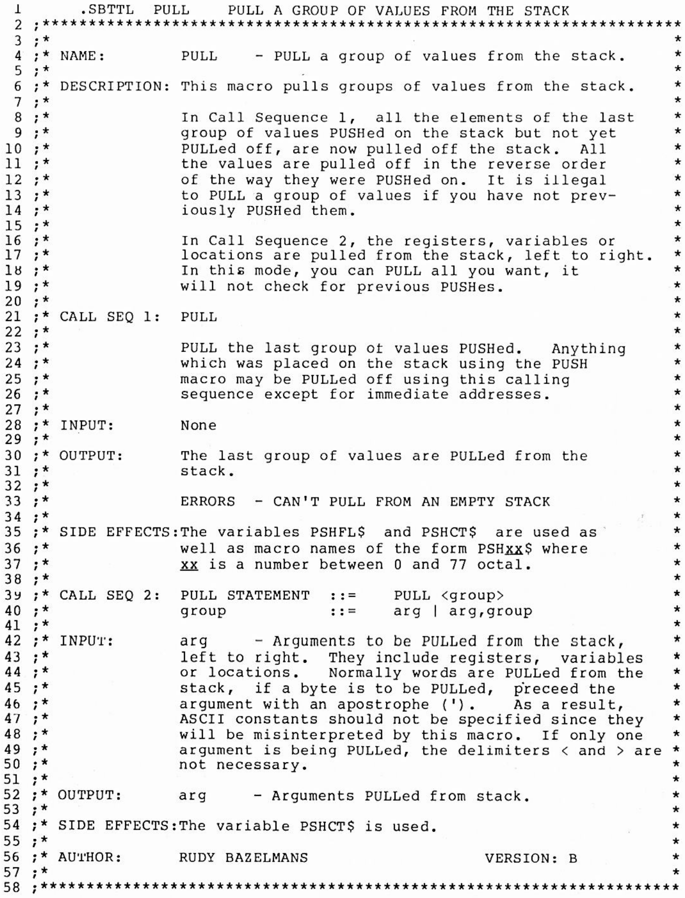
P.O. Box 160
Plymouth, IN 46563
(219) 935-5121

CIRCLE 42 ON READER CARD



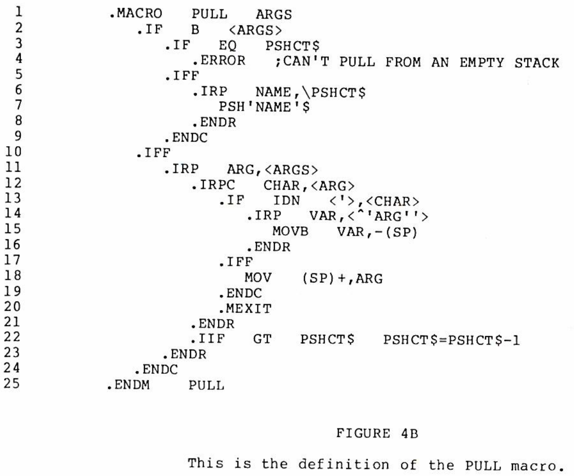
```
 1   .MACRO   PUSH   ARGS
 2      .IIF   B   <ARGS>   .ERROR   ;MISSING ARGUMENT ON PUSH
 3      PSHFL$=0
 4      .IRP   ARG,<ARGS>
 5         .IF   LE   ^O077-PSHCT$
 6            .ERROR ;STACK OVERFLOW
 7         .ENDC
 8         .IRPC   CHAR,<ARG>
 9            .IF   IDN   <'>,<CHAR>
10               .IRP   VAR,<^'ARG''>
11                  MOVB   VAR,-(SP)
12               .ENDR
13            .IFF
14               MOV   ARG,-(SP)
15            .ENDC
16            .MEXIT
17         .ENDR
18         PSHCT$=PSHCT$+1
19         .IF   EQ   PSHFL$
20            PUSH$   ARG,\PSHCT$
21            PSHFL$=1
22         .IFF
23            PUSH$   ARG,\PSHCT$,\<PSHCT$-1>
24         .ENDC
25      .ENDR
26   .ENDM   PUSH
27
28   .MACRO   PUSH$   ARG,NAME,NEXT
29      .IRPC   CHAR,<ARG>
30         .IF   IDN   <'>,<CHAR>
31            .IRP   VAR,<^'ARG''>
32               .MACRO   PSH'NAME'$
33                  MOVB   (SP)+,VAR
34                  .IIF   NB   <NEXT>,   PSH'NEXT'$
35                  PSHCT$=PSHCT$-1
36               .ENDM   PSH'NAME'$
37               .ENDR
38         .IFF
39            .MACRO   PSH'NAME'$
40               MOV   (SP)+,ARG
41               .IIF   NB   <NEXT>,   PSH'NEXT'$
42               PSHCT$=PSHCT$-1
43            .ENDM   PSH'NAME'$
44         .ENDC
45         .MEXIT
46      .ENDR
47   .ENDM   PUSH$
```

FIGURE 3B. Here is the definition of the PUSH macro.



```
 1       .SBTTL  PULL    PULL A GROUP OF VALUES FROM THE STACK
 2 ;*********************************************************************
 3 ;*                                                                   *
 4 ;* NAME:        PULL    - PULL a group of values from the stack.     *
 5 ;*                                                                   *
 6 ;* DESCRIPTION: This macro pulls groups of values from the stack.    *
 7 ;*                                                                   *
 8 ;*              In Call Sequence 1,  all the elements of the last    *
 9 ;*              group of values PUSHed on the stack but not yet      *
10 ;*              PULLed off, are now pulled off the stack.  All       *
11 ;*              the values are pulled off in the reverse order       *
12 ;*              of the way they were PUSHed on.  It is illegal       *
13 ;*              to PULL a group of values if you have not prev-      *
14 ;*              iously PUSHed them.                                  *
15 ;*                                                                   *
16 ;*              In Call Sequence 2, the registers, variables or      *
17 ;*              locations are pulled from the stack, left to right.  *
18 ;*              In this mode, you can PULL all you want, it          *
19 ;*              will not check for previous PUSHes.                  *
20 ;*                                                                   *
21 ;* CALL SEQ 1:  PULL                                                 *
22 ;*                                                                   *
23 ;*              PULL the last group ot values PUSHed.   Anything     *
24 ;*              which was placed on the stack using the PUSH         *
25 ;*              macro may be PULLed off using this calling           *
26 ;*              sequence except for immediate addresses.             *
27 ;*                                                                   *
28 ;* INPUT:       None                                                 *
29 ;*                                                                   *
30 ;* OUTPUT:      The last group of values are PULLed from the         *
31 ;*              stack.                                               *
32 ;*                                                                   *
33 ;*              ERRORS  - CAN'T PULL FROM AN EMPTY STACK             *
34 ;*                                                                   *
35 ;* SIDE EFFECTS:The variables PSHFL$  and PSHCT$  are used as        *
36 ;*              well as macro names of the form PSHxx$ where         *
37 ;*              xx is a number between 0 and 77 octal.               *
38 ;*                                                                   *
39 ;* CALL SEQ 2:  PULL STATEMENT  ::=    PULL <group>                  *
40 ;*              group           ::=    arg | arg,group               *
41 ;*                                                                   *
42 ;* INPUT:       arg     - Arguments to be PULLed from the stack,     *
43 ;*              left to right.  They include registers,  variables   *
44 ;*              or locations.   Normally words are PULLed from the   *
45 ;*              stack,  if a byte is to be PULLed,  preceed the      *
46 ;*              argument with an apostrophe (').    As a result,     *
47 ;*              ASCII constants should not be specified since they   *
48 ;*              will be misinterpreted by this macro.  If only one   *
49 ;*              argument is being PULLed, the delimiters < and > are *
50 ;*              not necessary.                                       *
51 ;*                                                                   *
52 ;* OUTPUT:      arg     - Arguments PULLed from stack.               *
53 ;*                                                                   *
54 ;* SIDE EFFECTS:The variable PSHCT$ is used.                         *
55 ;*                                                                   *
56 ;* AUTHOR:      RUDY BAZELMANS                   VERSION: B          *
57 ;*                                                                   *
58 ;*********************************************************************
```

FIGURE 4A. This is the documentation for the PULL macro.

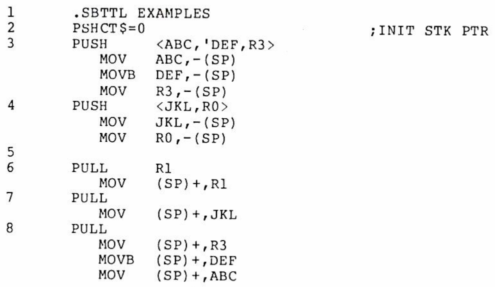


```
.MACRO   PULL   ARGS
   .IF   B   <ARGS>
      .IF   EQ   PSHCT$
         .ERROR   ;CAN'T PULL FROM AN EMPTY STACK
      .IFF
         .IRP   NAME,\PSHCT$
            PSH'NAME'$
         .ENDR
      .ENDC
   .IFF
      .IRP   ARG,<ARGS>
         .IRPC   CHAR,<ARG>
            .IF   IDN   <'>,<CHAR>
               .IRP   VAR,<^'ARG''>
                  MOVB   VAR,-(SP)
               .ENDR
            .IFF
               MOV   (SP)+,ARG
            .ENDC
            .MEXIT
         .ENDR
         .IIF   GT   PSHCT$   PSHCT$=PSHCT$-1
      .ENDR
   .ENDC
.ENDM   PULL
```

FIGURE 4B

This is the definition of the PULL macro.

FIGURE 4B. This is the definition of the PULL macro.



```
.SBTTL EXAMPLES
PSHCT$=0                        ;INIT STK PTR
PUSH     <ABC,'DEF,R3>
   MOV   ABC,-(SP)
   MOVB  DEF,-(SP)
   MOV   R3,-(SP)
PUSH     <JKL,R0>
   MOV   JKL,-(SP)
   MOV   R0,-(SP)

PULL     R1
   MOV   (SP)+,R1
PULL
   MOV   (SP)+,JKL
PULL
   MOV   (SP)+,R3
   MOVB  (SP)+,DEF
   MOV   (SP)+,ABC
```

FIGURE 5. This is a full example of the PUSH and PULL macros in action.

COMMENTS

I have optimized the macros in figures 3 and 4 as much as possible in an effort to decrease memory requirements for frequent users, increase speed and decrease complexity.

There are three important things to be gained from this paper.

1. The constructs used in these macros are complicated, but by understanding each of them, you should be able to design much more powerful macros.
2. The overall concept of using macros and variables to create assembly time stacks is useful in many applications, especially when writing structured macros.
3. You can use the macros in figures 3 and 4 in your own shop. The use of these macros is fully documented and can be used as is. It should save your programmers time and help reduce errors.

REFERENCES

Bazelmans, Rudy. "Are Macros Worth Using?" RSTS Professional, M. Systems Inc., 1981, Vol. 3, No. 3, pp. 20-22.

PDP-11 Macro-11 Language Reference Manual (AA-5075A-TC). Maynard, MA., Digital Equipment Corp.

A RSTS INTERNALS MANUAL by Mike Mayfield
More than 200 pages in a binder.
— PUBLISHED BY THE RSTS PROFESSIONAL —
Price - $95.00
SEND ORDERS TO:
M SYSTEMS, INC., BOX 361, FORT WASHINGTON, PA 19034-0361

WHY TAKE RISKS ?

NPI proves purchasing Disk Sub-systems does not have to be

HIGH RISK
VS
BIG SAVINGS

- Disk drives manufactured by the same vendor DEC buys them from
- No operating systems software changes
- One day installation
- 100% software transparency
- DEC's RM05-RM03-RM02 drivers are used
- 50% cost savings
- Single vendor maintenance in major cities
- Uses DEC's diagnostics
- Timely delivery
- DEC RP06-RP04-RM03 RM02 accepted on trade

NPI National Peripherals, Inc.

41 Chestnut Lane
Westmont, IL 60559
(312) 325-9700

CIRCLE 62 ON READER CARD

POLYSOFT APPLICATION LANGUAGE

By S. Zuk (Non-DECUS Member), Polyfibron Division, W. R. Grace and Company, Lexington, Massachusetts

Presented to DECUS Fall 1981 Conference — Los Angeles

1.0 INTRODUCTION

The Polysoft Application Language (PAL) was designed as an interface language between the polysoft data bases [1] and various business applications. PAL was developed as a user oriented language which allows development applications software without having an in-depth understanding of hardware and system software. A comprehensive report program [2] (Report Manager) was created to complement PAL.

2.0 LANGUAGE STRUCTURE

Each line of PAL source code consists of a label or line number, a function command and a series of parameters (variable slots or data base element numbers) required to execute a line of code.

PAL source code must be run through a compiler like Basic + 2 program before the application program can access data base. The PAL compiler will examine the source code for such items as correct function commands, data base element verification, missing loop logic and missing or incorrect parameter declaration. The compiled object code is stored in a virtual array format and is executed through the polysoft data base management system. Upon completion of the compiler an error listing is generated. Correction of errors and resubmission to the compiler must be done before the program is executed.

The PAL application programs functions are designed to handle record I/O with all the features and techniques of BASIC + 2 without the need for dealing with files at the bit and byte level. The end product is an executable program allowing for on-line interactive data base manipulation or batch processing.

3.0 LANGUAGE COMMANDS

The PAL commands fall into three main categories:

3.0.1 Commands designed to handle record I/O.

3.0.2 Commands designed to handle applications required logic to manipulate the records and elements within the data base.

3.0.3 Special feature commands

3.1 Record I/O Functions

The I/O function commands allow the user to add, change, delete and inquire on data base files, records or elements.

File Close [CLEAR]

Allows for selective closing of files over and above the data base managers dynamic housekeeping routines.

The code would appear as follows:

```
A. LABELXX,CLEAR  !R750      !    !
                        OR
B. LABELYY,CLEAR  !          !    !
```

A. Will close a specific file [R750]

B. Will close all files open at that time

Find Record [FR]

This command will allow the user to select a record from a specific file using the keys specifications for the file. It will request from the system a screen containing prompptable keys for searching. Once the command is executed it will save the record number of the record requested to be used by other functions.

The code would appear as follows:

```
LABEL11,FR  !R750,SLOT1  !SCR:SCREEN.TXT,LABEL99  !FIND HEADER
```

LABEL11 — Line Number

FR — Command

R750 — File to Search

SLOT1 — Storage Area for Record Number Found

SCR:SCREEN.TXT — User Defined Interface Screen-See Below

LABEL99 — Step to Go to Incase of Abort

FIND HEADER — Comment

```
--------------------------------------------------------------------------
!                                                          [SCREEN]     !
!                                                                       !
!     M A T E R I A L   R E Q U I S I T I O N   S E L E C T             !
!                                                                       !
!                                                                       !
!                                                                       !
!                                                                       !
!    TYPE CODE                   : ------                               !
!                                                                       !
!    PRODUCT DESCRIPTION         : --------------------                 !
!                                                                       !
!    ABBREVIATED NAME            :  ----------                          !
!                                                                       !
!                                                                       !
!                                                                       !
--------------------------------------------------------------------------
```

Highest Record [HR]

This allows the user to find the last and highest record number issued by the system software. This can be used for controlled record access.

The code would appear as follows:

```
LABELXX,HR  !R750,SLOT2    !    !GET REC NUM
```

HR — Command

R750 — File to Access

SLOT2 — Storage Area for Highest Record Number

Input File [IF]

The [IF] command will retrieve from the data base a logical group of elements from a record. Retrieval is based on the record number within a specific file. This record number is totally transparent to the user.

The code would appear as follows:

```
LABELCC,IF  !R750,SLOT3,G700    !ABORTEE    !
```

IF — Command

R750 — File to Access

SLOT3 — Holding Area Containing Record Number to Access

G700 — Group within the Record

ABORTEE — Line to Go To in Case of Incorrect Group Selection

NOTE: Associated with this file/group is an overlay screen that is used to display to the user record data.

Move Data [MOVE]

This instruction is like a data transfer statement in any other language. It allows the user to move elements from one file to another or move specific values to files or to other parts of the program without altering data.

The code would appear as follows:

```
A. LABEL11,MOVE   !F5002.1.R750,SLOT10,,F5002.1.R850,SLOT20 ! !
B. LABEL22,MOVE   !,,SLOT5,,,SLOT6   !   !
C. LABEL33,MOVE   !F5002.1.R750,SLOT10,,,,SLOT6   !   !
D. LABEL44,MOVE   !,,SLOT5,F5002.1.R850,SLOT20   !   !
```

A. This type of code will transfer the information contained in element [F5002.1.R750] with the record number [SLOT10] and place it in element [F5002.1.R850] with a record number index [SLOT20].

B. The slot to slot movement is passing data from [SLOT5] into [SLOT6].

C. Moves a value from data base field to a storage area [SLOT6].

D. Moves a value from a storage area [SLOT5] to a data base field.

Operation Mode [OM]

The function [OM] will allow the user program to handle data maintenance requirements as needed. It allows the user to add, delete, reinstate and inquire on records or groups within the record. Changes can be made at the record, group and field levels.

The code would appear as follows:

```
LABEL 11,OM   !              !ADD              !
LABEL 12,OM   !              DELETE            !
LABEL 13,OM   !              !REINSTATE        !
LABEL 14,OM   !              !INQUIRE          !
LABEL 15,OM   !              !CHANGE RECORD    !
LABEL 16,OM   !              !CHANGE GROUP     !
LABEL 17,OM   !F5002.1.R750  !CHANGE FIELD     !
```

The above statements indicate the function called [OM] and the operation to be performed i.e.,ADD.

Record Issue [RA]

The record issue function allows the user to get the next available record number for the file being operated upon.

The code would appear as follows:

```
LABEL22,RA     !SLOT25,R750   !     !
```

RA — Command
SLOT25 — Storage Area of Record Number for Later Referal
R750 — The File to Get the Next Record Number

Record Search [SL]

This command gives the application user the ability to search other data base files within the system using information that is supplied by either data from other files or from user response.

The code would appear as follows:

```
LABELVV,SL   !SLOT2,F5002.1.R750,SLOT20,,FR750.S      !,T     !
```

SL — Command
F5002.1.R750 — Key Field to Search
SLOT20 — Value User to Search F5002.1.R750
FR750.S — Second Key to Search
T — Constant Value used to Search Second Key

Save Record Number [SR]

Gives the user the ability to save record numbers for future reference within the application program. This instruction usually follows the [FR] command.

The code would appear as follows:

```
LABELQQ,SR   !SLOT200     !    !
```

SR — Command
SLOT200 — Storage Area for a Record Number

BRAND NEW • FULL 90 DAY WARRANTY

DEC

DEC 1200 BAUD PRINTERS

LA120-AA. EIA, KSR, Keyboard Only $1,995
LA120-BA. EIA, Keyboard & Keypad, KSR 2,075
LA120-RA. Receive Only 1,795

CASH PRICES ● IN STOCK
IMMEDIATE DELIVERY

CALL SONJA OR LAURIE AT:
(614) 889-0810

SCHERER'S
MINI COMPUTER MART
6145 Dolan Place Dublin, Ohio 43017

BRAND NEW*WARRANTY*ATD

CIRCLE 104 ON READER CARD

3.2 Logic Functions

Arithmetic Operations [A]

Allows a user to perform arithmetic operations on paired storage areas, fields and/or constant values. It allows for the four basic math functions plus the concatenation of string data.

The code would appear as follows:

```
LABELTT,A   !+,SLOT2,,SLOT3,,SLOT4,2,5      !     !
```

A — Command
+ — Type of Operation to be Performed [also -,*,/,~]
SLOT2 — Value 1
SLOT3 — Value 2
SLOT 4 — Sum of Value 1 and Value 2
2 — Number of Decimals to be Retained in the Result
5 — Number of Leading Zeroes to be Inserted in the Result

IF/THEN Type Compares [C]

This command is like the IF/THEN conditional of statement used in Basic. It allows for the testing of conditions within the application to determine alternate courses of action based on parameters of the test.

The code would appear as follows:

```
LABELPP,C       !=,SLOT1      !NO,LABELXX,$   !
```

C — Command
= — Type of Comparison to be Done
SLOT1 — Variable to Compare
NO — Constant Value
LABELXX — Branch Destination if Compare is True
$ — Indicates Comparison is to be Done on a String of Data

Extraction Functions [EXTRACT]

Extract function performs the same type of operations as the instruct, left and right functions in Basic. See attached documentation for more detail.

GOTO [GO]

This command allows the user to branch to specific parts of the application program unconditionally as in Basic.

The code would appear as follows:

```
LABEL23,GO  !    !LABEL99     !
```

GO — Command

LABEL99 — Branch to Destination

Input Prompts [IS]

This function will suspend the application and require the user to input a response that may be used to redirect the program flow or supply a variable to be used as a search value within the current application. This prompt will appear at the bottom of the screen and is independent of any overlay screens.

The code would appear as follows:

```
LABEL77,IS   !SLOT30,10,1   !ENTER Y or N,LABEL77   !
```

IS — Command

SLOT30 — Storage Area for User Input

10 — Ten Second Response Timer

1 — Maximum Number Characters to be Input

ENTER Y or N — Prompt Message to Appear on Screen Before Entry

LABEL77 — Return Step

Format Justification [JUST]

This command is similar to a picture statement, it allows for left and right justification and the padding of data with other characters. See attached documentation for detail of governing parameters.

3.3 Special Functions

Polysoft Application Language, like other languages contains special features which allows the user to generate internal system sequential control numbers, display messages, use special date and time features. Also provided is the ability allowing the application user to use external text files in conjunction with data base files. The attached documentation lists these functions and parameters governing the proper usage within the polysoft system.

Partial list of special functions:

Control Numbers	[CN]
Message Display	[DM]
Date/Time	[DT]
Special Screens	[SSP]
Text File Open	[TO]
Text File Close	[TC]

4.0 TESTING AND DEBUGGING

The testing and debugging of a PAL generated program uses the same techniques that one would use in debugging any other language. For example, a debugging display feature is an integral part of the language.

5.0 SUMMARY

Users who are familiar with basic programming conventions, and who have a sound knowledge of the particular data base file structure can adapt to the language quickly and easily. As in the design and programming of business type applications a good understanding of the business environment is also essential to the systems analyst. This form of language can translate design into functional applications in a short period of time.

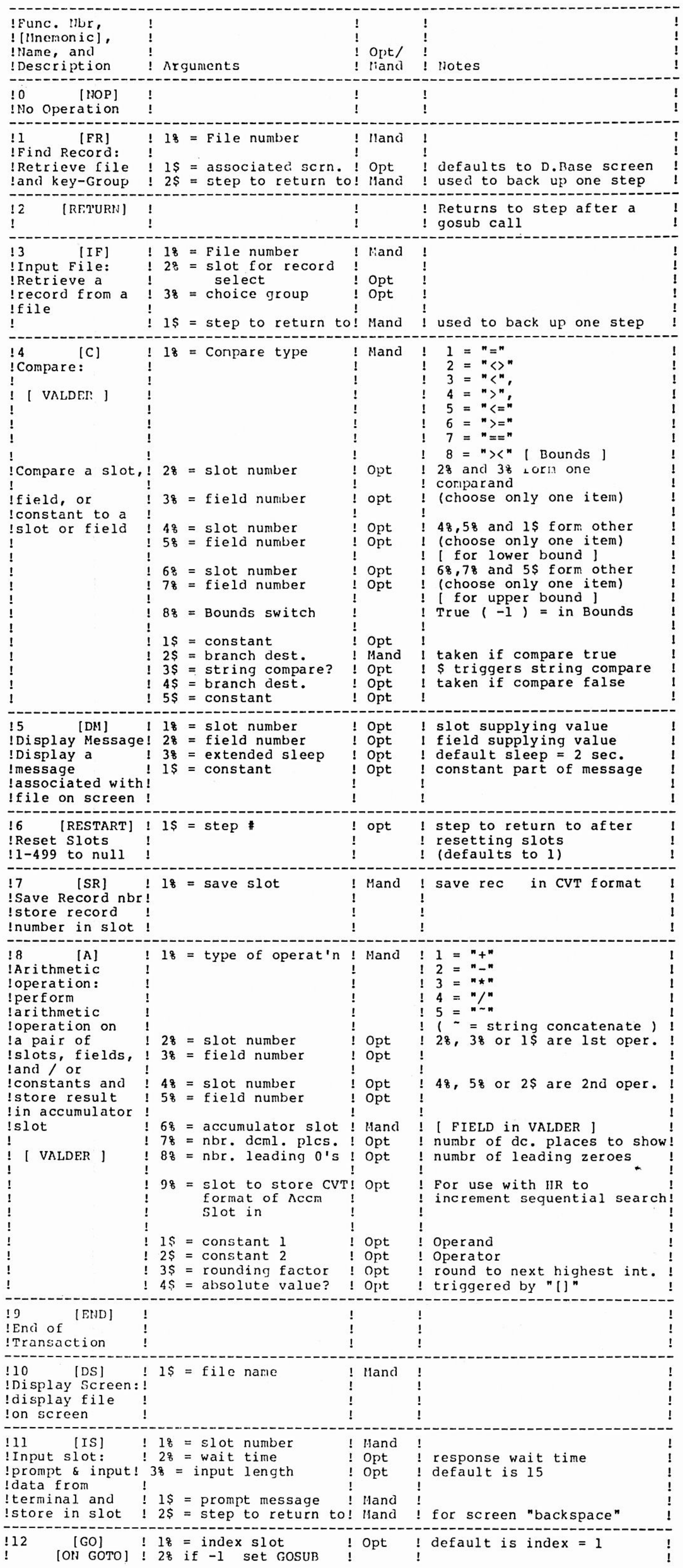
Attached are PAL functions that make up the language, along with a copy of a designed application interfacing with the polysoft data base using several of the language features.

