RSTS PROFESSIONAL

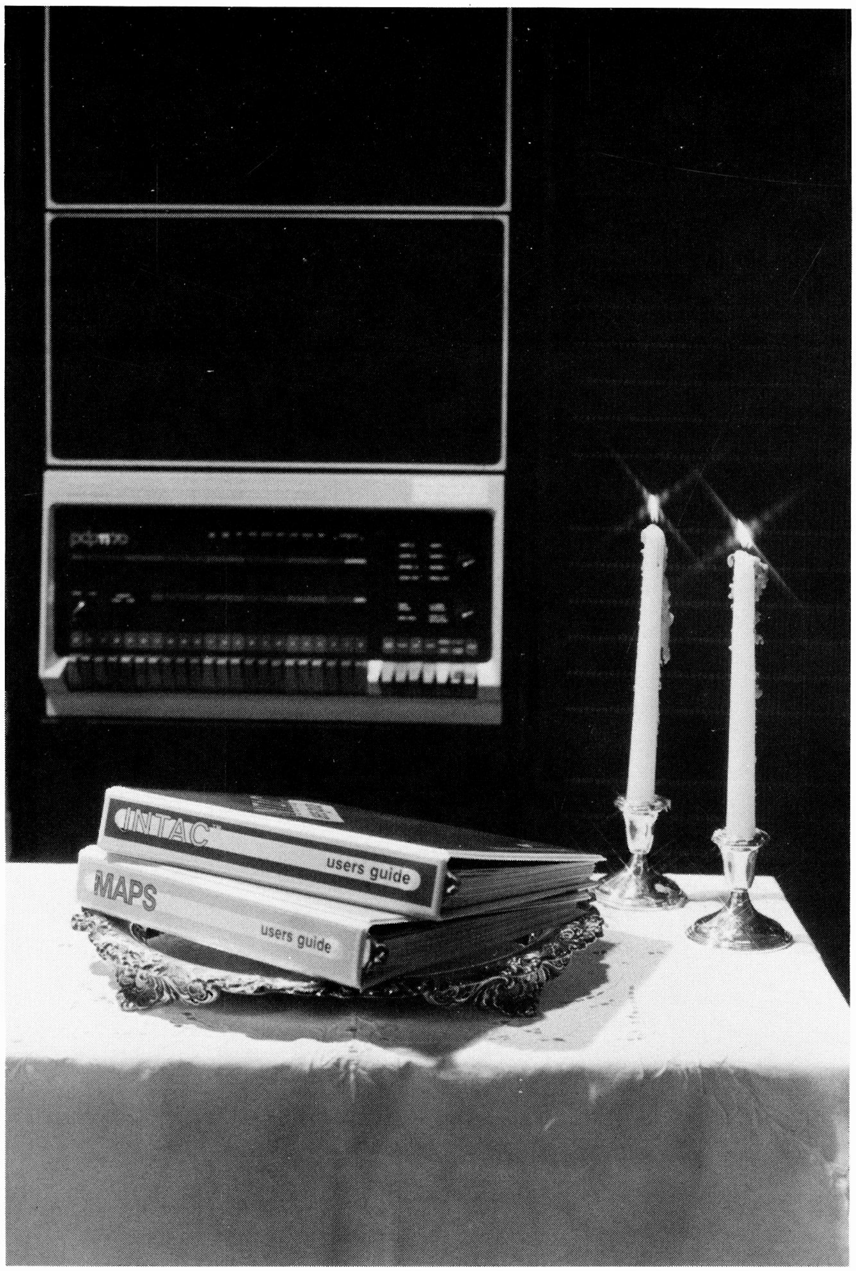
Volume 2, Number 4

December 1980

$7.50/issue, $20.00/year

I'M A RSTS/E LOVER
digital
I'd rather do it in BASIC!
When users talk, RSTS Listens.
TKB FOREVER
EVER AND EVER AND EVER AND EVER
Mulvey in '80
Cobol S
LOVE SHARING MY TIME WITH RSTS/E
digital
"Don't take chances, bet on RSTS/E" Las Vegas
DECUS
7.0 on the RSTS scale
RSTS/E HAS ARRIVED
digital
RSTS/E is a perfect "10"

INSIDE:



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

INTAC™
MAPS™

Interactive Data Base Management

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

Financial Modeling

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

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

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

Call us collect for more information.

ross systems
incorporated

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

If you're looking for speed, a fast track and reliability, enter the Southern 600.

Enter the 600 line-per-minute band printer that costs less than somebody else's 300 lpm printer.

The Southern Systems 600 lpm band printer is about as good as you are going to get.

We are the only company that designs, manufactures and services every printing system with a Southern Systems name on it. Our computer-to-printer interfacing covers the full PDP* line, including VAX 11/780, DECsystems 10* and 20,* as well as Hewlett-Packard, TI, Perkin-Elmer, Burroughs, Data General and others.

You get the most reliable hard copy output at the lowest cost of ownership on the market. Parallel or serial.

We have the technology and manufacturing skills that make our reliability, economy, prompt delivery and service a reality.

Southern Systems. We're leading the pack.

CALL TOLL-FREE: 800-327-6929

SSI SOUTHERN SYSTEMS, INCORPORATED
2841 Cypress Creek Rd., Ft. Lauderdale, FL 33309
(305) 979-1000; (800) 327-5602; Telex 522135

*Registered trademarks of Digital Equipment Corp.

O.K., Speedy, tell me about the following printer system(s):

□ 200 lpm impact matrix
□ the B series (300 or 600 lpm band)
□ the 2200 series (300, 600, 900 lpm drum)
□ the 1200 series (600 to 1200 lpm Chain Train)
□ the 2550 (1500 lpm charaband)
□ Serial Interfacing
□ Parallel Interfacing

My computer is a ____________

My requirements are: ____ immediate ____ 3-6 months ____ information only.

Name ____________ Title ____________