6.0 BIBLIOGRAPHY

[1] R.R. Jaques, A. Eloy, "Design of a Data Base Management System for W.R. Grace and Company" December 1981, Fall DECUS US Symposium

[2] J.M. Prigot, "Implementation of an Application Under the W.R. Grace Data Base System" December 1981, Fall DECUS US Symposium

POLYSOFT APPLICATION LANGUAGE - PAL



Func. Nbr, [Mnemonic], Name, and Description	Arguments	Opt/ Mand	Notes
0 [NOP]			
No Operation			
1 [FR]	1% = File number	Mand	
Find Record:			
Retrieve file	1$ = associated scrn.	Opt	defaults to D.Base screen
and key-Group	2$ = step to return to	Mand	used to back up one step
2 [RETURN]			Returns to step after a
			gosub call
3 [IF]	1% = File number	Mand	
Input File:	2% = slot for record		
Retrieve a	select	Opt	
record from a	3% = choice group	Opt	
file			
	1$ = step to return to	Mand	used to back up one step
4 [C]	1% = Compare type	Mand	1 = "="
Compare:			2 = "<>"
			3 = "<",
[VALDER]			4 = ">",
			5 = "<="
			6 = ">="
			7 = "=="
			8 = "><" [Bounds]
Compare a slot,	2% = slot number	Opt	2% and 3% form one
			comparand
field, or	3% = field number	opt	(choose only one item)
constant to a			
slot or field	4% = slot number	Opt	4%,5% and 1$ form other
	5% = field number	Opt	(choose only one item)
			[for lower bound]
	6% = slot number	Opt	6%,7% and 5$ form other
	7% = field number	Opt	(choose only one item)
			[for upper bound]
	8% = Bounds switch		True (-1) = in Bounds
	1$ = constant	Opt	
	2$ = branch dest.	Mand	taken if compare true
	3$ = string compare?	Opt	$ triggers string compare
	4$ = branch dest.	Opt	taken if compare false
	5$ = constant	Opt	
5 [DM]	1% = slot number	Opt	slot supplying value
Display Message	2% = field number	Opt	field supplying value
Display a	3% = extended sleep	Opt	default sleep = 2 sec.
message	1$ = constant	Opt	constant part of message
associated with			
file on screen			
6 [RESTART]	1$ = step #	opt	step to return to after
Reset Slots			resetting slots
1-499 to null			(defaults to 1)
7 [SR]	1% = save slot	Mand	save rec in CVT format
Save Record nbr			
store record			
number in slot			
8 [A]	1% = type of operat'n	Mand	1 = "+"
Arithmetic			2 = "-"
operation:			3 = "*"
perform			4 = "/"
arithmetic			5 = "~"
operation on			(~ = string concatenate)
a pair of	2% = slot number	Opt	2%, 3% or 1$ are 1st oper.
slots, fields,	3% = field number	Opt	
and / or			
constants and	4% = slot number	Opt	4%, 5% or 2$ are 2nd oper.
store result	5% = field number	Opt	
in accumulator			
slot	6% = accumulator slot	Mand	[FIELD in VALDER]
	7% = nbr. dcml. plcs.	Opt	numbr of dc. places to show
[VALDER]	8% = nbr. leading 0's	Opt	numbr of leading zeroes
	9% = slot to store CVT	Opt	For use with HR to
	format of Accm		increment sequential search
	Slot in		
	1$ = constant 1	Opt	Operand
	2$ = constant 2	Opt	Operator
	3$ = rounding factor	Opt	round to next highest int.
	4$ = absolute value?	Opt	triggered by "[]"
9 [END]			
End of			
Transaction			
10 [DS]	1$ = file name	Mand	
Display Screen:			
display file			
on screen			
11 [IS]	1% = slot number	Mand	
Input slot:	2% = wait time	Opt	response wait time
prompt & input	3% = input length	Opt	default is 15
data from			
terminal and	1$ = prompt message	Mand	
store in slot	2$ = step to return to	Mand	for screen "backspace"
12 [GO]	1% = index slot	Opt	default is index = 1
[ON GOTO]	2% if -1 set GOSUB		

```
!     [GOSUB]    !                        !       !                            !
!Computed GOTO:  !                        !       !                            !
!using index     ! 1$ = step number       ! Mand  ! index = 1                  !
!number in slot, ! 2$ =  "      "         ! Opt   ! index = 2                  !
!GOTO step       ! 3$ =  "      "         ! Opt   ! index = 3                  !
!number in       ! 4$ =  "      "         ! Opt   ! index = 4                  !
!Job-Stream      ! 5$ =  "      "         ! Opt   ! index = 5                  !
--------------------------------------------------------------------------------
!13      [STOP]  !                        !       !                            !
!Stops execution !                        !       !                            !
!of Job-Stream.  !                        !       !                            !
!"cont" reopens  !                        !       !                            !
!KB block mode   !                        !       !                            !
!and continues   !                        !       !                            !
!from next step. !                        !       !                            !
--------------------------------------------------------------------------------
!14      [CLEAR] ! 1% = file number       ! Opt   !                            !
!Clears a        !                        !       !                            !
!specific file   !                        !       !                            !
!or all open     !                        !       !                            !
!from DCS        !                        !       !                            !
!programs        !                        !       !                            !
--------------------------------------------------------------------------------
!15      [SL]    ! 1% = dest. slot        ! Mand  !                            !
!Save from       !                        !       !                            !
!lookup:         ! 2% = keyed field     #1! Mand  !                            !
!store record    ! 3% = slot            #1! Opt   !slot containing key val     !
!number pointed  ! 4% = field           #1! Opt   !field containing key val    !
!to by key(s)    !                        !       !                            !
!in dest. slot   ! 5% =(same as ARG 2%)#2! Opt    !                            !
!                ! 6% =(same as ARG 3%)#2! Opt    !                            !
!                ! 7% =(same as ARG 4%)#2! Opt    !                            !
!                !                        !       !                            !
!                ! 8% =(same as ARG 2%)#3! Opt    !                            !
!                ! 9% =(same as ARG 3%)#3! Opt    !                            !
!                ! 10%=(same as ARG 4%)#3! Opt    !                            !
!                !                        !       !                            !
!                ! 11%=(same as ARG 2%)#4! Opt    !                            !
!                ! 12%=(same as ARG 3%)#4! Opt    !                            !
!                ! 13%=(same as ARG 4%)#4! Opt    !                            !
!                !                        !       !                            !
!                ! 15%=stop after select?! Opt    ! -1% = stop                 !
!                !                        !       !                            !
!                ! 1$ = key constant    #1! Opt   !const containing key val    !
!                ! 2$ = (same as 1$)    #2! Opt   !                            !
!                ! 3$ = (same as 1$)    #3! Opt   !                            !
!                ! 4$ = (same as 1$)    #4! Opt   !                            !
--------------------------------------------------------------------------------
!16      [COPY]  ! 1% = operation mode    ! Mand  ! 0% = copy what's on screen  !
!Copy to printer !                        !       ! 1% = copy test file silent !
!port            !                        !       ! 2% = copy form feed        !
! (VT100 &       !                        !       !                            !
! DT80/1 only )  !                        !       !                            !
!                !                        !       !                            !
!                ! 1$ = file to copy      !       ! if option 1% set in 1%     !
!                !                        !       !                            !
--------------------------------------------------------------------------------
!17      [OM]    ! 1$ = operation mode    ! Mand  !                            !
!Set Opmode      !                        !       !                            !
!(ADD,           !                        !       !                            !
!DELETE,         !                        !       !                            !
!REINSTATE,      !                        !       !                            !
!INQUIRE,        !                        !       !                            !
!CHANGE RECORD,  !                        !       !                            !
!CHANGE FIELD,   !                        !       !                            !
!CHANGE GROUP)   !                        !       !                            !
--------------------------------------------------------------------------------
!18      [RA]    ! 1% = slot for rec num  ! Mand  ! remember to set status code!
!Record          ! 2% = file number       ! Mand  ! to an a it is set to Q by  !
!advance         !                        !       ! this Job Stream function   !
!issues a rec.   !                        !       !                            !
--------------------------------------------------------------------------------
!19      [MOVE]  ! 1% = source field      ! Opt   !                            !
!Move data       ! 2% = slot containing   ! Opt   ! (if blank, retrieve from   !
!Move data from  !      source record     !       !  work file)                !
!a slot,field,   ! 3% = source slot       ! Opt   !                            !
!or constant to  !                        !       !                            !
!a slot or field ! 4% = receiving field   ! Opt   ! [ one source argument and  !
!                ! 5% = slot containing   ! Opt   !   one receiving argument   !
! [ VALDER ]     !      receiving record  !       !   must be chosen. ]        !
!                !      number            !       !                            !
!                ! 6% = receiving slot    ! Opt   !                            !
!                ! 7% = -1% text file rd  ! Opt   ! read from open text file   !
!                !                        !       !                            !
!                ! 8% = -1% text file wr  ! Opt   ! write to open text file    !
!                !                        !       !                            !
!                ! 9% = -1% hold line     !       ! for writes : holds line and!
!                !                        !       ! inserts commas until 9% is !
!                !                        !       ! zero                       !
!                ! 11% = Fraction convert!        ! -1 = convert on move       !
!                !                        !       !                            !
!                !                        !       ! for reads : reads from line!
!                !                        !       ! held in memory instead of  !
!                !                        !       ! fetching new line ( NOTE   !
!                !                        !       ! reading past end of line   !
!                !                        !       ! does not error but will    !
!                !                        !       ! return a blank item        !
!                !                        !       !                            !
!                ! 10% = slot             !       ! slot containing index      !
!                !                        !       ! passed in text open        !
!                !                        !       !                            !
!                ! 1$ = source constant   ! Opt   !                            !
--------------------------------------------------------------------------------
!20      [WAIT]  ! 1% =   seconds to wait! Mand   ! must be reset to 0 before  !
!controlled      !                        !       ! exiting routine            !
!time - out      !                        !       !                            !
--------------------------------------------------------------------------------
!21      [CN]    ! 1% = field for Control! Mand   ! VALDER used active fld if  !
!                !                        !       ! field is blank             !
!Generate        ! 2% = Slot to store CN  ! Mand  ! FIELD for VALDER           !
!Control number  !                        !       !                            !
! [ VALDER ]     !                        !       !                            !
--------------------------------------------------------------------------------
!22      [DT]    ! 1% = Time/date format  ! Mand  ! 1 = 6 char date   (lngth=6) !
!Store Time or   !      desired           !       ! 2 = alpha date    (lngth=9) !
!Date in a slot  !                        !       ! 3 = ISO date      (lngth=8) !
!                !                        !       ! 4 = AM/PM time    (lngth=8) !
! [ VALDER ]     !                        !       ! 5 = 24-hour time  (lngth=5) !
!                !                        !       !                            !
!                ! 2% = if date forward   ! Opt   !-1 = date back function     !
!                !       back or day      !       !+1 = date forward function  !
!                !                        !       ! 2 = day of week, specify   !
!                !                        !       !     3 in 1%                !
!                !                        !       !                            !
!                ! 3% = destination slot  ! Mand  ! FIELD for VALDER           !
!                !                        !       !                            !
!                ! 4% = increment value   ! Opt   ! used with date back and for!
!                !                        !       !                            !
```

DEC

DEC CRT'S

VT101-AA.	EIA, Non Upgradeable	$ 995
VT131-AA.	w/AVO, PCO, & Screen Editing	1,575
VT100-AA.	EIA	1,375
VT132-AA.	AVO, Screen Editing	1,450

CASH PRICES ● IN STOCK
IMMEDIATE DELIVERY

CALL SONJA OR LAURIE AT:
(614) 889-0810

SCHERER'S
MINI COMPUTER MART

6145 Dolan Place Dublin, Ohio 43017

BRAND NEW * FULL 90 DAY WARRANTY

BRAND NEW*WARRANTY*ATD

CIRCLE 105 ON READER CARD

```
--------------------------------------------------------------------------------
!23   [EXTRACT] ! 1% = subfunction       ! Mand  ! INSTR = 1                   !
!Performs ext-  !                        !       ! LEFT  = 2                   !
!traction on    !                        !       ! RIGHT = 3                   !
!slots .        !                        !       !                             !
!(1) search for ! 2% = source slot       ! Mand  !                             !
!   given value ! 3% = slot with RIGHT   !       !                             !
!               !      val               !       ! use with RIGHT              !
!   in a slot   ! 4% = slot with LEFT    !       !                             !
!               !      val               !       ! use with LEFT               !
!   return beg  ! 5% = RIGHT ADD val     !       ! to set start ADD with 3%    !
!   val to slot ! 6% = LEFT  ADD val     !       ! to set end ADD with 4%      !
!(2) get from   ! 7% = receiving slot    ! Mand  !                             !
!   LEFT to sum ! 8% = INSTR start pos   !       !                             !
!   of 4% + 6%  ! 9% = ASCII search val  !       ! or string 1                 !
!   of 4% + 6%  !                        !       !                             !
!(3) get from   ! 1$ = search value      !       ! or 8%                       !
!   3% + 5% to  !                        !       !                             !
!   RIGHT       !                        !       !                             !
--------------------------------------------------------------------------------
!24      [TO]   ! 1% = Mode for open     ! Opt   ! default is zero             !
!Text file open !                        !       ! valid options are           !
!               !                        !       ! 0 = open for output         !
!               !                        !       ! 8192 = read only            !
!               !                        !       ! 2 = append mode             !
!               !                        !       !                             !
!               ! 2% = slot for message  ! Mand  !                             !
!               !       or index         !       !                             !
!               !                        !       !                             !
!               ! 3% - slot containing   ! Mand  !                             !
!               !      file name         !       !                             !
!               !                        !       !                             !
--------------------------------------------------------------------------------
!25      [TC]   !                        !       !                             !
!Text file close! 1% = slot   from index! Mand   !                             !
!               !      slot in open      !       !                             !
!               !                        !       !                             !
--------------------------------------------------------------------------------
!26      [MR]   !                        !       !                             !
!Move Record    ! 1% = record number     ! Mand  !                             !
! moves a rec.  !                        !       !                             !
! and returns   ! 2% = slot number of    ! Mand  !                             !
! slot with new !      old record        !       !                             !
! rec number    !                        !       !                             !
!               ! 3% = slot for new      ! Mand  !                             !
!               !      record number     !       !                             !
--------------------------------------------------------------------------------
!25      [JUST] !                        !       !                             !
! Justify       ! 1% = source slot       ! Mand  !                             !
!               !                        !       !                             !
!               ! 2% = receiving slot    ! Mand  !                             !
!               !                        !       !                             !
!               ! 3% = Length of field   ! Mand  ! insure that length is not   !
!               !                        !       ! greater than Data Base      !
!               !                        !       ! length or item may truncate!
!               !                        !       !                             !
!               ! 4% = Left/Right Just   ! Mand  !  0 = LEFT Justify           !
!               !                        !       ! -1 = RIGHT Justify          !
!               !                        !       !                             !
!               ! 5% = ASCII fill val    ! Mand  ! 0 or null will not work     !
!               !                        !       ! correctly on justifies      !
--------------------------------------------------------------------------------
!28      [DEBUG]!                        !       !                             !
!Debug module   !                        !       !                             !
! call          !                        !       !                             !
--------------------------------------------------------------------------------
!29      [SSP]  ! 1% = from slot         ! Mand  ! Must be sequential          !
!Slot Screen    ! 2% = to slot           ! Mand  ! Error only if too many      !
!Printer        !                        !       ! slots for screen            !
!               ! 3% = skip confirm      !       ! for use with copy features !
!               ! 4% = Interactive mode  !       ! True = interactive slot scr!
!               ! 1$ = screen file name  !       !                             !
--------------------------------------------------------------------------------
```

DEC® PDP-11® & LSI-11® Users

Until now, adding a magnetic tape subsystem to Digital Equipment's PDP-11 and LSI-11 series computers meant costly or custom-built interfaces and integrating components from a multiplicity of vendors. It didn't always work.

Now, with IDT's series 1050 magnetic tape and series 3000 cartridge tape subsystems, adding a cost-effective, high-performance data storage and retrieval system to PDP-11s and LSI-11s is a simple, uncompromising procedure.

Innovative Data Technology's tape subsystems are designed with compatibility in mind. Our ½" tape drives use industry-standard interfaces and require only one CPU slot for the controller card; or, if your card cage is full, IDT will put your terminator on the drive, avoiding the cost of an expansion chassis. In short, IDT offers complete turnkey systems, including tape coupler, formatter and drive.

DEC TM-11 COMPATIBLE TAPE SUBSYSTEM

$6550.00

IN 100 LOT
OEM QUANTITIES

If you have a PDP-11 or LSI-11, you'll want these features from Innovative Data Technology's tape subsystems:

- Single supplier support
- DEC standard TM-11 software compatibility
- Single slot connection or controller card
- Dual mode 800 cpi (NRZI) and 1600 cpi (PE), at 45 ips; 9-track Read-After-Write for mag tape.
- 1600 cpi ANSI/EMCA standard for cartridge tape
- Six-month warranty on complete system
- Factory installation and training available
- Attractive pricing

For complete details about Innovative Data Technology's tape subsystems for DEC computers, call or write for a complementary descriptive brochure.

IDT: where innovation puts you ahead

INNOVATIVE
DATA
TECHNOLOGY
4060 MORENA BLVD. • SAN DIEGO, CA 92117
(714) 270-3990 • TWX: 910-335-1610
IDT EAST (703) 759-3003

DEC, PDP-11 and LSI-11 are registered trademarks of Digital Equipment Corporation

CIRCLE 96 ON READER CARD

```
!30      [HR]     ! 1% = file number       ! Mand  ! retrieves highest rec num  !
!Highest Record ! 2% = storage slot      ! Mand  ! for sequential searches    !
! number        !                        !       !                            !
-------------------------------------------------------------------------------
!31      [WIDTH] ! 1% = width size        !       ! width limited to 128 on    !
! DT80/I Width  !                        !       ! input screens              !
! command       !                        !       !                            !
-------------------------------------------------------------------------------
!32     [GRAPHIC]!                        !       ! Set terminal in graphics   !
! Graphics mode !                        !       ! mode. Auto shut of after   !
!               !                        !       ! printing screen            !
-------------------------------------------------------------------------------
!33     [MRT]    ! 1% = Switch <> 0 on    !       ! True = get highest rec num !
! Use silent MRT!                        !       !        and store in slot   !
! If arg 15% set!                        !       !        defined in 2%       !
! in SL         ! 2% = Storage Slot      !       !                            !
-------------------------------------------------------------------------------
!34     [TS]     ! 1% to 15%              !       ! Slot number to save        !
! Tag Slot      !                        !       !                            !
! Saves slots in!                        !       !                            !
! Restarts or   !                        !       !                            !
! chains        !                        !       !                            !
-------------------------------------------------------------------------------
!35     [CHAIN]  ! 1% = Jobstream to      !       !                            !
! CHAIN         !       chain to         !       !                            !
!               ! 1$ = step to start at  !       ! printing screen            !
-------------------------------------------------------------------------------
!36     [TEN]    ! 1% = existence = 0%   !       !                            !
! VALDER ONLY   !      nonexistence = -1%!       !                             !
! Test for      !                        !       !                            !
! Existence     !                        !       !                            !
-------------------------------------------------------------------------------
!37     [TYPE]   ! 1% = type test         !       ! refer to RDF for type      !
! Test type on  ! 2% = Slot with value   !       !                            !
! slot          ! 2$ = branch dest.      ! Mand  ! taken if type true         !
!               ! 4$ = branch dest.      ! Opt   ! taken if type false        !
-------------------------------------------------------------------------------


!----------------------------------------------------------------------!
!          JOBSTREAM          IDENTIFICATION        HEADING            !
!                                                                      !
!                        PURPOSE :  SALES                              !
!                         STATUS :  ACTIVE LEXINGTON                   !
!                         STATUS :  INACTIVE ADAMS                     !
!                           DATE:   MAR/81                             !
!                                                                      !
!                         AUTHOR :  S R ZUK                            !
!                           NAME :  P44STR.STR                         !
!                           DATE:   MAR/81                             !
!                                                                      !
!  SUMMARY DESCRIPTION:                                                !
!  ======= ===========                                                 !
!     COMPLAINT REPORT CREATION AND UPDATE                             !
!                                                                      !
!                                                                      !
!----------------------------------------------------------------------!
[0001] 00000000,NOP       !        !        !
[0002] 00000400,DS        !                          !SCR:COMPLT.TXT !
[0003] 00000500,IS        !SLOT10,,1        !CHOICE,00000500         !
[0004] -         ,C       !=,SLOT10         !1,00001000,$    !
[0005] -         ,C       !=,SLOT10         !2,10000000,$    !
[0006] -         ,C       !=,SLOT10         !3,20001000,$    !
[0007] -         ,C       !=,SLOT10         !4,30001000,$    !
[0008] -         ,C       !=,SLOT10         !E,99999999,$    !
[0009] -         ,GO      !                 !00000500        !
!
!   INQUIRE
!
[0010] 00001000,OM        !                 !INQUIRE         !
[0011] 00001010,IS        !SLOT11,,6        !COMPLAINT NO or <CR>,00001010  !
[0012] -         ,C       !=,SLOT11         !,00000400,$     !
[0013] -         ,SL      !SLOT100,F9050.1.R944,SLOT11       !          !
[0014] -         ,C       !<>,SLOT100       !,00000600,$     !
[0015] -         ,DM      !SLOT11           !DOES NOT EXIST  !
[0016] -         ,GO      !                 !00001010        !
[0017] 00000600,IF        !R944,SLOT100,G944        !00000600           !
[0018] -         ,OM      !                 !ADD     !
[0019] -         ,CLEAR   !                 !                !
[0020] 00000700,IS        !SLOT2,,1         !AGAIN (Y or N),00000700        !
[0021] -         ,C       !=,SLOT2          !Y,00001000,$    !
[0022] -         ,GO      !                 !00000400        !
!
!   ADD
!
[0023] 10000000,RA        !SLOT1,R944       !
[0024] 10000002,IS        !SLOT13,,6        !COMPLAINT NUMBER or <CR>,10000002!
[0025] -         ,C       !=,SLOT13                  !,10005500,$     !
[0026] 10000100,SL        !SLOT14,F9050.1.R944,SLOT13,,FR944.S    !,T     !
[0027] -         ,C       !=,SLOT14         !,10001000,$     !
[0028] -         ,DM      !SLOT13 !COMPLAINT NO. EXISTS      !          !
[0029] 10000102,IS        !SLOT13,,6        !COMPLAINT NUMBER or <CR>,10000102!
[0030] -         ,C       !=,SLOT13         !,10005500,$     !
[0031] -         ,GO      !                 !10000100        !
--------------------------------------------------------------------------------

[0032] 10001000,IF        !R944,SLOT1,G944          !10005500           !
[0033] 10001009,MOVE      !F9062.1.R944,SLOT1,,,,SLOT12    !          !
[0034] 10001010,C         !=,SLOT12         !1,10005000,$    !
[0035] -         ,C       !=,SLOT12         !2,10005000,$    !
[0036] -         ,C       !=,SLOT12         !3,10005000,$    !
[0037] -         ,DM      !                 !ACTION FLAG MUST BE 1/2 or 3    !
[0038] 10001020,IS        !SLOT12,,1        !ACTION REQUESTED -,10001020     !
[0039] -         ,MOVE    !,,SLOT12,F9062.1.R944,SLOT1       !          !
[0040] -         ,GO      !                 !10001010        !
[0041] 10005000,CLEAR     !        !        !
```

SPECIAL OFFER

MACHINE READABLE 800 BPI DOS LABEL OF MAJOR PROGRAMS IN THIS ISSUE

Part of the proceeds will be going to the authors.

Send $50 to:

M SYSTEMS, INC.

Box 361, Fort Washington, PA 19034-0361

```
[0042] -           ,GO      !                   !10006000         !
[0043] 10005500,MOVE        !,,,FR944.S,SLOT1         !D          !
[0044] 10006000,IS          !SLOT2,,1           !AGAIN ( Y or N),10006000         !
[0045] -           ,C       !=,SLOT2            !Y,10000000,$     !
[0046] -           ,GO      !                   !00000400         !
!
!    CHANGE ROUTINE
!
[0047] 20001000,OM          !                   !CHANGE RECORD    !
[0048] 20001010,IS          !SLOT11,,6          !COMPLAINT NO or <CR>,20001010    !
[0049] -           ,C       !=,SLOT11           !,00000400,$      !
[0050] -           ,SL      !SLOT100,F9050.1.R944,SLOT11,,FR944.S     !,T         !
[0051] -           ,C       !<>,SLOT100         !,20000600,$      !
[0052] -           ,DM      !SLOT11             !DOES NOT EXIST   !
[0053] -           ,GO      !                   !20001010         !
[0054] 20000600,IF          !R944,SLOT100,G944        !20001010          !
[0055] -           ,OM      !                   !ADD    !
[0056] -           ,MOVE    !F9065.1.R944,SLOT100,,,,SLOT32 !        !
[0057] -           ,C       !=,SLOT32           !,20000650,$      !
[0058] -           ,MOVE    !,,,FR944.S,SLOT100       !D          !
[0059] 20000650,CLEAR       !                   !                 !
[0060] 20000700,IS          !SLOT2,,1           !AGAIN (Y or N),20000700          !
[0061] -           ,C       !=,SLOT2            !Y,20001000,$     !
[0062] -           ,GO      !                   !00000400 `       !
!
!   ROUTING CHANGE
!
[0063] 30001000,OM          !                   !INQUIRE          !
[0064] 30001010,IS          !SLOT11,,6          !COMPLAINT NO or <CR>,30001010    !
[0065] -           ,C       !=,SLOT11           !,00000400,$      !
[0066] -           ,SL      !SLOT100,F9050.1.R944,SLOT11,,FR944.S     !,T         !
[0067] -           ,C       !<>,SLOT100         !,30000600,$      !
[0068] -           ,DM      !SLOT11             !DOES NOT EXIST   !
[0069] -          *,GO      !                   !30001010         !
[0070] 30000600,IF          !R944,SLOT100,G944        !30001010          !
[0071] 30000650,IS          !SLOT30,,2          !ROUTING CODE CHANGE,30000650     !
[0072] -           ,C       !=,SLOT30           !,30000650,$      !
[0073] -           ,MOVE    !,,SLOT30,F9057.1.R944,SLOT100  !         !
[0074] 30000675,IS          !SLOT31,,6          !ROUTING DATE CHANGE,30000675     !
[0075] -           ,C       !=,SLOT31           !,30000675,$      !
[0076] -           ,MOVE    !,,SLOT31,F9058.1.R944,SLOT100  !         !
[0077] -           ,OM      !                   !ADD    !
[0078] -           ,CLEAR   !                   !                 !
[0079] 30000700,IS          !SLOT2,,1           !AGAIN (Y or N),30000700          !
[0080] -           ,C       !=,SLOT2            !Y,30001000,$     !

------------------------------------------------------------------------------

[0081] -           ,GO      !                   !00000400         !
!
!
[0082] 99999999,END         !        !          !
```

DEC
DEC 300
BAUD PRINTERS

LA34-AA EIA w/Forms . . . . $ 875
LA34-RA EIA, Receive Only 850
LA38-GA EIA, KP & Tractor 1,000
LA38-HA EIA, KP, Tractor & Stand 1,100
LA38-AA EIA, KP, Forms & Tractor 1,150

BRAND NEW * FULL 90 DAY WARRANTY

CASH PRICES ● IN STOCK
IMMEDIATE DELIVERY

CALL SONJA OR LAURIE AT:
(614) 889-0810

SCHERER'S
MINI COMPUTER MART
6145 Dolan Place Dublin, Ohio 43017

BRAND NEW*WARRANTY*ATD

CIRCLE 106 ON READER CARD

LETTERS to the RSTS Pro . . .

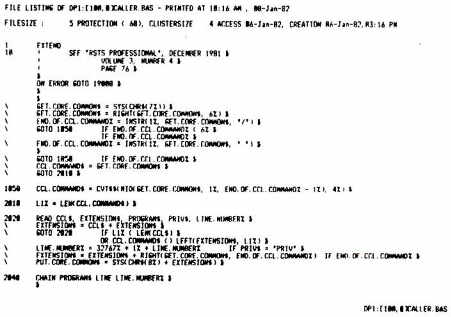
. . . continued from page 6

full length, e.g. SY/S is passed to $SYSTAT and SYSTAT/S. Some programs, such as $SWITCH and CALLER itself, make no allowance for abreviations.

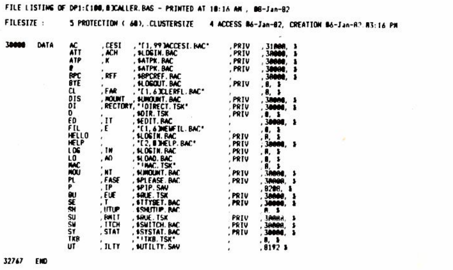
3. There is no equivalent of the PRIV specification in the CCL definition to ensure that the called program retains temporary privilege.

I enclose a listing of our version of CALLER.BAS, which works successfully on our SYSTIME 5000 computer, and which I believe overcomes these objections.

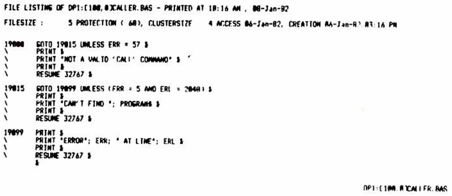
I have not been able in this version to incorporate calls to TECO.TEC and TYPE.TEC, as the "line number" in the CCL definition is not in these cases a true line number, but an indication of store requirements, and I have been unable to find a way of passing file specifications to TECO. Perhaps your correspondent, Mr. Fahey could advise me through your columns how he has managed to achieve this.



```
FILE LISTING OF DP1:[100,0]CALLER.BAS - PRINTED AT 10:16 AM , 08-Jan-82

FILESIZE :      5 PROTECTION < 60>, CLUSTERSIZE    4 ACCESS 06-Jan-82, CREATION 06-Jan-82,03:16 PM

1       EXTEND
10      !       SEE "RSTS PROFESSIONAL", DECEMBER 1981 &
        !               VOLUME 3, NUMBER 4 &
        !               PAGE 76 &
        &
        ON ERROR GOTO 19000 &
        &
        &
\       GET.CORE.COMMON$ = SYS(CHR$(7%)) &
\       GET.CORE.COMMON$ = RIGHT(GET.CORE.COMMON$, 6%) &
\       END.OF.CCL.COMMAND% = INSTR(1%, GET.CORE.COMMON$, "/") &
\       GOTO 1050       IF END.OF.CCL.COMMAND% < 6% &
                        IF END.OF.CCL.COMMAND% &
\       END.OF.CCL.COMMAND% = INSTR(1%, GET.CORE.COMMON$, " ") &
        &
\       GOTO 1050       IF END.OF.CCL.COMMAND% &
\       CCL.COMMAND$ = GET.CORE.COMMON$ &
\       GOTO 2010 &

1050    CCL.COMMAND$ = CVT$$(MID(GET.CORE.COMMON$, 1%, END.OF.CCL.COMMAND% - 1%), 4%) &

2010    LI% = LEN(CCL.COMMAND$) &

2020    READ CCL$, EXTENSION$, PROGRAM$, PRIV$, LINE.NUMBER% &
\       EXTENSION$ = CCL$ + EXTENSION$ &
\       GOTO 2020       IF LI% < LEN(CCL$) &
                        OR CCL.COMMAND$ <> LEFT(EXTENSION$, LI%) &
\       LINE.NUMBER% = 32767% + 1% + LINE.NUMBER%       IF PRIV$ = "PRIV" &
\       EXTENSION$ = EXTENSION$ + RIGHT(GET.CORE.COMMON$, END.OF.CCL.COMMAND%)  IF END.OF.CCL.COMMAND% &
\       PUT.CORE.COMMON$ = SYS(CHR$(8%) + EXTENSION$) &

2040    CHAIN PROGRAM$ LINE LINE.NUMBER% &
        &

                                                                DP1:[100,0]CALLER.BAS
```



```
FILE LISTING OF DP1:[100,0]CALLER.BAS - PRINTED AT 10:16 AM , 08-Jan-82

FILESIZE :      5 PROTECTION < 60>, CLUSTERSIZE    4 ACCESS 06-Jan-82, CREATION 06-Jan-82 03:16 PM

30000   DATA    AC      ,CESI   ,"[1,99]ACCESI.BAC"     ,PRIV   ,31000, &
                ATT     ,ACH    ,$LOGIN.BAC             ,PRIV   ,30000, &
                ATP     ,K      ,$ATPK.BAC              ,PRIV   ,30000, &
                @       ,       ,$ATPK.BAC              ,PRIV   ,30000, &
                BPC     ,REF    ,$BPCREF.BAC            ,       ,30000, &
                BYE     ,       ,$LOGOUT.BAC            ,PRIV   ,0, &
                CL      ,EAR    ,"[1,6]CLERFL.BAC"      ,       ,0, &
                DIS     ,MOUNT  ,$UMOUNT.BAC            ,PRIV   ,30000, &
                DI      ,RECTORY,"!DIRECT.TSK"          ,PRIV   ,30000, &
                D       ,       ,$DIR.TSK               ,PRIV   ,0, &
                ED      ,IT     ,$EDIT.BAC              ,       ,30000, &
                FIL     ,E      ,"[1,6]NEWFIL.BAC"      ,       ,0, &
                HELLO   ,       ,$LOGIN.BAC             ,PRIV   ,0, &
                HELP    ,       ,"[2,0]HELP.BAC"        ,PRIV   ,30000, &
                LOG     ,IN     ,$LOGIN.BAC             ,PRIV   ,0, &
                LO      ,AD     ,$LOAD.BAC              ,PRIV   ,0, &
                MAC     ,       ,"!MAC.TSK"             ,       ,0, &
                MOU     ,NT     ,$UMOUNT.BAC            ,PRIV   ,30000, &
                PL      ,EASE   ,$PLEASE.BAC            ,PRIV   ,30000, &
                P       ,IP     ,$PIP.SAV               ,       ,8200, &
                QU      ,EUE    ,$QUE.TSK               ,PRIV   ,30000, &
                SE      ,T      ,$TTYSET.BAC            ,PRIV   ,30000, &
                SH      ,UTUP   ,$SHUTUP.BAC            ,       ,0, &
                SU      ,BMIT   ,$QUE.TSK               ,PRIV   ,30000, &
                SW      ,ITCH   ,$SWITCH.BAC            ,PRIV   ,30000, &
                SY      ,STAT   ,$SYSTAT.BAC            ,PRIV   ,30000, &
                TKB     ,       ,"!TKB.TSK"             ,       ,0, &
                UT      ,ILTY   ,$UTILTY.SAV            ,       ,8192 &

32767   END
```



```
FILE LISTING OF DP1:[100,0]CALLER.BAS - PRINTED AT 10:16 AM , 08-Jan-82

FILESIZE :      5 PROTECTION < 60>, CLUSTERSIZE    4 ACCESS 06-Jan-82, CREATION 06-Jan-82 03:16 PM

19000   GOTO 19015 UNLESS ERR = 57 &
\       PRINT &
\       PRINT "NOT A VALID 'CALL' COMMAND" &
\       PRINT &
\       RESUME 32767 &

19015   GOTO 19099 UNLESS (ERR = 5 AND ERL = 2040) &
\       PRINT &
\       PRINT "CAN'T FIND "; PROGRAM$ &
\       PRINT &
\       RESUME 32767 &

19099   PRINT &
\       PRINT "ERROR"; ERR; " AT LINE"; ERL &
\       PRINT &
\       RESUME 32767 &
        &

                                                                DP1:[100,0]CALLER.BAS
```

Yours etc., Graham Clarke
Computer System Manager, Gloverall Ltd

Thank you, Graham. We'll try to publish a reply from Stephen Fahey.

—

Thank you for a fascinating and very useful publication. I follow the "MACRO MAN" articles with special interest as the first exposure to RSTS I had was converting a large typesetting suite (in MACRO-11) from RSX to RSTS. As a matter of interest, following the article on disk I/O from MACRO, the RSX macros for file handling — FCS (or File Control Services), which handle file opens (closes & blocking/deblocking of records from MACRO-11 without too much user brainwork, are available and appear to work under RSTS V7.0; they are all documented in the appropriate RSX manual, so I won't describe them here. Here also, a warning, anyone who is planning to convert some RSX code to RSTS should be aware that the RSX system MACRO library (LB:RSXMAC.SML) which is distributed with RSTS contains code for **all** the RSX monitor directives, including those which do not work at all under RSTS. Your code therefore assembles & TKB's without error, but will not run. I would suggest using LBR to remove those directives which don't work from the library so that MAC's fail. Further, if like me, you write all your mainline code in the blank PSECT, using the COMMON.MAC prefix file without prefixing your code with a .PSECT directive, the MAC works o.k., but TKB fails with an "Invalid load address in segment NNNNN", where NNNNNN is the PSECT name. I changed COMMON.MAC, and also changed it to give listing defaults the same as the RSX MACRO-11 assembler, don't worry, this doesn't affect RSTS sysgens.

Lastly (and best-ly, for us RSX types), the $EDMSG output string editor and .TPARS command line passes are both present (in LB:SYSLIB.OLB), as are the $SAVRG register save/restore co-routines. All these goodies are documented in the RSX System Library Routines manual.

Keep up the good work!

Hugh J.E. Davies, B.Sc.
Hertfordshire, England

*Thanks Hugh, and keep up **your** good work, also.*

—

I just finished reading the Dec. 1981 issue and was interested in the 'CALLER.BAS' article. Congratulations are due to Steven Fahey for coming up with his solution to free up small buffers.

For a lot of systems, this method would be more than suitable. However, systems with a large number of CCL's and a high amount of activity may find some disadvantages.

1. Two programs (CALLER) plus the normally executed program are run each time a CALL command is entered. The running of an extra program will create some additional overhead in starting the program, disk access and swapping (unless free memory is available).

2. Abbreviated commands (ie. SY-STAT, QU-EUE) cannot be used unless a separate entry is included for each possible abbreviation. This is limited only by the maximum size to which the program can be expanded. The more entries, the more overhead in finding the correct entry sequentially. In the case of QUE, anything but CALL QUE will not work.

3. Any batch jobs or programs containing CCL commands will have to be changed.

For non-privileged users you may or may not want the called program to retain its compiled

. . . continued on page 92

TIPS & TECHNIQUES

A Column For The Advanced RSTS/E User

By Steven L. Edwards, Software Techniques

TURNING COUSINS INTO RUN-TIME SYSTEMS

Making RTS's from BP2 programs.

A few issues back we discussed the benefits of creating multi-user tasks out of some of the CUSP's. We also mentioned that the limiting factor in creating sharable (multi-user) tasks was that the resident libraries generated had to be added to load at specific memory addresses. On systems with small amounts of physical memory, this means that only 1 or 2 CUSP's can be made sharable. On systems with large amounts of physical memory, this means that you have to fragment memory.

The long term solution is for DEC to allow us to add resident libraries without specifying the load address like we can with run-time systems. The short term solution is to create run-time systems from the MAC files generated by the BASIC-PLUS-2 compiler. The short term solution is the topic for this column.

The general flow of the procedure to change a BASIC-PLUS-2 program into a run-time system is to:

- Compile the source program into MACRO.
- Eliminate all of the funny control characters the BASIC-PLUS-2 compiler leaves behind. When the compiler generates it's MAC file it comments the code for literal strings with the literal string. Thus the comment for 'A$ = SYS(CHR$(6) + CHR$(26)...' is "CTRL-F CTRL-Z," which will confuse the MACRO assembler, so I use EDT V2 to eliminate all quoted strings from the file. (An SPR has been submitted.)
- Run the program BP2RTS (included in this column) which will separate the read-only code from the read-write code, and generate an ATPK command file. Note that you have to take a guess at the size the run-time system will be. The program defaults to a guess of 16KW.
- Execute the command file, which will:
  - Assemble the 2 macro source files generated by the program.
  - Link the read-write object module.
  - Link the read-only object module including the read-write symbol table.
  - Generate the run-time system from the read-only task using MAKSIL. (MAKSIL has a bug in it that prevents it from creating run-time systems larger than 16 KW, an SPR has been submitted, and a patch is included.)
  - ADD the read-only run-time system.
  - Link the read-write object module including the read-only symbol table.
  - Name the read-write task to the read-only run-time system.
  - Delete the files created that are no longer needed.
- Run the read-write task to make sure everything still works.
- Add the commands needed to add the read-only run-time system to your start-up command files.

NOTE

This procedure creates several files. These files are: RO.*, RW.*, RO1.CMD, and cuspname.CMD

Using this procedure we have 'converted' ATPK, SYSTAT, BATRUN, and SPLRUN into sharable run-time systems with the following results:

	R/W size	R/O (RTS) size
ATPK	3KW	13KW
SYSTAT	2KW	17KW
BATRUN	4KW	20KW
SPLRUN	4KW	21KW

These programs were 'converted' because of the high probability that more than one copy may be running at the same time.

This procedure, while not exactly a 'clean' procedure does accomplish the goal of allowing BASIC-PLUS-2 programs to share their read-only segments of code. Good luck.

```
>FIL PAT:MAKSIL.BAS,LB1:MAKSIL.BAS

Comparing: 1) PAT:MAKSIL.BAS to 2) LB1:MAKSIL.BAS

******************************
1) PAT:MAKSIL.BAS
        \ LOWCODE%=FNSHFT.RGT%(FNSHFT.RGT%(-(L.BXFR% AND (-2047%)),1%) &
                AND 32767%,5%) &
        \ IF (((L.BSA% OR L.BXFR%) AND 1%)=1%) THEN &
**********
2) LB1:MAKSIL.BAS
        \ LOWCODE%=((-(L.BXFR% AND (-2047%))/2%) AND 32767%)/32% &
        \ IF (((L.BSA% OR L.BXFR%) AND 1%)=1%) THEN &
******************************
1) PAT:MAKSIL.BAS
        \ TOP%=FNSHFT.RGT%(LOWCODE%+31%,5%)*32% &
        \ KWORDS%=FNSHFT.RGT%(TOP%,5%) &
        \ O.SIZE%=32%-(((KWORDS%+3%)/4%)*4%) &
**********
2) LB1:MAKSIL.BAS
        \ TOP%=((LOWCODE%+31%)/32%)*32% &
        \ KWORDS%=TOP%/32% &
        \ O.SIZE%=32%-(((KWORDS%+3%)/4%)*4%) &
******************************
1) PAT:MAKSIL.BAS
        \ HILOC%=FNSHFT.RGT%(L.BMXV%-L.BSA$,1%)-(BASE%*32%) &
        \ TOP.BLOCK%=(TOP%/8%) + 1% &
**********
2) LB1:MAKSIL.BAS
        \ HILOC%=((L.BMXV%-L.BSA%)/2%)-(BASE%*32%) &
        \ TOP.BLOCK%=(TOP%/8%) + 1% &
******************************
1) PAT:MAKSIL.BAS
                AND TOP.BLOCK%+FNSHFT.RGT%(BASE%,3%)+1% = TSK.FILE.SIZE% THEN &
                   IF FNWORD%(HILOC%-1%) <= 32% &
**********
2) LB1:MAKSIL.BAS
                AND TOP.BLOCK%+(BASE%/8%)+1% = TSK.FILE.SIZE% THEN &
                   IF FNWORD%(HILOC%-1%) <= 32% &
******************************
1) PAT:MAKSIL.BAS
        \ PRINT LIB$;" will load in a"; FNSHFT.RGT%(PARSIZE%,11%); &
                          "K-word partition using"; &
                          FNSHFT.RGT%(PARSIZE%,11%)-STACK%/1024%; &
                          "K-words physical memory." &
**********
2) LB1:MAKSIL.BAS
        \ PRINT LIB$;" will load in a"; PARSIZE%/2048%; &
                          "K-word partition using"; &
                          PARSIZE%/2048%-STACK%/1024%; &
                          "K-words physical memory." &
******************************
1) PAT:MAKSIL.BAS
15940   FNEND &
15950   DEF* FNSHFT.RGT%(X%,Y%)= &
        (X% AND 32767%)/(2%^Y%) OR ((2%^(15%-Y%)) AND (X%<0%)) &
        ! FUNCTION TO SHIFT RIGHT A 16-BIT INTEGER (X%), BY (Y%) BITS. &
**********
2) LB1:MAKSIL.BAS
15990   FNEND &

?6 Differences Found.
```

. . . continued on page 79

You Deserve An A+ In Accounting

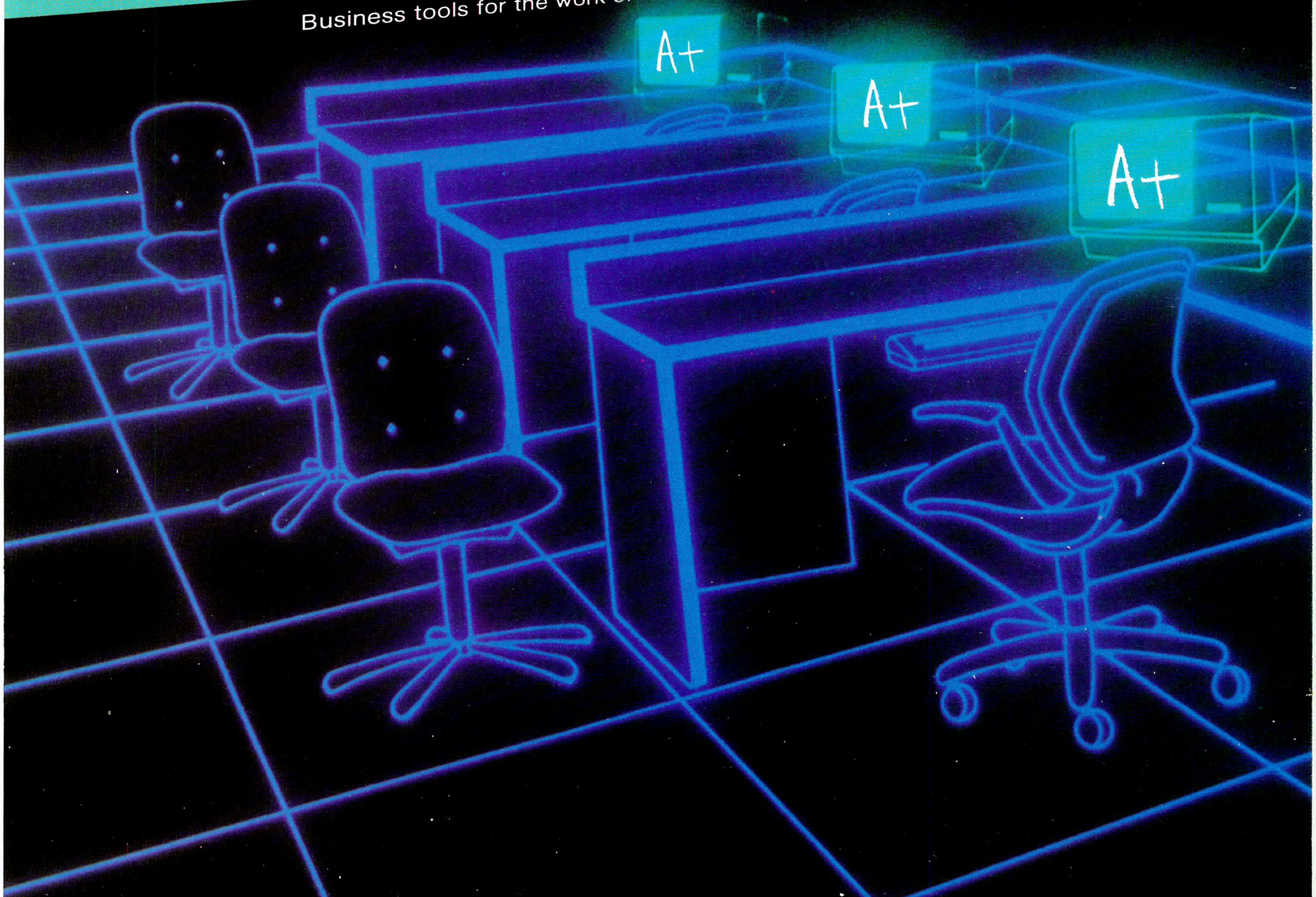
Software Techniques announces A+: State-of-the-art accounting software for the PDP-11. A system so flexible it adapts to the way you do business, not the other way around.

A+ has multi-company/multi-division capability, comprehensive financial reporting, and complete documentation, plus an optional English-language report writer.

If you'd like to see more, ask us to send you a demo. We think you'll agree it's an A+.

Software
Techniques
Incorporated

Business tools for the work of Man.



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

74/76 Northbrook Street
Newbury, Berkshire RG13 1AE
United Kingdom
Phone: 44 (0) 635 30840

CIRCLE 65 ON READER CARD

THE RSTS/E ENVIRONMENT

By Michael H. Koplitz

The RSTS/E environment is made up of three parts: addressing, the low segment of the task, and the high segment of the task. Each of these areas will be dealt with in this article.

ADDRESSING

There are three sets of Active Page Registers (APR) on the PDP-11/70 and 11/45 (two on other types of PDP-11s), kernel mode APRs, user mode APRs, and supervisor mode APRs. The Monitor uses the kernel mode APRs to map itself into memory. The user APRs map the user task into memory. The APR is actually a pair of sixteen-bit registers, the page address register (PAR) and the page descriptor register (PDR).

The page address register defines where the page actually begins in the memory (starting address). The page descriptor register defines the maximum length of the page and how it can be accessed (read or write, read only, etc.)

The sixteen-bit address generated when a program is compiled is treated as a relocatable (virtual) address. It defines which one of the active page registers is to be used to calculate a physical address. It also contains the byte offset within the page.

The PAR of the APR is handled as though it contains bits six through twenty one (bits six through seventeen for PDP-11s other than 11/70 and 11/45) of the 22-bit (or 18-bit) physical address, which is the starting address of the page. The PAR is combined with the byte offset within the page from the virtual address to get the physical address.

```
                 Virtual Address

 15   13 12                          0
 ----------------------------------
 !APR   ! byte offset with in page!
 ----------------------------------
 pointer   virtual address
 to APR


             Page Address Register

    ----------------------------
    ! starting address of page !
    ----------------------------
```

Addresses Up To 32KW

decimal	octal	binary (slash inserted between bits twelve and thirteen)
0KW - (4KW-1)	000000 - 017776	000/0000000000000 - 000/1111111111110
4KW - (8KW-1)	020000 - 037776	001/0000000000000 - 001/1111111111110
8KW - (12KW-1)	040000 - 057776	010/0000000000000 - 010/1111111111110
12KW - (16KW-1)	060000 - 077776	011/0000000000000 - 011/1111111111110
16KW - (20KW-1)	100000 - 117776	100/0000000000000 - 100/1111111111110
20KW - (24KW-1)	120000 - 137776	101/0000000000000 - 101/1111111111110
24KW - (28KW-1)	140000 - 157776	110/0000000000000 - 110/1111111111110
28KW - (32KW-1)	160000 - 177776	111/0000000000000 - 111/1111111111110

Bits thirteen through fifteen determine which APR (zero through seven) to use to calculate the physical address. Bits zero through twelve are the offset into the page. This offset is added to the PAR of the APR to determine the physical memory address.

Example: Take virtual address 72322 octal and convert it to a physical address, APR 3 is 1460 octal.

72322 (octal) virtual address gives:

APR = 3
Offset = 12322

12322
1460
160322 octal, the physical address in the memory.

The byte offset into the page from the virtual address is thirteen bytes long. This allows addressing of 4096 words, 4KW. An APR therefore maps 4KW and there are eight APRs so 4KW * 8APR = 32KW program size.

LOW SEGMENT OF A JOB

The first one thousand bytes of the user task have special meanings to the Monitor. So the 32KW task area is shortened by one thousand bytes. The figure below indicates what information is contained in this region of the low segment.

```
            The First 1000 Bytes
 ----------------------------------- 0
 !contolled by job -- user job image!
 ! or run-time system               !
 ----------------------------------- 60
 !used by the monitor for job con-  !
 ! text information to make job     !
 ! swappable                        !
 ----------------------------------- 110
 !used by the monitor for hardware  !
 ! floating point context infor-    !
 ! mation to make job swappable     !
 ----------------------------------- 170
 !default SP stack area             !
 ----------------------------------- 400
 !keyword    (KEY bits 8 - 15)      !
 !           (USRSP bits 0 - 7)     !
 ----------------------------------- 402
 !file request queue block (FIRQB)  !
 ----------------------------------- 442
 !transfer request block (XRB)      !
 ----------------------------------- 460
 !core common area (CORCMN)         !
 ----------------------------------- 660
 !controlled by job                 !
 ----------------------------------- 734
 !user-assignable PPN (USRPPN)      !
 ----------------------------------- 736
 !user-assignable default protection!
 ! code (USRPRT)                    !
 ----------------------------------- 740
 !user logical device name table    !
 ! (USRLOG)                         !
 ----------------------------------- 776
```

GENERAL DESCRIPTION

KEY — (bits eight through fifteen of the keyword) this byte defines the job's status in the RSTS/E environment. The keyword is refreshed by the monitor at different points during the timesharing session. The defined bits of the KEY are listed below:

JFLOCK Bit 14 — when one indicates that the job does not wish to be swapped.

JFBIG Bit 13 — when one indicates that the job can exceed its private memory maximum.

JFNOPR Bit 12 — when one indicates that the job is not logged in yet.

JFSYS Bit 11 — when one indicates that the job is running with temporary privileges.

JFPRIV Bit 10 — when one indicates that the job has permanent privileges.

JFFPP Bit 9 — when one indicates that the contents of the hardware floating point unit should be part of the context of this job.

JFSPRI Bit 8 — when one indicates that the job is running with the special run priority at ½ level higher than normal.

USRSP — (bits zero through seven of the keyword) is assigned a value of 400 (by COMMON.MAC). The Monitor automatically loads this value into the stack pointer register (R6) when a job is created.

FIRQB — the file request queue block is the main communication area between the Monitor and the job for Monitor directives that involve file or device operations. Below is a diagram of the FIRQB area.

```
                      FIRQB
---------------------------------------- 0
!unused          !return status        !
!                !  (FIRQB)            !
---------------------------------------- 2
!CALFID/.UUO sub !job number * 2       !
! func. (FQFUN)  ! (FQJOB)             !
---------------------------------------- 4
!MSB of file size!channel number * 2   !
! (FQFIL)        ! (FQERNO)            !
---------------------------------------- 6
!project and programmer number         !
!      (FQPPN)                         !
---------------------------------------- 10
!filename (2 words in Radix-50 format) !
!      (FQNAM1)                        !
---------------------------------------- 14
!file extension (in Radix-50 format)   !
!      (FQEXT)                         !
---------------------------------------- 16
!least significant bits of file size   !
!      (FQSIZ)                         !
---------------------------------------- 20
!buffer length (FQBUFL)                !
---------------------------------------- 22
!mode (FQMODE)                         !
---------------------------------------- 24
!status flag (FQFLAG)                  !
---------------------------------------- 26
!protection code !<> 0, prt. real code !
! (FQPROT)       !                     !
---------------------------------------- 30
!device name (two ASCII characters)    !
!      (FQDEV)                         !
---------------------------------------- 32
!<> 0, unit no.  !device unit no.      !
! real           !  (FQDEV)            !
---------------------------------------- 34
!cluster size (FQCLUS)                 !
---------------------------------------- 36
!# of entries in directory lookup      !
!      (FQNENT)                        !
----------------------------------------
```

XRB — is the main communication area between the Monitor and the user for Monitor directives handling file or device input/output. Below is a figure of the XRB.

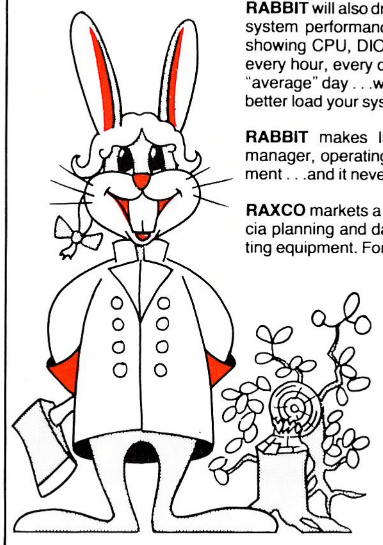
```
               XRB
------------------------------ 0
!buffer size in bytes (XRLEN)!
------------------------------ 2
!bytes actually transfered   !
! (XRBC)                     !
------------------------------ 4
!buffer address (XRLOC)      !
------------------------------ 6
!MSB block #   !channel number!
! (XRBLKM)     ! * 2 (XRCI)  !
------------------------------ 10
!least significant bits of   !
! the block number (XRBLK)   !
------------------------------ 12
!wait time for terminals     !
! (XRTIME)                   !
------------------------------ 14
!device modifier (XRMCD)     !
------------------------------
```

CORCMN — this is core common which is used as a common data exchange area when it is

VAX & RSTS/E (V. 7) USERS

OUR RABBIT SYSTEM ALWAYS TELLS THE TRUTH ABOUT YOUR COMPUTER

Like who is using it, when, where, what resources, and how much . . . all in great detail or summarized - your choice.



RABBIT will give you the most complete set of user accounting info you've ever seen — complete, detailed information for each user session. It even creates invoices, too, if you wish.

RABBIT will also draw a picture worth a 1000 words about your system performance. In fact it will draw you lots of pictures showing CPU, DIO, PAGE FAULTS (and the like) consumed every hour, every day, every week. It'll graphically depict your "average" day . . . with or without your biggest users so you can better load your system for peak response and throughput.

RABBIT makes life easier for the system user, system manager, operating management and the accounting department . . . and it never tells a lie.

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

RAXCO INC.

Suite 200
6520 Powers Ferry Rd.
Atlanta, GA 30339
(404) 955-2553

CIRCLE 110 ON READER CARD

necessary to exchange lengthy data between the monitor and the job, or between programs running under the same job number.

USRPPN — the project-programmer number used when an "@" is used in the file string scan.

USRPRT — the protection code default used in the file string scan.

USRLOG — the user's private logical device name table, three to four logical names can be stored here. A figure for this area if given below.

```
           USRLOG
                              offset
------------------------ 0
!logical device name   !
! in Radix-50          !
------------------------ 2
!physical name in two  !
! ASCII characters     !
------------------------ 4
!real unit !unit number!
! number   !           !
------------------------ 6
```

HIGH SEGMENT OF A JOB

The run-time system associated with the job is located in the high segment of the job task area. The run-time system takes up multiples of 4K words of virtual address space, due to APR mapping. The BASIC-PLUS run-time system can be generated to take up 13K words of memory but when it is used the user area is not increased by 3KW due to APR mapping (discussed in more detail later).

The Monitor uses certain areas of the high segment to get information from the job defining what work the Monitor is to do for the job, and to pass information to the job. The run-time system sets this area with entry points and values to define itself to the Monitor.

Also contained in the high segment of the task is the pseudo vector. The pseudo vector is used for the run-time system and the Monitor to communicate. In general, the pseudo-vector region contains the following:

1. Values and flags which define the capabilities of the run-time system to the Monitor.

2. Addresses pointing to locations within the run-time system where the monitor is to pass control when certain conditions occur.

DEC BEST VALUES

PRE-OWNED DEC EQUIPMENT

BUYING AND SELLING

SYSTEMS • CPU's • PERIPHERALS • TERMINALS
OPTIONS • MEMORY • COMPATIBLES

CALL DICK BAKER (305) 979-2844

dataware incorporated
Carico Center
2845 NW 62nd Street
Ft. Lauderdale, Florida 33309
Telephone (305) 979-2844

CIRCLE 49 ON READER CARD

```
Format of the Pseudo-Vector Region of the High Segment
----------------------------------------------------
!flags describing the run-time system (P.SIZE)     !  177732
----------------------------------------------------
!normal executable file extension (P.DEXT)         !  177734
----------------------------------------------------
!(former use now obsolete -- reserved word) (P.ISIZ) !  177736
----------------------------------------------------
!minimum size, in K words, of user job image (P.MSIZ)!  177740
----------------------------------------------------
!trap address for FIS hardware floating point option !  177742
!               (P.FIS)                            !
----------------------------------------------------
!crash entry point (default run-time system only)  !  177744
!               (P.CRAS)                           !
----------------------------------------------------
!start entry point (default run-time system only)  !  177746
!               (P.STRT)                           !
----------------------------------------------------
!entry point for new user (P.NEW)                  !  177750
----------------------------------------------------
!entry point for new user with program to run (P.RUN)!  177752
----------------------------------------------------
!trap address for various "bad" errors (P.BAD)     !  177754
----------------------------------------------------
!trap address for BPT instruction and T-bit traps  !  177756
!               (P.BPT)                            !
----------------------------------------------------
!trap address for IOT instruction (P.IOT)          !  177760
----------------------------------------------------
!trap address for non-Monitor EMT instructions(P.EMT)!  177762
----------------------------------------------------
!trap address for all TRAP instructions (P.TRAP)   !  177764
----------------------------------------------------
!trap address for FPP or FPU floating point units  !  177766
!               (P.FPP)                            !
----------------------------------------------------
!trap address when user types one CTRL-C (P.CC)    !  177770
----------------------------------------------------
!trap address when user types two CTRL-C (P.2CC)   !  177772
----------------------------------------------------
!maximum size (in K words) or user job image (P.SIZE)!  177774
----------------------------------------------------
!**********reserved for future use*******************!  177776
----------------------------------------------------
```

GENERAL DESCRIPTION

P.FLAG — this word is set with flags which define the capabilities of the run-time system to the monitor.

P.DEXT — the default runnable file extension, when a .RUN is executed and an extension is not given, P.DEXT is used.

P.MSIZ — minimum size for a user job in K word for this run-time system.

P.SIZE — maximum size that a user job image can be for this run-time system.

P.FIS — trap address for the hardware floating point instruction set.

P.BAD — Monitor passes control on to the run-time system at the location specified by P.BAD when the following synchronous traps occur:

1) Memory management unit exception

2) Job tries to execute a reserved instruction

3) Job issues an instruction with an odd address

P.BPT — contains the trap address for a BPT instruction and for T-bit traps.

P.IOT — contains the trap address for an IOT instruction.

P.EMT — contains the location to which control is transferred for non-Monitor EMT instructions.

P.TRAP — contains location to which control is transferred for all TRAP instructions.

P.FPP — contains trap address for FPP (or FPU) hardware floating point units.

P.CC — contains location to which control passes when the user types a ↑C.

P.2CC — contains location to which control passes when the user types a second ↑C.

P.CRAS — entry points used only by the system default run-time system.

P.NEW — Monitor passes control to this entry point under the assumption that "new user" or "next request" processing is to be done, as opposes to the P.RUN entry point, where it is known that a specific program is to be run under this run-time system.

P.RUN — Monitor passes control to this entry point when an executable program is to be run for a job under control of this run-time system.

AFTERTHOUGHTS

It is known that RSTS/E has a limitation of 31KW for user tasks **NOT 32KW** as prescribed by the use of APRs. The explanation given (by Digital) for this is that there is a problem with the fifteenth bit of the address being used as a sign bit. **How can this bit be used as a sign bit if the bits thirteen through fifteen are used to determine the APR?** If the physical address is always created by sending the virtual address to memory mangement then the 32KW of memory MUST always be addressable since RSTS/E would not be concerned with physical addressing. Since the APRs must be used in address calculations there must be some other reason why the 32KW of memory cannot be accessed.

The 32KW of memory is used to communicate between the Monitor and run-time system. This reserved area may prevent the user task from growing into that last kilo-word of memory because the Monitor would start to interupt the words in the last kilo-word of the task as run-time system entry points and status words. This may explain why RSTS/E will not allow the user task to grow past 31KW.

When using a run-time system, for example BASIC-PLUS, a 16KW run-time system, the user task never exceeds 16KW of memory. The run-time system is mapped by 4 APRs. The user task would be mapped by 4 APRs.

```
     16KW BASIC-PLUS Run-time System

     APR0 ------------------------
          !                      !
     APR1 !                      !
          !   user task area     !
     APR2 !      16KW            !
          !                      !
     APR3 !                      !
          !                      !
          ------------------------
          !                      !
          !   other memory       !
          !                      !
     APR4 ------------------------
          !                      !
     APR5 !                      !
          !                      !
     APR6 !  BASIC-PLUS RTS      !
          !        16KW          !
     APR7 !                      !
          !                      !
          ------------------------
                  memory
```

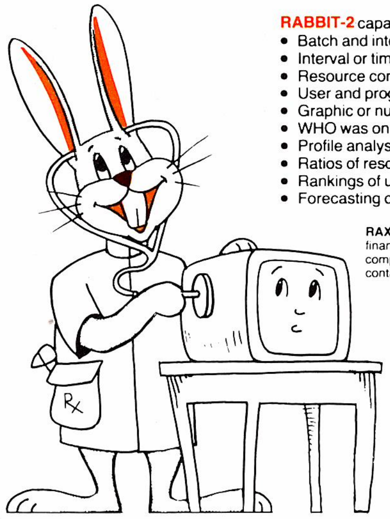
In this circumstance all 32KW are being used.

SYSTEM PERFORMANCE ANALYSIS FOR VAX AND RSTS/E USERS

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

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

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



RABBIT-2 capabilities include:

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

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

RAXCO INC.

Suite 200
6520 Powers Ferry Rd.
Atlanta, GA 30339
(404) 955-2553

CIRCLE 33 ON READER CARD

With the introduction of disappearing RSX the entire job area could be used for the user task. The problem of the user task accessing the top 1KW of the task (whatever it may be) becomes important. The 31KW job size maximum is established because RSTS/E cannot go to 32 KW for the user task. In conclusion it is discovered that the 32KW task image that RSTS/E promises is reduced by 2KW. The first one thousand bytes are preassigned by the monitor and the last kilo-word RSTS/E cannot access due to some secret internal problem. The 32KW of the user task cannot be accessed and Digital has indicated that it is a problem dealing with the sign bit. With the information presented here it seems incorrect to say that there is a sign bit problem, but rather the fixed locations in the high segment of memory must be used only by a run-time system and not the user task, because these locations have special meanings to the monitor.

Run-time systems must be mapped by the APRs. When a run-time system takes up less than a multiple of 4KW, the memory to the next multiple of 4KW is lost. In other words, if the BASIC-PLUS run-time system is generated with a size of 13KW, the 3KW to the next lower boundary (run-time systems are loaded from the high segment down to the low segment) is lost.

```
     13KW BASIC-PLUS Run-time System

     APR0 --------------------
          !                  !
     APR1 !                  !
          ! 16KW user task   !
     APR2 !                  !
```

RSTS
Application
Software

Accounting Two:
General Ledger
Accounts Receivable
Accounts Payable

Flexible reporting structure
System adapts to you

Table driven software
You make changes easily

Report formatter
End user creates report

Modular and integrated
You enter data only once

Menu selections
Easy to use

On-line support
Free-up your programming staff

Proven successful in Canada and USA
It works now

IAS
Computer
Corporation
Limited

6080 Young Street
Suite 401
Halifax, Nova Scotia
B3K 5L2

Contact: Pat Hunt (902) 453-4620

CIRCLE 112 ON READER CARD

```
       !                    !
 APR3  !                    !
       !                    !
       ----------------------
       !                    !
       !                    !
       ! other memory       !
       !                    !
 APR4  ----------------------
       ! unusable           !
       ----------------------
       ! beginning of RTS   !
 APR5  !                    !
       !                    !
 APR6  ! BASIC-PLUS RTS     !
       !     13KW           !
 APR7  !                    !
       !                    !
       ----------------------
               memory
```

APR4 contains the starting address of the 4KW segment that contains the first KW of this BASIC-PLUS run-time system. Therefore it is more efficient to generate a 16KW BASIC-PLUS run-time system so that the 3KW of memory is not wasted.

A 17KW BASIC-PLUS run-time system would cut the user task down to 12KW. An additional APR would be needed to map the run-time system. An APR cannot be used to map the run-time system and user task area at the same time.

```
   17KW BASIC-PLUS Run-time System

   APR0 -------------------
        !                 !
   APR1 !  User task 12KW !
        !                 !
   APR2 !                 !
        -------------------
        !                 !
        !                 !
        !  other memory   !
        !                 !
   APR3 -------------------
        ! unusable        !
        -------------------
        ! first 1KW of RTS!
   APR4 !                 !
        !                 !
   APR5 !                 !
        ! BASIC-PLUS RTS  !
   APR6 !     17KW        !
        !                 !
   APR7 !                 !
        -------------------
              memory
```

It can be seen from the above figure that a 17KW BASIC-PLUS run-time system reduces the overall task size down to 29KW, 3KW remember are unusable due to APR mapping. In conclusion it should be noted that it is useful to generate a run-time system to a 4KW boundary opposed to reducing it but not reducing it enough to reduce the use of an APR. Sizes of run-time systems must be examined to respect to their APR usage opposed to their memory usage.

TIPS & TECHNIQUES

. . . continued from page 72

```
1!      &
!       &
!       Title:          B P 2 R T S &
!       &
!       Description:    SPLIT BP2 .MAC INTO RO.MAC, AND RW.MAC &
!       &
!       Package:        In-House &
!       &
!       Version:        V7.0-01 &
!       &
!       Edit date:      21-JAN-82 &
!       &
!       Written by:     STEVEN L. EDWARDS &

11!     Copyright (C) 1982 &
!       Software Techniques &
!       Los Alamitos, CA 90720 &
!       &
!       Title  to and ownership of the software shall at all times remain &
!       in Software Techniques. &
!       &
!       The  information  in  this  document is subject to change without &
!       notice  and  should  not be construed as a commitment by Software &
!       Techniques. &
!       &
!       This  software  is  un-released  and  Software  Techniques has no &
!       commitment to support it at this time, unless stated elsewhere in &
!       writing. &

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

21!     V7.0-01         21-JAN-82       Initial conception. &

100!    &
!               General Description &
!       &

101!            THIS PROGRAM SPLITS THE MAC FILE GENERATED BY THE &
!       BASIC-PLUS-2 COMPILER INTO RO.MAC, AND RW.MAC.  THESE FILES CAN &
!       THEN BE ASSEMBLED.  RO.MAC CAN THEN BE MADE INTO A RUN-TIME &
!       SYSTEM. &

300!    &
!               Assembly instructions &
!       &

301!    OLD BP2RTS &
!       COM/OBJ &
!       BUI &
!       TKB @BP2RTS &

500!    &
!               Compile time variables &
!       &

501     .DEFINE .NAME$ = "Bp2rts" &
\       .DEFINE .VERSION$ = "V7.0-01" &
\       .DEFINE .CHAN.KB% = 1% &
\       .DEFINE .CHAN.IN% = 2% &
\       .DEFINE .CHAN.RO% = 3% &
\       .DEFINE .CHAN.RW% = 3% &
\       .DEFINE .CHAN.CF% = 3% &
\       .DEFINE .ASCII.L% = 76% &
\       .DEFINE .ASCII.N% = 78% &
!       &
!       Program name. &
!       Program version. &
!       Channel number for terminal I/O. &
!       Channel number for input file. &
!       Channel number for read-only output file. &
!       Channel number for read-write output file. &
!       Channel number for command file. &
!       ASCII value of 'L.' &
!       ASCII value of 'N.' &

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

901     DIM PAR_PARAM$(7%) &
!       &
!       Parameters for the TKB PAR directive. &

951     MAP     (FIRQB)            ! SYS() OFFSETS &
                 FQJOB.FQFUN%   ! BYTES 1 & 2 &
                ,FQFIL.FQSIZM%  ! BYTES 3 & 4 &
                ,FQPPN%         ! BYTES 5 & 6 &
                ,FQNAM1%        ! BYTES 7 & 8 &
                ,FQNAM2%        ! BYTES 9 & 10 &
                ,FQEXT%         ! BYTES 11 & 12 &
                ,FQSIZ%         ! BYTES 13 & 14 &
                ,FQBUFL%        ! BYTES 15 & 16 &
                ,FQMODE%        ! BYTES 17 & 18 &
                ,FQFLAG%        ! BYTES 19 & 20 &
                ,FQPROT%        ! BYTES 21 & 22 &
                ,FQDEV%         ! BYTES 23 & 24 &
                ,FQDEVN%        ! BYTES 25 & 26 &
                ,FQCLUS%        ! BYTES 27 & 28 &
                ,FQNENT%        ! BYTES 29 & 30 &
!       &
!       Map the Firqb data block. &
!       &
\       MAP     (FIRQB) &
                 FIRQB$ = 30%   ! FIRQB$ = SYS() &
!       &
!       Re-map the Firqb data block. &

999!    &
!               Start of Initialization &
!       &

1000    ONERROR GOTO 19000 &
!       &
!       Set standard error trap. &

1010    PRINT .NAME$ + HT + .VERSION$ + HT + "Software Techniques" &
                + CR + LF + "Split MAC into RO and RW." + CR + LF &
                UNLESS E0% &
!       &
!       Print standard header on 'RUN' entry. &

1030    READ PAR_PARAM$(TEMP_0%) &
                FOR TEMP_0% = 1% TO 7% &
!       &
!       Define various variables. &
```

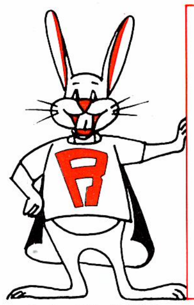
RABBIT-3

JOB ACCOUNTING and
PERFORMANCE MONITORING for
RSTS/E VERSION 7 USERS

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

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

RABBIT-3 is cheap. The basic system is available for $99/month. Furthermore you buy only the features you need to get your job done. Select from the options listed below:



AVAILABLE OPTIONS FOR
RABBIT-3 USERS

AUTO-CRASH RECOVERY
. . . automatic restart from system crashes.

DAILY DISK CATALOG
. . . generates disk accounting information for each user.

SECURITY TRACER RECORD
. . . provides step by step security information of user activities.

RAXCO INC.

Suite 200, 6520 Powers Ferry Road, Atlanta, GA 30339 U.S.A.
Telephone (404) 955-2553
Offices in Canada and the United States

CIRCLE 23 ON READER CARD

TERMINALS FROM TRANSNET

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

	DESCRIPTION	PURCHASE PRICE	PER MONTH 12 MOS.	24 MOS.	36 MOS.
DEC	LA36 DECwriter II	$1,095	$105	$ 58	$ 40
	LA34 DECwriter IV	995	95	53	36
	LA34 DECwriter IV Forms Ctrl.	1,095	105	58	40
	LA120 DECwriter III KSR	2,295	220	122	83
	LA120 DECwriter III RO	2,095	200	112	75
	VT100 CRT DECscope	1,695	162	90	61
	VT101 CRT DECscope	1,195	115	67	43
	VT125 CRT Graphics	3,295	315	185	119
	VT131 CRT DECscope	1,745	167	98	63
	VT132 CRT DECscope	1,995	190	106	72
	VT18XAC Personal Computer Option	2,395	230	128	86
TEXAS INSTRUMENTS	TI745 Portable Terminal	1,595	153	85	58
	TI765 Bubble Memory Terminal	2,595	249	138	93
	TI Insight 10 Terminal	695	67	37	25
	TI785 Portable KSR, 120 CPS.	2,395	230	128	86
	TI787 Portable KSR, 120 CPS	2,845	273	152	102
	TI810 RO Printer	1,695	162	90	61
	TI820 KSR Printer	2,195	211	117	80
LEAR SIEGLER	ADM3A CRT Terminal	595	57	34	22
	ADM5 CRT Terminal	645	62	36	24
	ADM32 CRT Terminal	1,165	112	65	42
	ADM42 CRT Terminal	1,995	190	106	72
DATAMEDIA	EXCEL 12 CRT Terminal	1,695	162	90	61
	EXCEL 42 Smart Buffered CRT	995	96	54	36
	COLORSCAN 10 Color CRT	3,195	307	171	116
TELEVIDEO	925 CRT Terminal	850	82	46	31
	950 CRT Terminal	1,075	103	57	39
NEC SPINWRITER	Letter Quality, 7715 RO	2,895	278	154	104
	Letter Quality, 7725 KSR	3,295	316	175	119
GENERAL ELECTRIC	2030 KSR Printer 30 CPS	1,195	115	67	43
	2120 KSR Printer 120 CPS	2,195	211	117	80
HAZELTINE	Executive 80/20	1,345	127	75	49
	Executive 80/30	1,695	162	90	61
EPSON	MX-80 F/T Printer	745	71	42	27
	MX-100 Printer	895	86	48	32
TIMEPLEX	E0400 4 Channel Stat Mux	1,525	147	82	55
	E0800 8 Channel Stat Mux	2,050	197	110	74

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

MICROCOMPUTERS
APPLE • COMMODORE • HP85 • DEC LSI 11

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

TRANSNET CORPORATION
1945 ROUTE 22 • UNION, N.J. 07083 • (201) 688-7800
TWX 710-985-5485 800-526-4965 OUTSIDE N.J.

CIRCLE 28 ON READER CARD

ELECTRONIC MAIL. PRACTICALLY SPEAKING.

Sooner or later you will be using electronic mail. It just makes good sense. When you do, you will want a system that is complete—a delivery system, a scheduling system, and an information manager. Your electronic mail system will become an essential part of your office environment. INTECOM is such an electronic mail system.*

INTECOM's power is easy to control. It relates to the way you work. Electronic IN, OUT, and HOLD baskets are just what you would expect. You can scan your IN basket, selecting only those message subjects you wish to read. Or, you can place a message into your HOLD basket for a number of days to have it automatically reappear in your IN basket on the appointed day. You can even have INTECOM recall specific messages by providing your own selection criteria. Replying, forwarding, and sending to groups are as easy as can be. And these are just a few of the features in store for you.

You owe yourself a closer look. Write for a brochure or give us a call direct.

INTECOM...*the* INtelligent TExt COMmunicator.

NCCS
USER
110

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

Visit us at
DEXPO 82
Atlanta
Marriott
May 10-12
Booth
807

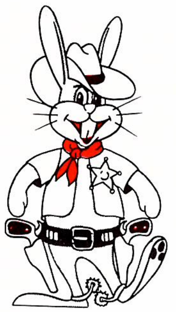
*INTECOM is currently available on DEC computers using the RSTS operating system.
RSTS is a registered trademark of Digital Equipment Corporation.
INTECOM is a trademark of Logic eXtension Resources.

CIRCLE 76 ON READER CARD

```
1110    OPEN "KB:KB.IO" FOR OUTPUT AS FILE #.CHAN.KB% &
!       &
!       Open the terminal. &

2000!   &
!               Start of MAIN &
!       &

2010    PRINT #.CHAN.KB%, "Input file <Exit> "; &
\       LINPUT #.CHAN.KB%, INPUT_FILE$ &
\       GOTO 32700 &
                IF      LEN(INPUT_FILE$) = 0% &
                OR      EDIT$(INPUT_FILE$, -1%) = "EXIT" &
\       INPUT_FILE$ = INPUT_FILE$ + ".MAC" &
                UNLESS  INSTR(1%, INPUT_FILE$, ".") &
\       FIRQB$ = SYS(CHR$(6%) + CHR$(-10%) + INPUT_FILE$) &
\       INPUT_NAME$ = RAD$(FQNAM1%) + RAD$(FQNAM2%) &
\       INPUT #.CHAN.KB%, "Generate MAC files <Yes> "; TEMP_0$ &
\       GOTO 5000 &
                IF      (ASCII(TEMP_0$) AND 95%) = .ASCII.N% &
!       &
!       Get the input file name. &
!       Fill in the input file name. &
!       Extract the file name. &
!       Ask the user if they want the mac files. &

3000!   &
!               DO THE RO FILE. &
!       &

3010    PRINT #.CHAN.KB%, "Generating RO.MAC" &
\       OPEN INPUT_FILE$ FOR INPUT AS FILE #.CHAN.IN% &
                ,ACCESS READ &
\       OPEN "RO.MAC" FOR OUTPUT AS FILE #.CHAN.RO% &
\       TEMP_0% = FNCOPY%(HT + ".RADIX" + HT + "10", .CHAN.RO%) &
\       PRINT #.CHAN.RO%, HT + ".RADIX" + HT + "10" + CR + LF &
                + HT + ".ENABL" + HT + "GBL" + CR + LF &
                + HT + ".PSECT" + HT + "$CODE,RW,I,LCL,REL,CON" + CR + LF &
                + "$CODE:" + CR + LF &
                + HT + ".PSECT" + HT + "$PDATA,RW,D,LCL,REL,CON" + CR + LF &
                + "$PDATA::" + CR + LF &
\       TEMP_0% = FNSKIP%(HT + ".PSECT" + HT + "$PDATA", .CHAN.RO%) &
\       TEMP_0% = FNCOPY%(HT + ".PSECT" + HT + "$STRNG", .CHAN.RO%) &
\       TEMP_0% = FNSKIP%(HT + ".PSECT" + HT + "$CODE", .CHAN.RO%) &
\       TEMP_0% = FNSKIP%("20$:", .CHAN.RO%) &
\       PRINT #.CHAN.RO%, "START::" &
\       TEMP_0% = FNCOPY%(HT + ".END" + HT + "$CODE", .CHAN.RO%) &
\       PRINT #.CHAN.RO%, HT + ".END" &
\       CLOSE #.CHAN.IN%, .CHAN.RO% &
!       &
!       Keep the user informed. &
!       Open the input file. &
!       Open the ro file. &
!       Do the ro file. &
!       Close the input file. &
!       Close the ro file. &

4000!   &
!               DO THE RW FILE. &
!       &

4010    PRINT #.CHAN.KB%, "Generating RW.MAC" &
\       OPEN INPUT_FILE$ FOR INPUT AS FILE #.CHAN.IN% &
                ,ACCESS READ &
\       OPEN "RW.MAC" FOR OUTPUT AS FILE #.CHAN.RW% &
\       TEMP_0% = FNCOPY%(HT + ".RADIX" + HT + "10", .CHAN.RW%) &
\       PRINT #.CHAN.RW%, HT + ".RADIX" + HT + "10" + CR + LF &
                + HT + ".ENABL" + HT + "GBL" + CR + LF &
                + HT + ".PSECT" + HT + "$CODE,RW,I,LCL,REL,CON" + CR + LF &
                + "$CODE:" + CR + LF &
                + HT + ".PSECT" + HT + "$IDATA,RW,D,LCL,REL,CON" + CR + LF &
                + "$IDATA::" + CR + LF &
                + HT + ".PSECT" + HT + "$ARRAY,RW,D,LCL,REL,CON" + CR + LF &
                + "$ARRAY:" + CR + LF &
                + HT + ".PSECT" + HT + "$TDATA,RW,D,LCL,REL,CON" + CR + LF &
                + "$TDATA:" + CR + LF &
                + HT + ".PSECT" + HT + "$STRNG,RW,D,LCL,REL,CON" + CR + LF &
                + "$STRNG::" &
\       TEMP_0% = FNSKIP%(HT + ".PSECT" + HT + "$FLAGR,RW,D,GBL,REL,CON", .CHAN.RW%) &
\       TEMP_0% = FNCOPY%(HT + ".PSECT" + HT + "$PDATA", .CHAN.RW%) &
\       TEMP_0% = FNSKIP%(HT + ".PSECT" + HT + "$STRNG", .CHAN.RW%) &
\       TEMP_0% = FNCOPY%("10$:" + HT + ".WORD" + HT + "20$", .CHAN.RW%) &
\       PRINT #.CHAN.RW%, "10$:" + HT + ".WORD" + HT + "START" &
\       TEMP_0% = FNCOPY%("20$:", .CHAN.RW%) &
\       PRINT #.CHAN.RW%, HT + ".END" + HT + "$CODE" &
\       CLOSE #.CHAN.IN%, .CHAN.RW% &
!       &
!       Keep the user informed. &
!       Open the input file. &
!       Open the rw file. &
!       Do the rw file. &
!       Close the input file. &
!       Close the rw file. &

5000!   &
!               GENERATE THE CONTROL FILES. &
!       &

5010    INPUT #.CHAN.KB%, "Generate command files <Yes> "; TEMP_0$ &
\       GOTO 5900 &
                IF      (ASCII(TEMP_0$) AND 95%) = .ASCII.N% &
\       INPUT #.CHAN.KB%, "Enter a guess at the size of the RTS <16> "; TEMP_0% &
\       TEMP_0% = 16% &
                UNLESS  TEMP_0% &
\       TEMP_1% = (((TEMP_0% + 3%) / 4%) * 4%) &
\       STACK% = (TEMP_1% - TEMP_0%) * 1024% &
\       TEMP_1% = TEMP_1% / 4% &
!       &
!       Ask if they want the command file. &
!       According to MAKSIL, the PAR and STACK parameters are defined as &
!       follows for run-time systems of various sizes: &
!       &
!       Size                                    Par &
!       ------------------------------------------------ &
!          1K -  4K                             PAR=160000:020000 &
!          5K -  8K                             PAR=140000:040000 &
!          9K - 12K                             PAR=120000:060000 &
!         13K - 16K                             PAR=100000:100000 &
!         17K - 20K                             PAR=060000:120000 &
!         21K - 24K                             PAR=040000:140000 &
!         25K - 28K                             PAR=020000:160000 &
!       &
!       Size                                    Stack &
!       ------------------------------------------ &
!       1K 5K  9K 13K 17K 21K 25K               STACK=3072 &
!       2K 6K 10K 14K 18K 22K 26K               STACK=2048 &
!       3K 7K 11K 15K 19K 23K 27K               STACK=1024 &
!       4K 8K 12K 16K 20K 24K 28K               STACK=0000 &

5020    OPEN "RO.CMD" FOR OUTPUT AS FILE #.CHAN.CF% &
\       PRINT #.CHAN.CF%, "RO/-HD,RO,RO=RO,RW.STB,LB:BP2COM/LB/-MA" + CR + LF &
                + "LB:SYSLIB/LB:RSXRTS:RSXIO:RSXAST:RSXSST:RSXDIR" + CR + LF &
                + "/" + CR + LF &
                + "STACK=" + NUM1$(STACK%) + CR + LF &
                + "PAR=RO:" + PAR_PARAM$(TEMP_1%) + CR + LF &
                + "GBLDEF=O.FLAG:0" + CR + LF &
                + "EXTSCT=.99998:0" + CR + LF &
                + "//" &
\       CLOSE #.CHAN.CF% &
!       &
```

RABBIT-4 FILE SECURITY GUNS DOWN DATA RUSTLERS ON RSTS/E SYSTEMS

Do You Know Who The Pesky Varmints Reading Your Confidential Information Are?



RABBIT-4 will help you track'm and catch'm, cause it has a proven record of performance. **RABBIT-4** will let you:

□ Log secured file accesses
□ Signal OPSER of violations
□ Roll-out the bandits
□ Freeze system activities with

6 levels of file security to keep your data safe and secure, **RABBIT-4** will also:

- Secure up to 64 data files
- Provide 32 user descriptions plus wild cards
- Restrict file access to specified programs
- Identify intrusions and intruders
- Recover automatically from system crashes

SEE OUR TALENTED RABBITS AT THE ATLANTA DECUS MEETING
and pick up a free Rabbit Executive Coloring Book Souvenir!
We'll have a hospitality suite where we'll demonstrate
Rabbit's resourcefulness in
* Performance Analysis * Job Accounting * Resource Accounting
* Data Management * Financial Planning * File Security

RAXCO INC.

RAXCO Inc., Suite 200, 6520 Powers Ferry Road, Atlanta, GA 30339 • (404) 955-2553

CIRCLE 21 ON READER CARD

CAMS
(Computer Assisted Medical System)
is now available to RSTS/E Users!

New software package handles patient information, billing, all insurance processing.

Easy to learn, simple to operate!

- **Designed for RSTS/E Timesharing**
- **Basic Plus 2**
- **RMS Files**
- **VT100 Full Screen**

NATIONWIDE INSTALLATIONS
REFERENCES ON REQUEST

CALL TODAY!
(217) 529-8700

SSP

155 West Lake Drive
Springfield, IL 62703

CIRCLE 113 ON READER CARD

```
!       Create the read-only TKB command file. &
!       Include the RSX run-time system modules needed. &
!       Setup the stack. &
!       Setup the partition. &
!       Setup the flag word for a non-kbm run-time system. &
!       Setup a dummy extend section for MAKSIL. &

5030    OPEN INPUT_NAME$ + ".CMD" FOR OUTPUT AS FILE #.CHAN.CF% &
\       PRINT #.CHAN.CF%, "$ALLOW NO ERRORS" + CR + LF &
                + "MACRO RW=RW" + CR + LF &
                + "MACRO RO=RO" + CR + LF &
                + "$ALLOW WARNING ERRORS" + CR + LF &
                + "!" + CR + LF &
                + "! Ignore the undefined symbol error message." + CR + LF &
                + "!" + CR + LF &
                + "TKB ,RW,RW=RW" + CR + LF &
                + "!" + CR + LF &
                + "! Ignore the 4 multiply defined symbol error messages." &
                + CR + LF &
                + "! If TKB ends with 'TASK HAS ILLEGAL MEMORY LIMITS,' " &
                + "you guessed too small." + CR + LF &
                + "!" + CR + LF &
                + "TKB @RO" + CR + LF &
                + "!" + CR + LF &
                + "! If MAKSIL aborts with 'Incorrect file size,' " &
                + "you guessed too small." + CR + LF &
                + "! If MAKSIL aborts with 'Partition or stack parameter " &
                + "incorrect for file size,'" + CR + LF &
                + "!  you guessed too large." + CR + LF &
                + "!" + CR + LF &
                + "$ALLOW WARNING ERRORS" + CR + LF &
                + "RUN [1,2]MAKSIL" + CR + LF &
                + "RO/RTS" + CR + LF &
                + "RO.TSK" + CR + LF &
                + "YES" + CR + LF &
                + "RO.CMD" + CR + LF &
                + "RO1.CMD" + CR + LF &
                + "!" + CR + LF &
                + "! Ignore the 4 multiply defined symbol error messages." &
                + CR + LF &
                + "!" &
        PRINT #.CHAN.CF%, "$ALLOW WARNING ERRORS" + CR + LF &
                + "TKB @RO1" + CR + LF &
                + "RUN [1,2]MAKSIL" + CR + LF &
                + "RO/RTS" + CR + LF &
                + "RO.TSK" + CR + LF &
                + "NO" + CR + LF &
                + "YES" + CR + LF &
                + "RO.STB" + CR + LF &
                + "SY0:[0,1]" + INPUT_NAME$ + ".RTS" + CR + LF &
                + "$ALLOW FATAL ERRORS" + CR + LF &
                + "UT REMOVE " + INPUT_NAME$ + CR + LF &
                + "UT ADD " + INPUT_NAME$ + CR + LF &
                + "$ALLOW WARNING ERRORS" + CR + LF &
                + "!" + CR + LF &
                + "! Ignore the 3 multiply defined symbol error messages." &
                + CR + LF &
                + "!" + CR + LF &
                + "TKB " + INPUT_NAME$ + "=RW,RO.STB" + CR + LF &
                + "$ALLOW NO ERRORS" + CR + LF &
                + "UT NAME " + INPUT_NAME$ + "=" + INPUT_NAME$ + ".TSK" &
                + CR + LF &
                + "UNSAVE RO.MAC" + CR + LF &
                + "UNSAVE RO.OBJ" + CR + LF &
                + "UNSAVE RO.TSK" + CR + LF &
                + "UNSAVE RO.STB" + CR + LF &
                + "UNSAVE RW.MAC" + CR + LF &
                + "UNSAVE RW.OBJ" + CR + LF &
                + "UNSAVE RW.STB" + CR + LF &
                + "UNSAVE RO.CMD" + CR + LF &
                + "UNSAVE RO1.CMD" &
\       CLOSE #.CHAN.CF% &
!       &
!       Create the command file. &

5900    GOTO 32700 &
!       &
!       End of the command file generator. &

14100!  &
                COPY A UNTIL A STATEMENT IS FOUND. &
!       &

14110   DEF* FNCOPY%(TEMP_0$, TEMP_0%) &
\       UNTIL   TEMP_0$ = TEMP_1$ &
\               LINPUT #.CHAN.IN%, TEMP_1$ &
\               PRINT #TEMP_0%, FNGLOBALIZE$(TEMP_1$) &
                        UNLESS  TEMP_0$ = TEMP_1$ &
```

```
\       NEXT &
\       FNEND &
!       &
!       Begin the control loop. &
!               Read a line. &
!               Write the line &
!                       unless we are done. &
!       End the control loop. &
!       End of the function. &

14200!  &
!               SKIP STATEMENTS UNTIL A STATEMENT IS FOUND. &
!       &

14210   DEF* FNSKIP%(TEMP_0$, TEMP_0%) &
\       LINPUT #.CHAN.IN%, TEMP_1$ &
                UNTIL   TEMP_0$ = TEMP_1$ &
\       PRINT #TEMP_0%, FNGLOBALIZE$(TEMP_1$) &
\       FNEND &
!       &
!       Read lines &
!               until we find our match. &
!       Write it to the output file. &
!       End of the function. &

14300!  &
!               GLOBALIZE STATEMENT LABELS. &
!       &

14310   DEF* FNGLOBALIZE$(TEMP_2$) &
\       GOTO 14320 &
                IF      ASCII(TEMP_2$) <> .ASCII.L% &
\       TEMP_1% = INSTR(1%, TEMP_2$, ":") &
\       TEMP_2$ = MID(TEMP_2$, 1%, TEMP_1%) + ":" &
                + MID(TEMP_2$, TEMP_1% + 1%, 255%) &
!       &
!       Don't fiddle with it if not a line number. &
!       Find the colon. &
!       Add another colon. &

14320   FNGLOBALIZE$ = TEMP_2$ &
\       FNEND &
!       &
!       Return the line to the user. &
!       End of the function. &

19000!  &
!               Error Handler &
!       &

19011   IF      ERR = 11% &
        THEN    PRINT #.CHAN.KB%, "?Bad MAC file format, check for " &
                        + "control characters." &
                        IF      (ERL = 14110% &
                        OR       ERL = 14210%) &
\               RESUME 32700 &
!       &
!       End of file on device. &

19300   IF      ERL = 2010% &
        OR      ERL = 2030% &
        THEN    PRINT #.CHAN.KB%, "Bad filename - " + ERT$(ERR) &
\               RESUME 2010 &
!       &
!       In case the user types in some garbage. &

19999   ONERROR GOTO 0 &
!       &
!       Give up. &

20000!  &
!               Data Statements &
!       &

20010   DATA "160000:020000", "140000:040000", "120000:060000"
20020   DATA "100000:100000", "060000:120000", "040000:140000"
20030   DATA "020000:160000" &

32700!  &
!               Completion Routines &
!       &

32710   CLOSE #.CHAN.KB%, .CHAN.IN%, .CHAN.RO%, .CHAN.RW%, .CHAN.CF% &
!       &
!       Close all channels (the fast way). &

32767   END
```

datasouth announces...
THE TOTAL PRINTER PACKAGE!

With so many matrix printers available today, it's tough to find the right one for your application. Some offer the speed you need, others the communications flexibility and still others the forms handling capability. But no printer offers **all** the features you need... until now.

datasouth
computer corporation

The DS 180 provides a total package of performance features for any application where quality impact printing is required. Not a "hobby-grade" printer, the DS 180 is a real work-horse designed to handle your most demanding printer requirements.

High Speed Printing—Bidirectional printing at 180 cps offers throughput of over 200 1pm on average text. A 9-wire printhead generates a 9x7 font with true lower case descenders and underlining.

Non-volatile Format Retention—A unique keypad featuring a non-volatile memory makes programming the DS 180 quick and easy. Top of form, tabs, perforation skipover, communications parameters and many other features may be entered and stored from the keypad. The DS 180 even remembers the line where you stopped printing—eliminating the need to reset the top of form at power-on.

Communications Versatility—Interfaces include RS232, current loop and 8-bit parallel. Baud rates from 110-9600 may be selected. A 1K buffer and X-on, X-off handshaking ensure optimum throughput.

Forms Handling Flexibility—Forms ranging from 3"-15" may be fed from the front or bottom, and an adjustable printhead provides crisp and clear copy on forms with as many as 6-parts.

For more information on how the DS 180's low-cost total printer package can fill your application, contact us at Datasouth. The DS 180 is available for 30-day delivery from our sales/service distributors throughout the U.S.

4216 Stuart Andrew Blvd. • Charlotte, North Carolina 28210 • 704/523-8500

CIRCLE 73 ON READER CARD

VAX™ SPOKEN HERE!*

There's only one accounting, manufacturing software company that speaks the language of VAX™...

Interactive Management Systems.

IMS has been developing business application software for the VAX family of computers since its introduction by Digital Equipment Corporation. As a result, we know from long experience what *you,* the VAX-user, need in order to get your accounting and manufacturing functions on-line.

Our truly native mode software is written entirely in VAX-11 Basic, utilizing VAX-11RMS. And that makes it compatible with DEC's Datatrieve query language. You'll also find IMS Software easy to integrate, exceptionally versatile. For more information, see our new IMS Software Portfolio, featuring the most extensive line of Business Applications Software for VAX and RSTS/E. Clip and mail the coupon below... or call us today at (617) 489-3550.

ims INTERACTIVE
MANAGEMENT SYSTEMS

The software company that speaks your language!

While in Atlanta at DECUS,
come talk with us at DEXPO-82, Booth #816!

*We also speak RSTS/E.
DEC, VAX, Datatrieve are trademarks of Digital Equipment Corporation.

CIRCLE 122 ON READER CARD

VAX Spoken Here!

For free IMS Software Portfolio, clip and mail to: Interactive Management Systems, 375 Concord Avenue, Belmont, MA 02178-3094

Name ____________ Title ____________

Company ____________

Address ____________

City ____________ State ____________

Zip ____________ Phone ____________

THE DEC/RSTS DECISION SUPPORT SOLUTION

When it comes to comprehensive financial, organizational or strategic planning, there is only one choice for the RSTS user:

FCS-EPS

FCS-EPS is the one system that brings the power of the computer directly into the hands of the planning professional.

More than just a modeling system, FCS-EPS is a sophisticated, open-ended system easily applied to virtually any planning task. Over 60 built-in planning-oriented functions allow you to be immediately productive. "What if" analysis, goal-seeking, a customized report writer, hierarchical consolidation, and a built-in financially oriented language make the utility of FCS-EPS virtually unlimited, all without the need to know any cryptic computer language.

The system is also available on the Decsystem 10 and 20, VAX and over 40 other hardware/operating system combinations.

700 users of FCS-EPS can't be wrong. Find out more about truly user-oriented financial planning systems. Contact EPS today. San Jose, CA 800/538-7578 or 408/292-6212; Toronto 416/279-8711; London (01) 579-6931.

Clip and mail to EPS, Inc., 1788 Technology Drive, San Jose, CA 95110

_____Yes. Send me information on FCS-EPS

_____Yes. Send me "Selecting and Evaluating Financial Modeling Systems."

Name_____________________Title_____________

Company__________Address____________________

City______________State__________Zip________

Phone____________Computer now in use________

CIRCLE 103 ON READER CARD

```
>BP2

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


Basic2

OLD LB1:SYSTAT.BAS

Basic2

COM /MAC

Basic2

EXIT
>;
>;      At this point you must edit the MAC file to
>;      eliminate the control characters from the
>;      comments in the file.
>;
>RUN BP2RTS

Bp2rts  V7.0-01 Software Techniques
Split MAC into RO and RW.

Input file <Exit> SYSTAT.MAC
Generate MAC files <Yes> YES
Generating RO.MAC
Generating RW.MAC
Generate command files <Yes> YES
Enter a guess at the size of the RTS <16> 17
>@ SYSTAT
```

```
>MACRO RW=RW
ERRORS DETECTED:  0
>MACRO RO=RO
ERRORS DETECTED:  0
>!
! Ignore the undefined symbol error message.
!
TKB ,RW,RW=RW
%TKB -- *DIAG*-154 UNDEFINED SYMBOLS SEGMENT RW


%Task exit status: ERROR
>!
! Ignore the 4 multiply defined symbol error messages.
! If TKB ends with 'TASK HAS ILLEGAL MEMORY LIMITS,' you guessed too small.
!
TKB @RO
%TKB -- *DIAG*-MODULE RSXIO  MULTIPLY DEFINES SYMBOL ..CRLF


%TKB -- *DIAG*-MODULE RSXIO  MULTIPLY DEFINES SYMBOL ..PTXT


%TKB -- *DIAG*-MODULE RSXIO  MULTIPLY DEFINES SYMBOL ..RSTT


%TKB -- *DIAG*-MODULE RSXAST MULTIPLY DEFINES SYMBOL ..PMD


%Task exit status: ERROR
>!
! If MAKSIL aborts with 'Incorrect file size,' you guessed too small.
! If MAKSIL aborts with 'Partition or stack parameter incorrect for file size,'
!  you guessed too large.
!
RUN [1,2]MAKSIL
MAKSIL  V7.0-07+/MU PATCH       RSTS V7.0-07 Softec 11/23
Resident Library name? RO/RTS
Task-built Run-Time System input file <RO.TSK>? RO.TSK
The run-time system is not aligned
Edit mode (Yes/No) <Yes>? YES
Task-builder command input file <RO.CMD>? RO.CMD
The task-builder commands have been changed as follows
  PAR=RO:060000:120000              PAR=RO:060000:120000
  STACK=3072                        STACK=3072
  EXTSCT=.99998:0                   EXTSCT=.99998:003674

RO will load in a 20 K-word partition using 17 K-words physical memory.
003674 (octal) bytes may be used for expansion.

Corrected command file name <RO.CMD>? RO1.CMD
Please task build again using RO1.CMD
>!
! Ignore the 4 multiply defined symbol error messages.
!
TKB @RO1
%TKB -- *DIAG*-MODULE RSXIO  MULTIPLY DEFINES SYMBOL ..CRLF


%TKB -- *DIAG*-MODULE RSXIO  MULTIPLY DEFINES SYMBOL ..PTXT


%TKB -- *DIAG*-MODULE RSXIO  MULTIPLY DEFINES SYMBOL ..RSTT


%TKB -- *DIAG*-MODULE RSXAST MULTIPLY DEFINES SYMBOL ..PMD


%Task exit status: ERROR
>RUN [1,2]MAKSIL
MAKSIL  V7.0-07+/MU PATCH       RSTS V7.0-07 Softec 11/23
Resident Library name? RO/RTS
Task-built Run-Time System input file <RO.TSK>? RO.TSK
%Run-time system maximum job size (28) exceeds calculated maximum of 12
The run-time system is correctly aligned
Edit mode (Yes/No) <Yes>? NO
Include symbol table (Yes/No) <Yes>? YES
Symbol table input file <RO.STB>? RO.STB
Run-Time System output file <SY:[0,1]RO.RTS>? SY0:[0,1]SYSTAT.RTS
RO built in 17 K-words, 669 symbols in the directory
RO.TSK renamed to RO.TSK<40>
Utility ADD suppressed
>UT REMOVE SYSTAT
?Can't find file or account
>UT ADD SYSTAT
>!
! Ignore the 3 multiply defined symbol error messages.
!
TKB SYSTAT=RW,RO.STB
%TKB -- *DIAG*-MODULE RO     MULTIPLY DEFINES SYMBOL $IDATA


%TKB -- *DIAG*-MODULE RO     MULTIPLY DEFINES SYMBOL $OTSVA


%TKB -- *DIAG*-MODULE RO     MULTIPLY DEFINES SYMBOL $STRNG


%Task exit status: ERROR
>UT NAME SYSTAT=SYSTAT.TSK
>UNSAVE RO.MAC
>UNSAVE RO.OBJ
>UNSAVE RO.TSK
>UNSAVE RO.STB
>UNSAVE RW.MAC
>UNSAVE RW.OBJ
>UNSAVE RW.STB
>UNSAVE RO.CMD
>UNSAVE RO1.CMD

>RUN SYSTAT

SYSTAT  V7.0-07 RSTS V7.0-07 Softec 11/23
Output Status to? /S

RSTS V7.0-07 Softec 11/23 status at 04-Mar-82, 07:02 PM Up: 5:25:58

Job    Who     Where    What     Size     State     Run-Time  Pri/RB   RTS
 1    [OPR]    Det      ERRCPY   10/31K   SR C12        7.5     0/6   ...RSX
 2    [OPR]    Det      OPSRUN   21/31K   SL C14       24.7   -16/6   ...RSX
 3    [OPR]    Det      QUMRUN   24/31K   SL C08       20.5   -16/6   ...RSX
 4    [OPR]    Det      SPLRUN    4/31K   SL C01        0.1   -16/6   SPLRUN
 5    [OPR]    Det      BATRUN    4/31K   SL C13        0.2   -16/6   BATRUN
 7    [SELF]   KB9      ATPK      3/31K   SL         1:48.8    -8/6   ATPK
 8    [SELF]   P0J7     SYSTAT    2/31K   RN Lck        7.1    -8/6   SYSTAT
 9    [SELF]   P1J5     SLEEPR    2/31K   SL            0.1   -16/6   ...RSX
```

LOOK at the "tear-out" cards in this issue.
There's subscription cards for you or a friend.
There's a FREE gift for you. Bring in new subscribers and collect rewards. See "BOUNTY HUNTERS" card.
There's a READERS INQUIRY card for your convenience.

EXTRACT

By Stephen Munyan, 135 Brattle St., Holden, MA 01520

EXTRACT is a program which was designed to allow programmers to copy lines of one program into another quickly without having to worry about going through BASIC-Plus immediate mode statements to extract the lines. This program has been in use on our system for over 1 year, and is quite widely used.

As an example of how this program can be used, we will assume that the programmer has written a program which contains several routines which need to be used in a new program under development. For convince we will call them ROUTINE.BAS and APPLIC.BAS. In this case we want to copy lines 10, 50, 700-750, and line 1000 from ROUTINE.BAS and place them into a new file called APPLIC.BAS. To accomplish this we would issue the following procedure:

```
RUN $EXTRACT
EXTRACT — Program Line Extraction Program — V7.0-01
Output=Input [/A[ppend]]
#APPLIC.BAS=ROUTINE.BAS
```

Enter the line numbers to be extracted from the input file separated by commas. A dash may appear between entries to allow ranges of lines to be extracted. Once all lines to be extracted have been entered press CTRL/Z

```
* 10,50
* 700-750
* 1000
* .Z
Extraction Complete
```

In the example above, several line number combinations were entered on the same line. As many entries as desired can be entered on the same line as long as they are separated by commas. For example, if we wanted to enter all of the lines on the same line we could have entered: 10, 50, 700-750, 1000 all on the same line.

If the lines being extracted are to be placed at the end of an existing program, the /APPEND switch can be used. If this switch is used, the program is assumed to be lacking an END statement since when the output file is OLD'ed, it will ignore any statements that were appended to the file after the END statement.

As an optional patch, line 1015 can be updated to use the RECORDSIZE option to allow BASIC-Plus to use larger recordsizes on the input file. This will speed the extraction since the program will spend less time waiting for I/O to be processed. Depending on the amount of space allocated to each user, the size of the Record can vary from 4096 to 16384 bytes.

```
                    1 !!!!!   EXTRACT          Version 7.0     Edit 1A
        !!!!!                                                         &
        !!!!!   Written by Stephen J. Munyan    (C) 1981              &
        !!!!!                                                         &
        !!!!!   This software is solely for non-commercial use and may be only &
        !!!!!   used and/or copied with the inclusion of this notice.  &
        !!!!!   No title to or ownership of this software is hereby transfered. &
        !!!!!                                                         &
        !!!!!   The information in this software is subject to change without &
        !!!!!   notice and should not be construed as a commitment by  &
        !!!!!   the author.                                           &
        !!!!!                                                         &
        !!!!!   EXTRACT --  Program to extract line numbers from a BASIC+ &
        !!!!!               program given the upper and lower bounds  &
        !!!!!                                                         &
        !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! &

110     EXTEND                                                        &

1000    DIM LIN%(255%, 1%), NUM.CHK%(5%)                              &
                                                                      &
                        ! Setup the array to hold the line number pairs that &
                        ! will be specified by the user.              &
                        !                                             &
                        !       LIN%(255%, #%)                        &
                        !                  0%) = Lower bound specified &
                        !                  1%) = Upper bound specified &

1005    ON ERROR GOTO 10000 \                                         &
        PRINT \                                                       &
        PRINT "EXTRACT - Program Line Extraction Program - V7.0-01" \ &
        PRINT \                                                       &
        LIN.PTR%=0%                                                   &
                                                                      &
                        ! Display the Program Banner Line             &

1010    OPEN "KB:" AS FILE 10% \                                      &
        PRINT "Output = Input [/A[PPEND]]" \                          &
        PRINT \                                                       &
        PRINT "# "; \                                                 &
        INPUT LINE #10%, RESPONSE$ \                                  &
        RESPONSE$=CVT$$(RESPONSE$, 103%) \                            &
        APPEND.MODE%=0% \                                             &
        EQUAL%=INSTR(1%, RESPONSE$, "=") \                            &
        GOTO 1090                                                     &
             UNLESS EQUAL% \                                          &
        SLASH%=INSTR(1%, RESPONSE$, "/") \                            &
        IF SLASH%                                                     &
            THEN IF SLASH% < EQUAL%                                   &
                        THEN 1090                                     &
                        ELSE IF MID(RESPONSE$, SLASH%+1%, 1%) <> "A"  &
                                  THEN 1090                           &
                                  ELSE APPEND.MODE%=-1% \             &
                                       RESPONSE$=LEFT(RESPONSE$, SLASH%-1%) &
                                                                      &
                        ! Scan the line looking for syntax errors     &

1015    OUTPUT.FILE$=LEFT(RESPONSE$, EQUAL%-1%) \                     &
        INPUT.FILE$=RIGHT(RESPONSE$, EQUAL%+1%) \                     &
        INPUT.FILE%=1% \                                              &
        OUTPUT.FILE%=2% \                                             &
        OPEN INPUT.FILE$ FOR INPUT AS FILE INPUT.FILE%, MODE 8192%    &
                                                                      &
                        ! Open the input file in read only mode       &

1020    IF APPEND.MODE%                                               &
             THEN OPEN OUTPUT.FILE$ AS FILE OUTPUT.FILE%, MODE 2%     &
             ELSE OPEN OUTPUT.FILE$ FOR OUTPUT AS FILE OUTPUT.FILE%   &
                                                                      &
                        ! Open the output file in the mode specified such that &
                        ! we will either create a new file, or append to an &
                        ! existing file                               &

1040    PRINT \                                                       &
        PRINT "Enter the line numbers to be extracted from the input file" \ &
        PRINT "separated by commas.  A dash may appear between entries to" \ &
        PRINT "allow ranges of lines to be extracted.  Once all lines" \ &
        PRINT "to be extracted have been entered press CTRL/Z" \      &
        PRINT                                                         &
                                                                      &
                        ! GIVE THE HELP MESSAGE                       &

1050    PRINT "* "; \                                                 &
        INPUT LINE #10%, RESPONSE$ \                                  &
        RESPONSE$=CVT$$(RESPONSE$, 103%) \                            &
        WHILE LEN(RESPONSE$) \                                        &
             COMMA%=INSTR(1%, RESPONSE$, ",") \                       &
             COMMA%=LEN(RESPONSE$)+1%                                 &
                  UNLESS COMMA% \                                     &
             WORK$=LEFT(RESPONSE$, COMMA%-1%) \                       &
             RESPONSE$=RIGHT(RESPONSE$, COMMA%+1%) \                  &
             DASH%=INSTR(1%, WORK$, "-") \                            &
             IF DASH%=0%                                              &
                  THEN DASH%=LEN(WORK$)+1% \                          &
                       WORK$=WORK$+"-"+WORK$                          &
                                                                      &
                        ! Scan the command line looking for either a line &
                        ! number or a dash to indicate which type of command &
                        ! is being processed.  ie: x or x-y           &
                        !                                             &
                        ! Note:  x is converted into x-x              &

1055         LOW%=VAL(LEFT(WORK$, DASH%-1%)) \                        &
             HIGH%=VAL(RIGHT(WORK$, DASH%+1%)) \                      &
             IF (LOW% < 1%) OR (HIGH% > 32767%) OR (LOW% > HIGH%)     &
                  THEN 1080                                           &
                  ELSE LIN.PTR%=LIN.PTR%+1% \                         &
                       IF LIN.PTR% > 255%                             &
                            THEN PRINT "?Only 255 entries can be specified "; &
                                       "-- Extraction will now proceed" \ &
                                 PRINT \                              &
                                 PRINT "The following entries will not be "; &
                                       "processed:" \                 &
                                 PRINT \                              &
                                 PRINT WORK$+","+RESPONSE$ \          &
                                 PRINT \                              &
                                 PRINT "Re-run the program with /APPEND to "; &
                                       "include these lines in the file" \ &
                                 GOTO 2000                            &
                                                                      &
                        ! Try to find a place in the search array for this &
                        ! request.  If no slot is found give a warning message &
                        ! then start processing.                      &

1060         LIN%(LIN.PTR%, 0%)=LOW% \                                &
             LIN%(LIN.PTR%, 1%)=HIGH%                                 &
                                                                      &
                        ! Otherwise load up the current low-high values &

1070    NEXT \                                                        &
        GOTO 1050                                                     &
                                                                      &
                        ! End of the scanning loop
1080    PRINT \                                                       &
        PRINT "?The following entry is invalid '"+WORK$+"' -- Proceeding" \ &
        PRINT \                                                       &
        GOTO 1070                                                     &
                                                                      &
                        ! All errors including invalid line numbers will &
                        ! trap here                                   &

1090    PRINT \                                                       &
        PRINT "?Command syntax error -- Please re-enter" \            &
        PRINT \                                                       &
        GOTO 1010                                                     &

2000    VALID.LINE%=0%                                                &
                                                                      &
                        ! Set the transfer flag to zero so we do not transfer &
                        ! any information that is not part of a valid BASIC+ &
                        ! line.                                       &

2005    INPUT LINE #INPUT.FILE%, TEXT$ \                              &
        GOTO 2050                                                     &
             IF CONTINUATION%                                         &
                                                                      &
                        ! Get a line of text from the file.  If we are currently &
                        ! in a continuation line, then this text should be sent &
                        ! to the output processor.  If the line is not valid, &
                        ! then it will not be printed by the processor to the &
                        ! output file.                                &

2010    CHANGE LEFT(CVT$$(TEXT$, 103%), 5%) TO NUM.CHK% \             &
        NUMBER%=0% \                                                  &
        FOR I%=1% TO NUM.CHK%(0%) \                                   &
             GOTO 2015                                                &
                  UNLESS (NUM.CHK%(I%) > 47%) AND (NUM.CHK%(I%) < 58%) \ &
             NUMBER%=NUMBER%*10% + (NUM.CHK%(I%) - 48%) \             &
        NEXT I%                                                       &
                                                                      &
                        ! Scan the first 5 characters of the line looking for &
                        ! for a line number                           &

2015    GOTO 2005                                                     &
             UNLESS NUMBER%                                           &
                                                                      &
                        ! Ignore blank lines that are not within the range &
                        ! of our scan                                 &

2020    GOTO 2025                                                     &
             IF (NUMBER% >= LIN%(I%, 0%)) AND (NUMBER% <= LIN%(I%, 1%)) &
                  FOR I%=1% TO LIN.PTR% \                             &
        VALID.LINE%=0% \                                              &
        GOTO 2050                                                     &
                                                                      &
                        ! Check to see if the line that we are looking at &
                        ! is within a valid range.  If the number is not &
                        ! in that range set VALID.LINE% to zero so that &
                        ! the line will not print, then get the next line. &

2025    VALID.LINE%=-1%                                               &
                                                                      &
                        ! Set the valid line flag to -1 to indicate that the &
                        ! routine below is to print the current line. &

2050    PRINT #OUTPUT.FILE%, TEXT$;                                   &
             IF VALID.LINE% \                                         &
        CONTINUATION%=0% \                                            &
        IF MID(TEXT$, LEN(TEXT$)-2%, 3%)=CHR$(10%)+CHR$(13%)+CHR$(0%) &
           OR MID(TEXT$, LEN(TEXT$)-1%, 2%)=CHR$(10%)+CHR$(13%)       &
             THEN CONTINUATION%=-1% \                                 &
                  GOTO 2005                                           &
                                                                      &
                        ! Print the line if it is valid.  if the line &
                        ! is invalid, skip the print but check to see if the &
                        ! line is a continauation.                    &
                        !                                             &
                        ! Possible continautions are:                 &
                        !                                             &
                        !         LF CR NULL                          &
                        !         LF CR                               &
                        !       & CR LF                               &

2055    TEMP$=CVT$$(TEXT$, 2%) \                                      &
        IF RIGHT(TEMP$, LEN(TEMP$)-2%)="&"+CHR$(13%)+CHR$(10%)        &
             THEN CONTINUATION%=-1% \                                 &
                  GOTO 2005                                           &
                                                                      &
                        ! Take care of the special case of extend mode &
                        ! continauation which may have the following form: &
                        !                                             &
                        ! PGM LINE [NOTE 1] & CR LF                   &
                        !                                             &
                        ! Note 1:                                     &
                        !                                             &
                        ! There may be any Number of spaces and tabs  &
                        !         in this position.                   &

2060    CONTINUATION%=0% \                                            &
        GOTO 2005                                                     &
                                                                      &
                        ! If we reach this point then there is no continuation &
                        ! line so clear the flag so that the first routine &
                        ! will start scanning for a new line to process &

10000   E%=ERR \                                                      &
        E$=CVT$$(RIGHT(SYS(CHR$(6%)+CHR$(9%)+CHR$(E%)), 3%), 4%) \    &
        IF ERR=11% AND ERL=1050%                                      &
             THEN RESUME 2000                                         &
             ELSE IF ERL=1055%                                        &
                       THEN RESUME 1080                               &
                                                                      &
                        ! Take care of ^Z to line entry and also numeric &
                        ! translation errors that occur when we are converting &
                        ! the user's input into machine readable form. &

10005   IF ERR=11% AND ERL=2005%                                      &
             THEN PRINT \                                             &
                  PRINT "Extraction Complete" \                       &
                  RESUME 32767                                        &
                                                                      &
                        ! Take care of the end of file condition      &

10010   IF ERL=1015%                                                  &
             THEN PRINT "?Unable to access '"+INPUT.FILE$+"':  "+E$ \ &
                  RESUME 1010                                         &
                                                                      &
                        ! If we are unable to access the input file give &
                        ! an error on the user's terminal.            &

10015   IF ERL=1020%                                                  &
             THEN PRINT "?Unable to access '"+OUTPUT.FILE$+"':  "+E$ \ &
                  RESUME 1010                                         &
                                                                      &
                        ! If we are unable to access the output file give &
                        ! and error on the user's terminal.           &

10020   IF ERR=11% AND ERL=1010%                                      &
             THEN RESUME 32767                                        &
                                                                      &
                        ! Take care of ^Z exit from the filename routine &

10999   PRINT "?Unexpected error:  ";E$;" at line";ERL \              &
        RESUME 32767                                                  &
                                                                      &
                        ! Take care of any unexpected error that may occur &

32767   END
```

FAST-11

SYSTEM DEVELOPMENT TOOL FOR RSTS/E BASIC -PLUS

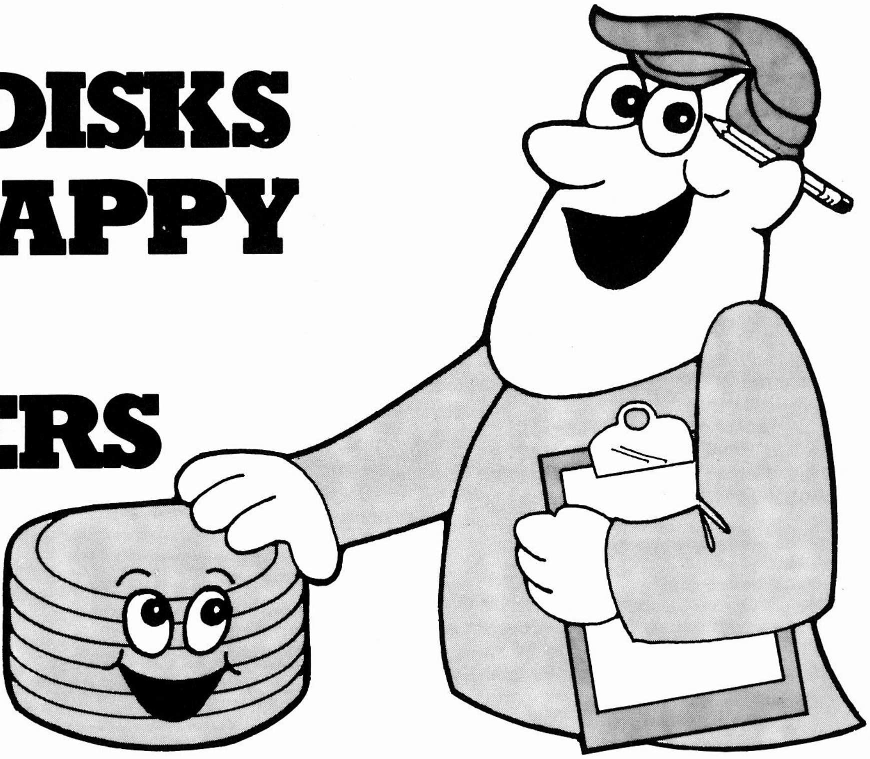
- Automated Definition of Record Layouts
- ISAM, Sequential, Random, and Tape Files
- Version 7 RSTS/E Large File support
- Program generated, RNO compatible Document Facility
- Supports numerous Data Types
- DMS 500/FAM Compatible

CASHER ASSOCIATES INC.
1371 Beacon Street • Brookline, MA 02146

(617) 232-9111
(212) 757-2868

CIRCLE 78 ON READER CARD

HAPPY DISKS MAKE HAPPY SYSTEM MANAGERS



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

- Supports mixed disk types.
- Untangles directory links.
- Creates contiguous UFD's to proper size.
- Creates accounts in sorted order.
- Locates files adjacent to user's UFD.
- Optimizes clustersizes.
- Saves accounting data.
- Is easy to use and fully documented.
- Provides logging and statistics.

LAST CHANCE TO BUY DSKBLD AT THE $500 INTRODUCTORY PRICE

□ **Send me more information on DSKBLD**

Name ____________________________
Company ____________________________
Address ____________________________
____
____
Telephone ____________________________

Here's What DSKBLD Users Say:

"We have about 130 terminals in the field. The day after I used DSKBLD they called to find out what I had done to fix the response problems!...I used to have a lot of "crazy" weekend hours—now its a simple production job I can trust to my operators"

—New York user

"I would recommend it to anyone with a disk bound system. DSKBLD lets me rebuild a 300 MB disk in an hour and improves system throughput."

—Oklahoma user

"Our users noticed the results after the first use...FIP needed dropped from over 50% to under 20% immediately...After seeing the results, we would pay twice the price for it."

—Minnesota user

"Cache hits went from 45% to 81%...Overall throughput increased almost 100%...cut $15/hour overtime by 75%...have recommended it to others."

—Massachusetts user

"Can rebuild all my disks in one Sunday instead of spending three Sundays each month... definitely met my objectives... am ordering a license for my other CPU."

—Washington user

"I like the safeguards built into DSKBLD—especially for the unsophisticated user...very easy to install."

—Colorado user

"I've already recommended it to two other users."

—Colorado user

MANUS
SERVICES CORPORATION
AUTHORIZED
digital
COMPUTER DISTRIBUTOR

1700 Westlake Avenue North
Seattle, Washington 98109
(206) 285-3260

CIRCLE 39 ON READER CARD

RPGED.TEC

By Austin Kinsella, Regional Technical College, Carlow, Ireland

The Regional Technical College in Carlow is one of a number of similar 3rd level institutions in Ireland, providing mainly 2 and 3 years courses in technical subjects. The College has a PDP 11/34 RSTS/E system, on which the bulk of the time is consumed by students on our 2 year data processing course. During their second year, these students spend some time learning RPG. As we cannot afford the overhead of multiple copies of RPGESP, and as the students are already familiar with TECO for editing Cobol and Basic-Plus sources, it was decided to provide them with a simple RPG forms editor in TECO. The listing of this editor is attached. After squeezing, it adds less than 1K to each user's buffer, so that multiple copies can be run without causing swapping.

When running, the editor displays a row of column numbers and the current form mask, with legal fields denoted by ↑ or *, (or C for command) and unused columns by spaces. Cursor movement will not leave the cursor in an illegal column. Facilities are provided for left and right cursor movement, up and down lines, line renumbering, and new line creation with automatic line numbering. To minimize screen updating, only 6 text lines over the mask are displayed. The editor is loaded by EIRPGED, and is run by MR. Exit back to TECO is by ↑Z, and MR can be re-issued after any intervening TECO commands, for example a search to move to a new position in the file. We have the editor installed with protection <104> in one of our library accounts, but it can go anywhere.

RPGED is not foolproof, and there remains considerable scope for development. The primary design goal was to provide easy RPG forms editing with a low memory overhead, and we feel this has been achieved. Because most of our terminals are VT52s, only VT52 escape sequences are used in the editor, but on a VT100 the editor should run faster in ANSI mode by scrolling the text window up and down over a fixed mask. Other changes that might be desirable would be to make RPGED executable by CCL, to accept numeric arguments on the movement commands, and to retain a column position on macro entry or line change. We have a second version of the editor called SRTED which has the same functions but displays Sort rather than RPG form types.

In conclusion I should point out that I have been aware of the need for this editor for over a year, but it took the arrival of the back issues of RSTS PRO, with the articles on TECO, to spur me to write it.

Summary of RPGED commands:

↑B:	Back cursor to last legal column, or start of previous line if at start.
↑U:	Up to start of previous line, change mask if necessary.
↑D:	Down to start of next line, change mask if necessary.
<sp>:	Cursor Right to next legal column or next line if at end.
<cr>:	Make new line of 74 cols, insert mask type, insert number if numbering.

10 WAYS TO PRINT NOW, PAY LATER.

Call 800-243-9054* for details.

1. **CHARGE-A-PRINTER.** We'll install the line printer you need today, and defer payment until next quarter.
2. **MORE LPM FOR LESS $.** The Printer Store sells the identical printers that mini vendors offer, at up to 50% less.
3. **UPGRADE YOUR SYSTEM, NOT YOUR BUDGET.** Double your system throughput *now* with a new, high-speed line printer or RLPS.
4. **ORDER NOW, GET 3 MONTHS *FREE* MAINTENANCE.** Save on the cost of ownership for a limited time only.
5. **CONVERT-A-LEASE.** Lease the line printer you need today, with the option to purchase anytime.
6. **GUARANTEED TRADE-IN.** Trade in your old line printer for an additional 10-20% off our already low prices.
7. **TWO LINE PRINTERS FOR THE PRICE OF ONE.** Our PXS Switch lets you use your old printer as a back-up (free for a limited time only when you purchase a new printer).
8. **SHORT-TERM RENTALS FOR HIGH-SPEED LINE PRINTERS.** The most economical way to get increased throughput for peak periods.
9. **WE'LL REFURBISH YOUR LINE PRINTER,** and provide a "loaner" while it's in for repairs,or sell you a refurbished line printer at 20-30% of the cost of new.
10. **LET MA BELL PAY FOR YOUR REMOTE LINE PRINTER.** Our Remote Line Printer System pays for itself in line use savings.

The Printer Store

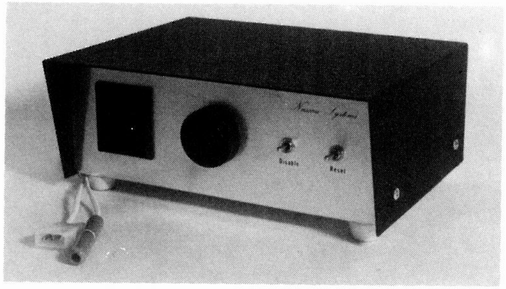
Digital Associates Corporation
1039 E. Main Street, Stamford, CT 06902
*In CT call (203) 327-9210.

CIRCLE 90 ON READER CARD

Hardware Protection
For DEC Equipment Users

C-XX Overtemperature Protection System

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



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

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

CIRCLE 27 ON READER CARD

COMPUTER CONSTIPATION?

System not putting out what it should . . ?

RSTS PERFORMANCE SPECIALISTS
Budget-Minded Consulting

ASSOCIATED SYSTEMS TECHNIQUES, INC.
10528 Lower Azusa Road
Suite 105
El Monte, California 91731

CIRCLE 124 ON READER CARD

"Whether it's systems or software . . . the answer is in the Technique."

↑F<arg>: change to mask type arg.
↑A: Toggle off/on auto-line numbering.
↑R: Force line renumbering (if auto-numbering, out-of-sequence inserts cause renumbering after ↑Z).
↑J: Jump to start next field.
↑H: Half jump to col 44 if I form or col 32 if 0 form.
↑Z: Return to TECO.
<del>: Replace current cursor character with space.
<other>: Replace current cursor character with character typed.

```
! RPGED.TEC !
! TECO-based editor for RPG forms !
! AK RTC Carlow Jan 82 !

! Registers Used (contents clobbered) !

! A$ Auto line number macro !
! A% Auto line number flag !
! C$ C Form Mask !
! D$ Display mask macro !
! E$ E Form Mask !
! F$ F Form Mask !
! H$ H Form Mask !
! I$ I Form Mask !
! J$ List of mask types !
! L$ L Form Mask !
! M$ Current mask !
! N$ Renumbering macro !
! N% Renumbering needed flag !
! O$ O Form Mask !
! R$ Main macro !
! S$ Scratch !
! S% Scratch !
! T% Terminal input !
! U$ Update screen macro !

@^UR? ! Load Keyboard Monitor !

    ! Initialisation !

  155^T 72^T 155^T 74^T                       ! Home & clear !
  155^T 89^T 51^T 32^T                        ! Line 20 col 1 !
  7< @^A%1234567890% > @^A%1234%              ! Display col nos !
  1UA 0L                                      ! Set ALN flag, start of line !
  0,8ET MU                                    ! Set no echo , update screen !
  ^Q"G 5AUT MD '                              ! If in a line, do mask display !

    ! Main input char & process loop !

  <                                           ! Keep returning here till ^Z !
   ^TUT                                        ! Get char and save it !
   QT-32"G D                                   ! If not control delete char !
            QT-127"N @QTI%%  QT^T                ! If not delete insert and echo !
                    | @I% % 32^T '                ! Else insert space and echo !
           -(0^Q)QM-32"E 32UT                    ! If cursor on mask space, do SP !
                          | 0^Q+73"L 4UT          ! Else at EOL do ^D !
                          | F< '''                ! Else flowback for more !
                                                 ! We fall into LBO if SP or ^D !
   !LBO!                                       ! We jump  here with simulated input !
   QT-32"E <                                   ! SP: loop start !
             C 155^T 67^T                         ! Advance cursor !
             -(0^Q)QM-32"N 0; '                   ! Exit loop if column legal !
            >                                    ! Else loop again !
            0^Q+73"L 4UT @O!LBO! '  F< '         ! If at EOL do ^U else flowback !
   QT-1"E QA"E 1UA | 0UA ' F< '                ! ^A: set/reset ALN flag !
   QT-6"E ^TUT @QTI%% R                        ! ^F: get argument, put in buffer!
           ::@S%^EGJ%"S -D MD | D 7^T'  F<'      ! If valid do mask else buzz !
   QT-13"E ^T^[                                ! CR: eat LF !
             L 37<@I%  %> @13I%%@10I%%           ! Make new line !
             -L QA"G MA | 5C '                   ! Into line, do ALN if on else cursor to col 6!
             D @5QMI%% R                         ! Insert form type !
             QA"E 0L'                            ! If not  ALN, back to line no field !
             MU F< '                             ! Update, loop !
   QT-26"E 0; '                                ! ^Z: jump out of main loop !
   QT-10"E <                                   ! ^J: loop start !
             C 155^T 67^T                         ! Advance cursor !
             -(0^Q)QM-((-(0^Q)-1)QM)"E F< '       ! If 2 mask chars same try again !
             -(0^Q)QM -32"N 0; '                  ! Else exit loop if not space !
            >                                    ! End loop !
            0^Q+73"L 4UT @O!LBO! '  F< '         ! Do ^D if EOL else flowback !
   QT-2"E <                                    ! ^B: start loop !
            155^T 68^T R                          ! Back cursor !
            -(0^Q)QM-32"N 0;'                     ! If not space, exit loop !
           >                                     ! Else try again !
           0A-10"E 21UT @O!LBO! ' F< '           ! If at EOL do ^U else flowback !
   QT-11"E 0L K MU F< '                        ! ^K: kill line, update !
   QT-4"E L                                    ! ^D: down line !
           Z-."E -L                              ! If no line, back up !
              | 5A-(5QM)"N 5AUT MD ' '            ! Else do mask if wrong !
           MU F< '                               ! Update screen, flowback !
   QT-21"E -L MU 5A-(5QM)"N 5AUT MZ ' F< '     ! ^U: up line, update, change mask if needed !
   QT-18"E MN MU F< '                          ! ^R: do renumbering !
   QT-8"E                                      ! ^H: half jump !
           5QM-73"E 0^Q+43< C 155^T 67^T> '        ! If type I advance to col 44 (won't if already > ) !
           5QM-79"E 0^Q+31<C 155^T 67^T>'          ! If type O advance to 32 !
             F< '                                ! Flowback !
   3<7^T>  F<                                   ! Anything else, buzz !
  >                                           ! End main loop !

    ! Tidy up before exiting to TECO !

  QN"G 155^T 89^T 54^T 32^T                   ! If renumbering needed !
       @^A%Renumbering...% MN '                 ! Say so and do it !
  8,0ET 155^T 72^T 155^T 74^T                 ! Reset ET flags clear screen !
?
@^UA? ! Load Auto-Line-Number Macro !

  -L \+10 US                                  ! Look at last line number, add 10 !
  2L ^Q"G \-QS"G F' | 1UN ''                  ! Set renumber flag if insert too high !
  -L  5D  QS\ 0^Q US 0L 5+QS<@I%0%>           ! Into line, delete spaces, insert no, left pad 0 !
  0L 5C                                       ! Position after line number !
?
@^UD? ! Load display mask macro !

  .US @^US%G%                                 ! Remember place, build command in S$ !
  @QT:^US%%                                   ! Append form type wanted !
  MS                                          ! Do command-put mask in buffer !
  155^T 89^T 50^T 32^T                        ! Lin 19 col 1 !
  QS,.T QS,.XM QS,.D                          ! Type mask,save it,delete it !
  155^T 89^T 49^T 32-(0^Q)^T                  ! Reposition cursor !
?
@^UU? ! Load update screen macro !

  ./76 US                                     ! No of previous lines in S% !
  QS-5"G 5US '                                ! If more than 5, say 5 !
  155^T 89^T 44^T 32^T                        ! Set cursor so typeout ends at mask !
  5-QS"G 5-QS<155^T 75^T 155^T 66^T>'         ! Clear surplus lines !
  -5TT                                        ! Do type out !
  155^T 89^T 49^T 32-(0^Q ^T                  ! Put cursor after . !
?
@^UN? ! Load renumber macro !

  J 10US                                      ! Start of file, set first number !
  <                                           ! Start of loop !
    5D QS\ QS+10US                             ! Del number, insert new one, increment  !
    0^QUR 0L 5+QR<@I%0%>                       ! Insert leading 0 as needed !
    L .-Z;                                     ! Next line, jump out at eof !
  >                                            ! End loop !
    0UN -L                                    ! Clear renum needed flag !
?

       ! Load form types !

                                              ! List of valid form types !
```

♥

UK DECUS and the
PDP-11 Commercial Sig
Announce
A RSTS/E Seminar

CARL MARBACH
DAVE MALLERY
AL CINI

3 MAJOR TOPICS

OVERCOMING FUTURES SHOCK:
VMS from a RSTS Perspective

FROM ALL ENEMIES—
FOREIGN AND DOMESTIC:
Protecting a RSTS System

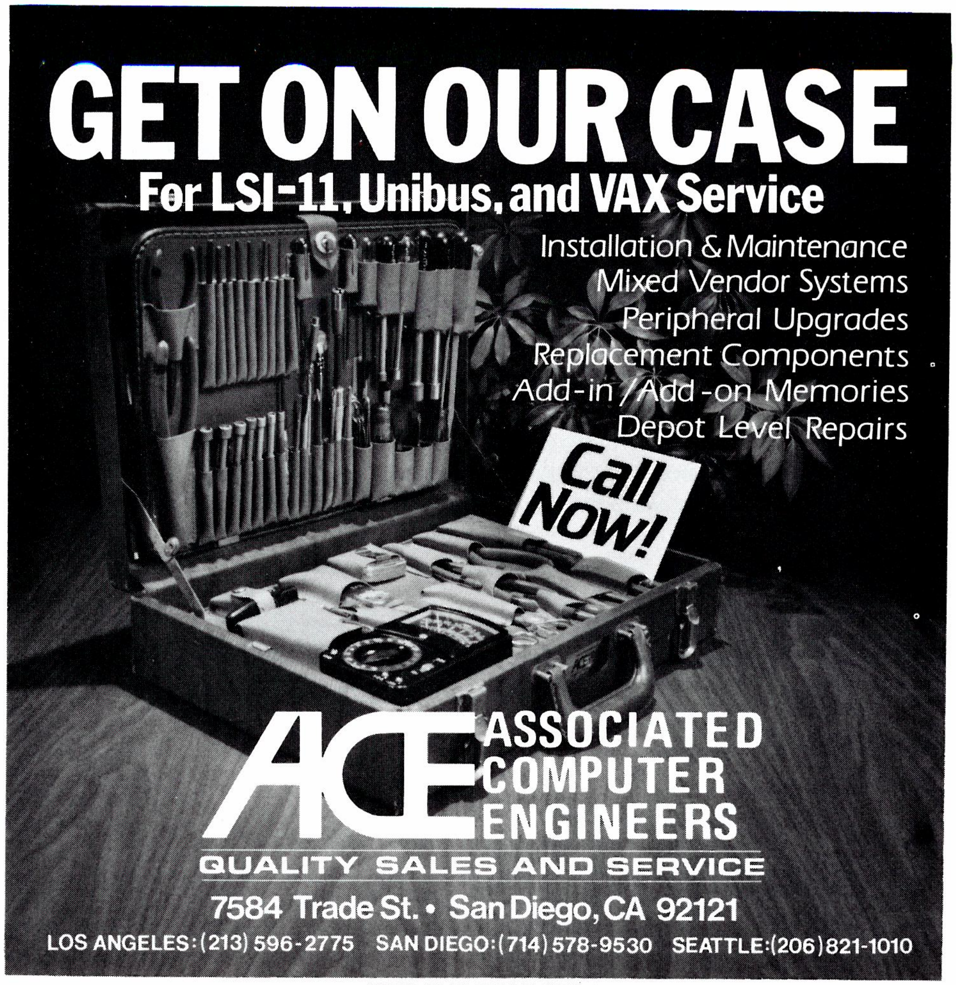
CHOOSING HARDWARE FOR
COMMERCIAL APPLICATIONS:
How to Size your System

IN LONDON

Tuesday, June 8, 1982
From 10 A.M. to 7 P.M.

Details available at the DECUS office:
Reading [0734] 583555 ext. 3229
PDP11 Commercial Sig

CIRCLE 126 ON READER CARD



GET ON OUR CASE
For LSI-11, Unibus, and VAX Service

Installation & Maintenance
Mixed Vendor Systems
Peripheral Upgrades
Replacement Components
Add-in/Add-on Memories
Depot Level Repairs

Call Now!

ACE ASSOCIATED COMPUTER ENGINEERS
QUALITY SALES AND SERVICE
7584 Trade St. • San Diego, CA 92121
LOS ANGELES:(213) 596-2775 SAN DIEGO:(714) 578-9530 SEATTLE:(206) 821-1010

CIRCLE 92 ON READER CARD

LOGIN

. . . continued from page 11

```
12070    M%(3%) = M%(3%) - 1%
         \ GOSUB 11000
         \ M%(1%), M%(2%) = 6%
         \ M%(3%) = ATT.JOB%
         \ M%(4%) = 0%
         \ M%(5%) = PROG%
         \ M%(6%) = PROJ%
         \ PRINT
         \ PRINT "Attaching to job";ATT.JOB%
         \ CHANGE M% TO LOGIN$
         \ Z$ = SYS(LOGIN$)
         \ RETURN
                 ! IF JOB IS DETACHED UNDER THIS ACCOUNT
                 ! THEN PRINT THE NUMBER OF USERS LOGGED
                 ! IN UNDER THIS ACCOUNT AND ATTEMPT TO
                 ! ATTACH TO THE SPECIFIED JOB NUMBER.

19000    !

                 ! E R R O R   H A N D L I N G   R O U T I N E


19005    E$ = CVT$$(RIGHT(SYS(CHR$(6%)+CHR$(9%)+CHR$(ERR)),3%),4%)
                 ! E$ = SYSTEM ERROR MESSGE

19010    IF      ERL = 2040%
         THEN    RESUME 2050%
                 ! IF NO MORE TEMP FILES TO DELETE, THEN
                 ! TRAP ERROR AND CONTINUE WITH THE PROGRAM.

19020    IF      ERR = 5%
         THEN    PRINT E$
                 \ RESUME 9000%
                 ! IF FILE OR ACCOUNT NUMBER SPECIFIED CAN
                 ! NOT BE FOUND ON THE DEVICE, THEN PROMPT
                 ! USER TO THIS FACT.

19030    IF      ERR = 52%  AND  ERL = 31020
         THEN    RET.LINE% = 0%
                 \ RESUME 1000
                 ! ILLEGAL LINE NUMBER
                 ! DEFAULT OF ZERO

19040    IF      ERR > 49%  AND  ERR < 53%
         THEN    PRINT "?Illegal job number"
                 \ RESUME 12030
                 ! IF JOB NUMBER TO ATTACH TO IS IN
                 ! ENTERED IN A ILLEGAL FORMAT, TRAP
                 ! FOR IT AND RESUME

19998    PRINT E$;BELL$;" at line ";ERL
         \ RESUME 9000%
                 ! END OF ERROR HANDLING ROUTINE

29999    !

                 ! C C L   E N T R Y   P R O C E S S I N G

30000    ACCOUNT$ = RIGHT(SYS(CHR$(7%)),6%)
         \ GOSUB 10000
         \ IF    ACCOUNT$ = NULL$
           THEN  ENTRY% = 0
           ELSE  ENTRY% = -1%
                 ! GET ACCOUNT # FROM CORE COMMON
                 ! OBTAIN JOB STATUS DATA
                 ! DETERMINE IF PROJECT-PROGRAMMER # HAS BEEN ENTERED

30010    GOTO 1000
                 ! ENTER INTO MAIN PROGRAM

30999    !

                 ! C H A I N   E N T R Y   P R O C E S S I N G

31000    CR$ = CHR$(13%)
         \ RET.PGM$ = NULL$
         \ RET.LINE% = 0%
         \ COMMON$ = SYS(CHR$(7%))
         \ GOSUB 10000
         \ P% = INSTR(1%,COMMON$,CR$)
         \ IF    P%<>0%
           THEN  31010
           ELSE  ENTRY% = 0%
                 \ GOTO 1000
                 ! SET CR$ = <CR>
                 ! SET RETURN PROGRAM TO NULL
                 ! SET RETURN LINE TO 0
                 ! GET CORE COMMON
                 ! OBTAIN JOB STATUS
                 ! IS ANYTHING IN CORE COMMON?
                 ! IF YES, SEE WHAT IT IS
                 ! ELSE SET ENTRY TYPE AND PROCEED

31010    ENTRY% = -1%
         \ ACCOUNT$ = LEFT(COMMON$,P%-1%)
         \ COMMON$ = RIGHT(COMMON$,P%+1%)
         \ P% = INSTR(1%,COMMON$,CR$)
         \ IF    P%<>0%
           THEN  31020
           ELSE  1000
                 ! GET ACCOUNT
                 ! DELETE ACCOUNT FROM CORE COMMON
                 ! IS THERE MORE IN CORE COMMON?
                 ! IF YES, SEE WHAT IT IS
                 ! ELSE PROCEED WITH MAIN LINE

31020    RET.PGM$ = LEFT(COMMON$,P%-1%)
         \ COMMON$ = RIGHT(COMMON$,P%+1%)
         \ P% = INSTR(1%,COMMON$,CR$)
         \ IF    P% = 0%
           THEN  1000
           ELSE  RET.LINE% = VAL(LEFT(COMMON$,P%-1%))
                 \ GOTO 1000
                 ! GET PROGRAM TO RETURN TO
                 ! SEE IF LINE NUMBER TO CHAIN TO
                 ! IF NOT, PROCEED
                 ! ELSE GET LINE NUMBER AND PROCEED

32767    END
```

RPTMAN — REPORT MANAGER

By Jonathan M. Prigot, Systems Programmer, Polyfibron Division, W.R. Grace and Company, Lexington, MA

The report manager program, RPTMAN, is designed to allow users to organize data from a file and print the organized data.

RPTMAN allows the user to format the data horizontally, vertically, sort on any given field, generate a number of pre-formatted reports, or generate various forms such as cutting tickets, acknowledgements, etc.

1. REPORT MANAGER SELECT SCREEN

RPTMAN is entered from the DCS select screen. It presents the user with the option of generating a horizontal list (HL), vertical list (VL), sorted horizontal list (SL), generate a pre-formatted reports (PL), or generate forms such as cutting tickets, etc. (FM). The abort option (AB) is also provided to allow the user to return to the main DCS program.

2. SELECTION CRITERIA SCREEN

Selecting HL, VL, or SL will bring the user to the SELECT SCREEN. The select screen is used to get general information on the input data file, the output file/device, and whether you wish to limit the range of the report.

The items on the select screen (and their meanings) are:

1. RDF INDEX — The default for this item is LIB:RDF.VIR. If you enter an invalid RDF specification, RPTMAN will erase the invalid entry.
2. DATA FILE — The name of the data file to use (e.g. R756)
3. POINTER FILE — (used only if SL was selected) This is SL's workfile. The default name is the data file name +.PTR.
4. DELETED/QUEST RECORDS — Default is <cr> (i.e. print no deleted records and no questionable records.) If you do wish to display either deleted records and/or questionable records, respond with D and/or Q to this option, else just enter a <cr> (carriage return).
5. OUTPUT FILE — Default is LP: (the main 'line-printer'). Where to 'print' the report. This can either be a file, a terminal screen or a printer. The output can be directed to the printer attached to a Datamedia DT80/1 terminal by specifying KBZ: NOTE!: If you specify a filename for this item, it will be put in your assigned account.
6. MAXIMUM WIDTH — Default 80 for KB:, 132 otherwise. Maximum width of the output device.
7. FORWARD/BACKWARD LIST — Default is forward. Since records are stored on the system in order of creation date, you can sometimes get your report faster by asking for a backwards listing. This is especially true if the data you desire is recent.
8. HEADING OF LIST — Default is the name of the data file.
9. LIST BLANK ELEMENTS — Default is no. This will suppress the print of blank fields in a VL.
10. LIMIT SEARCH — Default is no. If you wish to limit the scope of the report, respond with Y<cr> to this. It will invoke the SELECTION CRITERIA screen.

Assuming that you responded to the LIMIT SEARCH question with either a <cr> or N<cr>, RPTMAN will then ask you to CONFIRM ALL SELECTIONS. If you respond with anything other than a Y<cr>, RPTMAN will blank the SELECTION CRITERIA screen to allow you to re-enter the data.

3. SELECT FIELDS

The SELECT FIELDS screen is invoked by answering Y<cr> to the LIMIT SEARCH question on the SELECT CRITERIA screen. This screen is used to specify the characteristics the fields within the record must have in order to be listed.

The questions on this screen are:

1. USE CREATE DATE — Default is no. This question allows you to select records created within a certain period of time. If you respond with a Y<cr> to this question, you will be further prompted with:
2. AFTER — This is the date of the earliest record you want. Separate the fields within this question by typing a <cr> after day, after month, and after year. Entering a <cr> alone for the date means use the earliest record in the file.
3. BEFORE — The date of the last record you want. Operation same as AFTER. A <cr> for the day means use the latest date in the file.
4. FIELD NUMBER OR NAME — The user can enter either the field number (e.g. F3001.1.R789) or any part of the field name (e.g. DUE DATE). If there is more than one field containing the specified name, or if RPTMAN cannot find the field you specify, it will so inform you and reposition the cursor for another trial. Entering a <cr> alone ends field specification.
5. BETWEEN — Sets the lowest value allowable.
6. AND — Sets the highest value allowable.

After you enter <cr> to terminate select field specification, the system will then ask you to confirm your selections. If you enter anything but Y<cr>, the system will erase the screen and allow you to redo your selections. Once you confirm your selections, the system will ask you to confirm all your selections. If you respond with anything except Y<cr>, the system will return to the SELECTION CRITERIA screen for re-entry.

4. VERTICAL LISTING (VL) SCREEN

The VL program will inform you that it is [WORKING].

5. HORIZONTAL LISTING (HL) SCREEN

The HL program screen is used to select what fields within the record will be printed, and whether the contents of the field are to be counted or totaled. Those fields that are designated as alphanumeric fields are counted, while those that are numeric are totaled.

The screen prints out the field number and the field name, then waits for your response. The legal responses are:

1. <cr> — Do not list this field.
2. Y — List this field and use the field name on the report.
3. heading — Use this title for the field heading.
4. resp/T — List this field using either the default heading or this heading (per items 2 and 3 above), and give field count or total at the end of the report.
5. LAST — Do not print this field or any field that comes after this field.
6. REST — Print this field and all the other fields that come after.
7. up-arrow key — Back up one field to allow re-selection.

After all desired fields have been listed, the user will be returned to the SELECT SCREEN, unless the report was sent to the user's keyboard, in which case RPTMAN will ask the user to type a <cr> to continue.

6. SORTED LIST (SL) SCREENS

There are two screens associated with the SL program:

The SORT SPECIFICATION SCREEN, and the FIELD SELECTION SCREEN.

The SORT SPECIFICATION screen specifies the fields to sort and the direction to sort them in. As in the select screen above, either field number or all or part of the field name can be used to specify the field. The order of the sort can be either ascending or descending for the individual field.

The FIELD SELECTION screen is similar to the HL screen, with the addition of a SUBTOTAL (/S:) option. The /S: option operates in a similar fashion to the TOTAL (/T) option in that it provides a sub-count or sub-total of the field. It is used by appending a /S:trigger-field-name-or-number to the Y or field-heading specification. You may either specify the trigger field's number or a portion of its title. If the field specification is not acceptable, you will be notified, and allowed to re-enter your specification.

During the running of the program, the selected field will be subtotaled whenever the contents of the associated trigger-field changes.

After all fields have been specified, the program will prompt for whether you want the report in report format or tape format. Report format has page numbering and report and field titles. Tape format does not; it is pure data. If you choose tape format, you will be further prompted as to whether to separate the output with spaces or commas between the data. Because the output from the tape option is usually used as input to another program, your response to this question depends on what the next program requires.

BIBLIOGRAPHY

[1] R.R. Jaques, A. Eloy, "Design of a Data Base Management System for W.R. Grace and Company" December 1981, Fall DECUS U.S. symposium. ♥

PDP-11® & VAX® users...

ComDesign's TC-3 Concentrates Multiple Terminals Over One 1 Phone Line . . . Without a DZ11!

● COST SAVINGS TO 90%
Totally eliminates DZ11s, extra modems and phone lines.

● STATISTICAL MULTIPLEXING
Up to eight 9600 bps terminals over 1 phone line.

● ERROR CORRECTION
Phone line errors eliminated automatically.

● PROVEN RELIABILITY
Over 3 years MTBF.

● FAST ECHO RESPONSE

● CONCENTRATION
Multiple terminals—one line.

● COMPLETELY TRANSPARENT
No software changes needed.

● BUILT-IN SYSTEM MONITORS
Real-time data and signal displays, statistics and error counters.

● POWERFUL DIAGNOSTIC PANEL
Modem loop-back, unit self-tests, and terminal diagnostics.

Only $240/mo. for a TC-3 . . . Connect 8 Terminals on 1 Phone Line, Without a DZ11.

For further information and the sales office for your area call toll-free (800) 235-6935 or in California (805) 964-9852.

ComDesign

ComDesign, Inc.
751 South Kellogg Avenue
Goleta, California 93117

® Registered Trademarks of DEC

CIRCLE 46 ON READER CARD

Did Your PDP-11 Die

When you tried to run WORD PROCESSING?

Now There is a WP System Designed Specifically for the DEC User

With the WP Saturn Word Processing/List Processing Software, you have low overhead, quick response and power.

DO WHAT SO MANY HAVE DONE:

COME TO SATURN
FOR THE ANSWERS

- Faster and more powerful than the stand alones
- Faster and friendlier than other computer based systems
- More features to solve your word and list processing problems
- Extensive help commands
- Saturn runs on RT-11/TSX, RSX, RSTS, IAS, VMS
- Saturn's ASCII Format interfaces to standard DEC files
- Assembler Language provides maximum speed and efficiency
- User Friendly Prompting helps you master word processing with minimal training
- Extensive List Processing: Forms, prompted data entry, and sort/select; plus flexible merging of lists and documents

WP SATURN FEATURES

FILE FUNCTIONS: Automatic backup • Uses ASCII source files • File directories location, type, date • File Merge • Password security • Print formatted file • Review formatted file • Store formatted file

SCREEN EDITOR: Advance-Forward & Backward • Automatic Carriage Return • Column Operations • Copy, Cut, Delete • Cursor Controlled • Display of Operator Actions • Formatted Text Displayed • Vertical and Horizontal Scroll • Load from another document • Paste • Prompts • Replace-Global or Selective • Search

TEXT FORMATTING: Automatic Hyphenation, Formula, User defined dictionary • Automatic Index & Table of Contents • Automatic Outline Formats, Outline, Decimal • Centering • Date Insertion • Headers & Footers • Justification • Line Indent • Line Spacing • Multilevel Superscripts & Subscripts • Overstrike, Underscore • Page Numbering • Page Size • Page Test • Paragraph Indent • Redefine Format Values • Reformat Without Retyping • Table Protect (no page breaks) • Tabs, Left, Right, Any character • Variable Spacing • Word Wrapping • Footnotes

LIST PROCESSING: Form Generation • Prompted Data Entry • Multi-Key Sort/Select/Exclude • Record Search • Sequential, Binary • Unrestricted Field Insertion • Field replacement, Global, Selective

For additional information call toll free 1-800-328-6145

SATURN SYSTEMS
6875 Washington Avenue South, Suite 218, Edina, MN 55435 (612) 944-2452

CIRCLE 26 ON READER CARD

BACK ISSUE OFFER
ALL 10 BACK ISSUES
OF
THE RSTS PROFESSIONAL

$75.00

Send check to:
THE RSTS PROFESSIONAL
BOX 361
FT. WASHINGTON, PA 19036

PAYMENT MUST ACCOMPANY ORDER

LETTERS

. . . continued from page 71

privileges, if any. This can be done in CALLER by adding another DATA element to each command description and dropping temporary privileges if indicated.

Last year we were faced with the problem of not enough small buffers on a system with 30+ jobs and in excess of 200 CCL's. While the number was sufficient at the time, the system was still growing and we had set the number of small buffers to the maximum when doing the systems, At the time, it was decided that regardless of the solution decided upon it was absolutely necessary that it meet the following requirements:

1. Any additional system overhead must be negligible.
2. The change must be transparent to the users.
3. Abbreviated commands were to be allowed.
4. The command must execute the same as a CCL.

The final solution, arrived at mainly due to the necessity of not adding any overhead to the system, was to create a new run-time system which contained the CCL commands and also served as a keyboard monitor. It was made permanently resident and took 2K words of memory. A minor change to the LOGIN program and it became the job's private default run-time system at login time. All of the above requirements were met and upon benchmarking its efficiency, we found it to be faster and used less CPU time than a normal CCL.

Anyone interested in this approach may write to us. We are also enclosing our renewal subscription for the forthcoming year. The articles in the *RSTS Professional* are excellent and I look forward to seeing it 6 times annually.

Keep up the good work.

D.D. (Bud) Mundy, President
DMD Computer Consultants Inc.
Agincourt, Ontario

—

Thank you for the honorarium I received for my article in the Dec. 1981 *RSTS Professional.* It was totally unexpected. As contributing to your magazine was a group effort, I have given this honorarium to my company's Children's Hospital of Pittsburgh charity drive.

Once again, thank you for letting us participate in the *RSTS Professional.* I look forward to working with you again in the future.

David Froble
Senior Technical Consultant
Transcomm Data Systems, Inc.

—

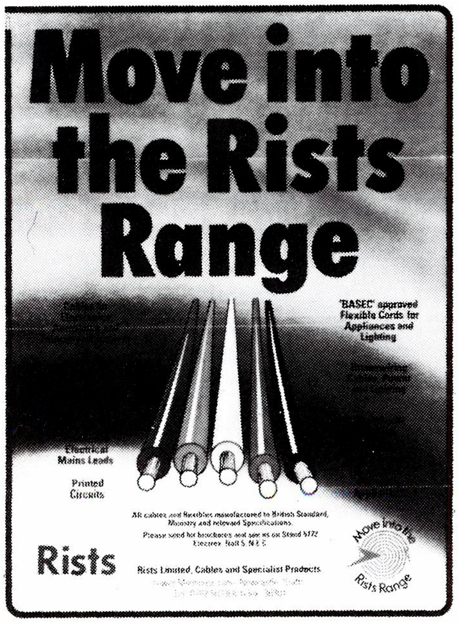
I am very grateful for the first copy of the journal '*RSTS Professional*' which I have just received. It is indeed a very impressive journal and I am sure that it will serve us well in our work.

Dr. S. Ron, Head of Institute
Tel-Aviv Univ. Medical School, Israel

We're here to be of service, Dr. Ron.

—

We could not find any Tea Co's, but will this do?


Move into the Rists Range
'BASEC' approved Flexible Cords for Appliances and Lighting
Electrical Mains Leads
Printed Circuits
All cables and flexibles manufactured to British Standard, Ministry and relevant Specifications.
Please send for brochures and see us on Stand 5172 Electrex, Hall 5, NEC
Rists
Rists Limited, Cables and Specialist Products
Move into the Rists Range

Thanks for a great magazine.

Nick Wright & Mark Itzcovitz
Piccadilly Computer Services Ltd.
Stanmore, Middlesex, UK

P.S. See you at this year's DECUS UK Commercial SIG?

Here we go again!!!
P.S. Yes. (And thanks for that great ad above.)♥

CLASSIFIED

Send Classified Ads to: RSTS Classified, P.O. Box 361, Ft. Washington, PA 19034-0361. $1⁰⁰ per word, first 12 words free with one year's subscription. [Be sure to include a phone number or address in your message.]

NEW 11/70 priced to move quickly. Call for specs (812) 479-6951.

GOOD DEAL on DECNET/E software license. Call Online Data Processing, Inc. (509) 484-3400.

FOR SALE Two RL02's in H9642 cabinet, controller, bootstrap ROM,11 cartridges. 1 year old. DEC maintained, DEC deinstalled. Available immediately. Asking $7000. Call Bret (814) 849-3057.

TIMESHARING AVAILABLE on PDP 11/70 RSTS. Software also. Call Wabash Time Share in Indiana (317) 448-1686.

Rapidly expanding Southern California Company with good benefits has openings for experienced DEC software systems programmers to work on high level database management system. 3-5 years PDP-11, RSTS/E, Basic-Plus, BP2 and MACRO required. Must have thorough understanding of RSTS/E operating system. VAX/VMS and VAX/BASIC desirable. Must be willing to relocate. Send resume with salary requirements to:

NCCS, Inc.
2235 Meyers Ave. Escondido, CA 92025

RSTS RESCUE SQUAD

We salvage all kinds of disasters:

- unreadable disks
- ruined UFDs and MFDs repaired
- immediate response
- telephone DIAL-UP
- on-site
- software tools
- custom recovery
- 90% success to date
- more than 1 GB rescued to date

Brought to you by
On Track Systems, Inc.
and a well known (and read)
RSTS expert.
CALL 24 HOURS
215–542-7133

LOOKING FOR DEVELOPMENT TIME?

NO KILOCORE TICK CHARGES
NO CPU CHARGES
RSTS/E TIME

$7 PER HOUR CONNECT TIME

BASIC PLUS
BASIC PLUS, PASCAL, "C" } WITH CROSS COMPILER SUPPORT
WORD-II WORD PROCESSING
WAFE, TECO, EDT } PROGRAM EDITING

BUDGET BYTES
212-944-9230

OMNICOMPUTER, INC.
1430 Broadway, New York, N.Y. 10018

DEC BEST VALUES

PRE-OWNED DEC EQUIPMENT

BUYING AND SELLING

SYSTEMS • CPU's • PERIPHERALS • TERMINALS
OPTIONS • MEMORY • COMPATIBLES

CALL DICK BAKER (305) 979-2844

dataware incorporated
Carico Center
2845 NW 62nd Street
Ft. Lauderdale, Florida 33309
Telephone (305) 979-2844

GROSSMAN GRAPHICS
215
537-0782
5041 frankford
phila., pa.
19124

It's 3:00 am. Do you know who is on your computer? Lots of Luck from Lock-11.

LAUSD thanks Kevin Herbert & RSTS Pro for Term Links on V7.

Interested in used DEC equipment. Contact Liz Endicott, P.O. Box 948, Kilken, TX 76541.

ABLE DH/DM's, CACHE 4/34, backplanes and other goodies. Dave Mallery (215) 364-2800

For Sale, PDP-11/34A, 128KW, RL01's, RX02's, DZ11, LA180, VT100's like new. (205) 852-8950.

For Sale, One (1) 11/40 CPU and RP04 Disk-96KW. (403) 488-9671.

SAVE YOUR quarters, games for RSTS/E are here. Interactive, real-time games for VT52 and VT100 terminals. Some of the games available:

BLKADE: As many as eight players try to force each other to collide with their growing tails on the screen.

STRWRS: You have five minutes to destroy the Death Star with your single X-wing fighter.

SUBS: Two players manuever around islands in an attempt to sink the other player's submarine.

Games come on 9-track magtape with their own high-quality user manuals. Order the first game for $39.95, $29.95 for each additional game. Individual manuals available for $9.95 each.

Send your check, or write or call:

INFINITY SOFTWARE CORPORATION
2210 Wilshire Blvd.
Suite 801
Santa Monica, California 90403
(213) 820-2702

Ready for ABLE?? Call Dave — he's ready and ABLE. (215) 364-2800.

Backup sites needed PDP11-RSTS, Los Angeles area, call Jim Cecil 213—926-0519. Will Reciprocate.

The *FAMOUS* RSTS PROFESSIONAL TEE-SHIRT is now for sale!

Send size desired and $6.95 for each shirt to:
RSTS TEE-SHIRT
P.O. Box 361
Ft. Washington, PA 19034-0361

Shirts available in adults sizes only: Small - Medium - Large - X-Large

11/40, reborn, but now retired. CPU with RSTS options. CHEAP. Dave Mallery (215) 364-2800

Expert consulting, programming services. RSTS/E business/financial applications. Practicomp, (415) 665-0628, San Francisco.

Still $200!

MORE FEATURES!

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

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

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

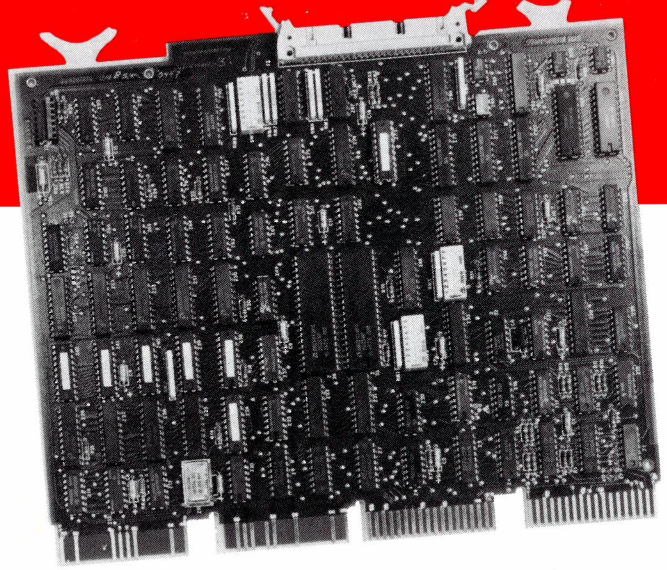
User's manual and source code included.
9-Track $200. RKO5 $260. ppd
T. F. Hudgins & Associates, Inc.
P.O. Box 10946, Houston, Texas 77018
Woods Martin 713/682-3651

*TM Digital Equipment Corporation

PLAN PLUS FINANCIAL PLANNING SYSTEM

RSTS/E, $4,500.
PLAN-PLUS LIMITED
P.O. Box 327
London, W4 4QD
Tel. UK 01 834 7334, Ext. 7935
[24 Hours]
Telex 8813271 GECOMSG

DEC®-COMPATIBLE PERIPHERAL CONTROLLERS



Dataram Corporation offers the industry's widest range of DEC-compatible peripheral controllers — from comparatively simple NRZI tape controllers to complex 300 MB storage module drive (SMD) controllers.

An impressive array of state-of-the-art controllers, all built around high-speed bipolar microprocessors. All software compatible with the host LSI-11®, PDP®-11, or VAX® minicomputer...and all available now.

And Dataram's controllers are designed to save you money, and, more importantly, space — our controllers typically occupy half the space required for the comparable controller from DEC. Doing it with a level of performance that makes any member of this family worth looking at.

The chart shows our current family of peripheral controllers, growing every day. If you don't see the controller you need, we're probably working on it right now. Call us and discuss your requirements.

CONTROLLER	DESCRIPTION	COMPATIBILITY
C03	Cartridge disk controller	RK05
C33	Cartridge disk controller	RK05
T03	NRZI mag tape controller	TM11/TU10
T04/N	NRZI mag tape controller	TM11/TU10
T04/D	Dual density mag tape controller	TM11/TU10
T34/N	NRZI mag tape controller	TM11/TU10
T34/D	Dual density mag tape controller	TM11/TU10
T36	Dual density mag tape controller	TM11/TU10
S03/A	80MB/300MB SMD controller	RM02/RM05
S03/A1	160MB SMD controller	RM02
S03/B	80MB/300MB SMD controller	RK07
S03/C	200MB/300MB SMD controller	RP06
S03/D	96MB CMD controller	RK06
S33/A	80 MB/300 MB SMD controller	RM02/RM05
S33/A1	80 MB/160 MB SMD controller	RM02
S33/B	80 MB/300 MB SMD controller	RK07
S33/C	200 MB/300 MB SMD controller	RP06
S33/D	96 MB CMD controller	RK06

Products printed in red are LSI-11 Bus compatible.
Products printed in black are UNIBUS® compatible for PDP-11 and/or VAX minicomputers.

DEC, LSI-11, PDP, UNIBUS and VAX are registered trademarks of Digital Equipment Corporation.

DATARAM CORPORATION

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

Canada: Ahearn & Soper Ltd., 416-245-4848 • **Denmark:** Technitron ApS, 02 96 98 22 • **Finland:** Systek OY, (80) 73 72 33 • **France:** YREL, (03) 956 81 42 • **Hungary/Poland/Rumania:** Unitronex Corporation, WARSAW 39 6218 • **Italy:** ESE s.r.l., 02/607 3626 • **Netherlands:** Technitron b.v., (020) 45 87 55 • **Sweden:** M. Stenhardt AB, (08) 739 00 50 • **Switzerland:** ADCOMP AG, 01/730 48 48 • **United Kingdom:** Sintrom Ellinor Ltd., (0734) 85464 • **West Germany:** O.E.M.-Elektronik GmbH, 07 11-79 80 47 • **Yugoslavia:** Institut "Jozef Stefan", 263-261 • **Australia/New Zealand:** Anderson Digital Equipment, (03) 544-3444 • **India:** Infosystems Private Limited, 79281 • **Israel:** Minix Computers & Systems Ltd., 03-298783 • **Japan:** Matsushita Electric Trading Co., Ltd., 06 (282) 5111 • **Taiwan:** Rabbit Associates, Ltd., 7219573-5 • **Hong Kong:** Automated Systems (HK) Ltd., 5-630256-9 • **Malaysia:** Automated Systems (M) Sdn Bhd., 773777 • **Indonesia:** P. T. Daya ASL, 584306 • **Singapore:** Automated Systems (PTE) Ltd., 2354133

CIRCLE 55 ON READER CARD

Thousands sold worldwide
since 1978

Super Max

The single-board, 16-line ABLE DH/DM™ that enhances any UNIBUS system with DH-performance at DZ-prices.

A few years ago, we broke new ground with our DMAX/16™, the original alternative to the DEC DH11. DMAX cut the space requirements from nine slots to three and became an immediate worldwide success. Now we've come up with something even better. This time it's ABLE DH/DM™, an alternative that achieves the optimum cluster size – 16 lines with modem control on a *single* board. You can compare MTBF, price and throughput and find that ABLE DH/DM beats everything in its class. No one else comes close.

The ABLE DH/DM™ is today's answer to VAX system needs for DMA communications multiplexing and serves all standard UNIBUS systems equally well. Each 16-line ABLE DH/DM™ installs in any standard hex-width slot at only one unit bus load and is DH11 compatible to the diagnostic level. Just plug it in and see it run – up to 19.2K baud using only half the UNIBUS bandwidth of a DEC DH11.

Key ABLE DH/DM™ features include on-board diagnostics with LED display, modem control on all lines, improved on-board silo depth and variable PROM set for proprietary OEM applications.

Keep up with ABLE and optimize your VAX, PDP-11 or System 20. Write or call today for details on our full line of UNIBUS-compatible special-memory, general-purpose and data-communications products plus the MAGNUM™ computer series.

***Now enhances VMS operation!**

ABLE the computer experts

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

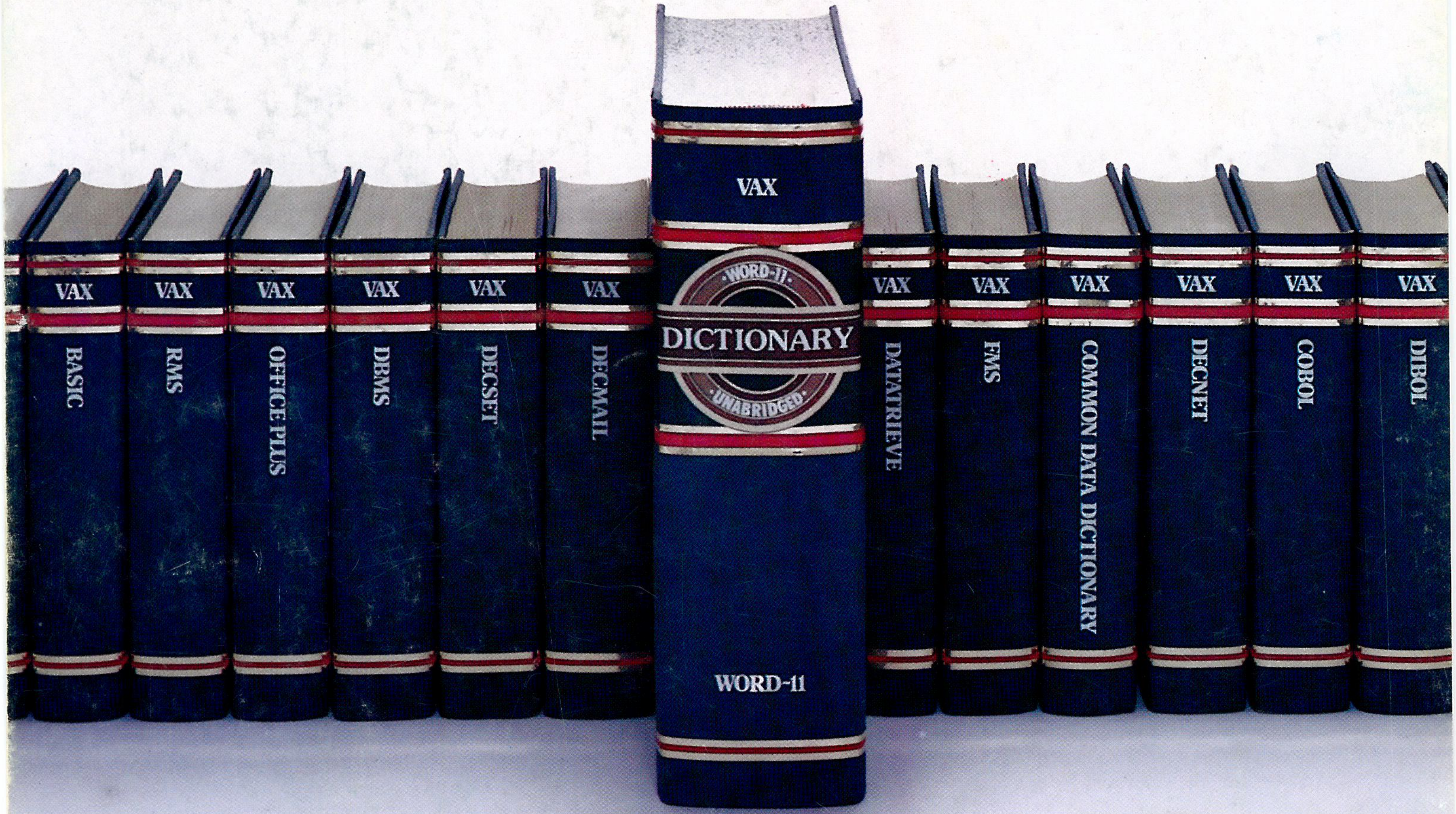
ABLE COMPUTER, ABLE Computer House,
London Road, Newbury, Berkshire, England RG13 2QJ.
44(0635) 32125. TELEX 848715 ABLE G.

ABLE COMPUTER GmbH,
Forsthausstrasse 1, 8013 Haar (Near Munich), West Germany.
49 089/463080, 463089. TELEX 05213883 ABLE D.

DEC, UNIBUS, VAX and PDP are trademarks of Digital Equipment Corporation.

CIRCLE 56 ON READER CARD

WORD-11.
The Word Processing System Vital For VAX.



The Unabridged Addition.

No matter how many additions you've made to your VAX,™ it won't be complete without WORD-11.™ It's the sophisticated word processing system designed to help you use all the other information you've stored. With features like list processing, built-in dictionaries for spelling error detection, automatic table of contents and footnoting that make report writing a snap.

WORD-11 runs concurrently with data processing on multiple terminals. It's easy to use. And it's been up and running in hundreds of installations all over the world for years.

WORD-11 is easy to install and operate. It's cost effective. And it's supported by a dedicated team of experienced specialists. It could be the most important addition you make to your VAX installation. For details, Call, (714) 993-4160.

WORD-11 is also available for Digital's RSX-11M,™ RSX-11M PLUS™ and RSTS/E™ operating systems.

Data Processing Design, Inc.
AUTHORIZED digital COMPUTER DISTRIBUTOR

CORPORATE OFFICE
181 W. Orangethorpe, Suite F
Placentia, CA 92670
714-993-4160 Telex 182-278

N Y OFFICE
420 Lexington Avenue, Suite 633
New York, NY 10170
212-687-0104

WASHINGTON D C OFFICE
4520 East-West Highway, Suite 550
Bethesda, MD 20814
301-657-4098

WORD-11 is a trademark of Data Processing Design, Inc.
VAX, RSX-11M, RSX-11M PLUS, and RSTS/E are trademarks, and remaining titles are products or trademarks of Digital Equipment Corporation.

CIRCLE 59 ON READER CARD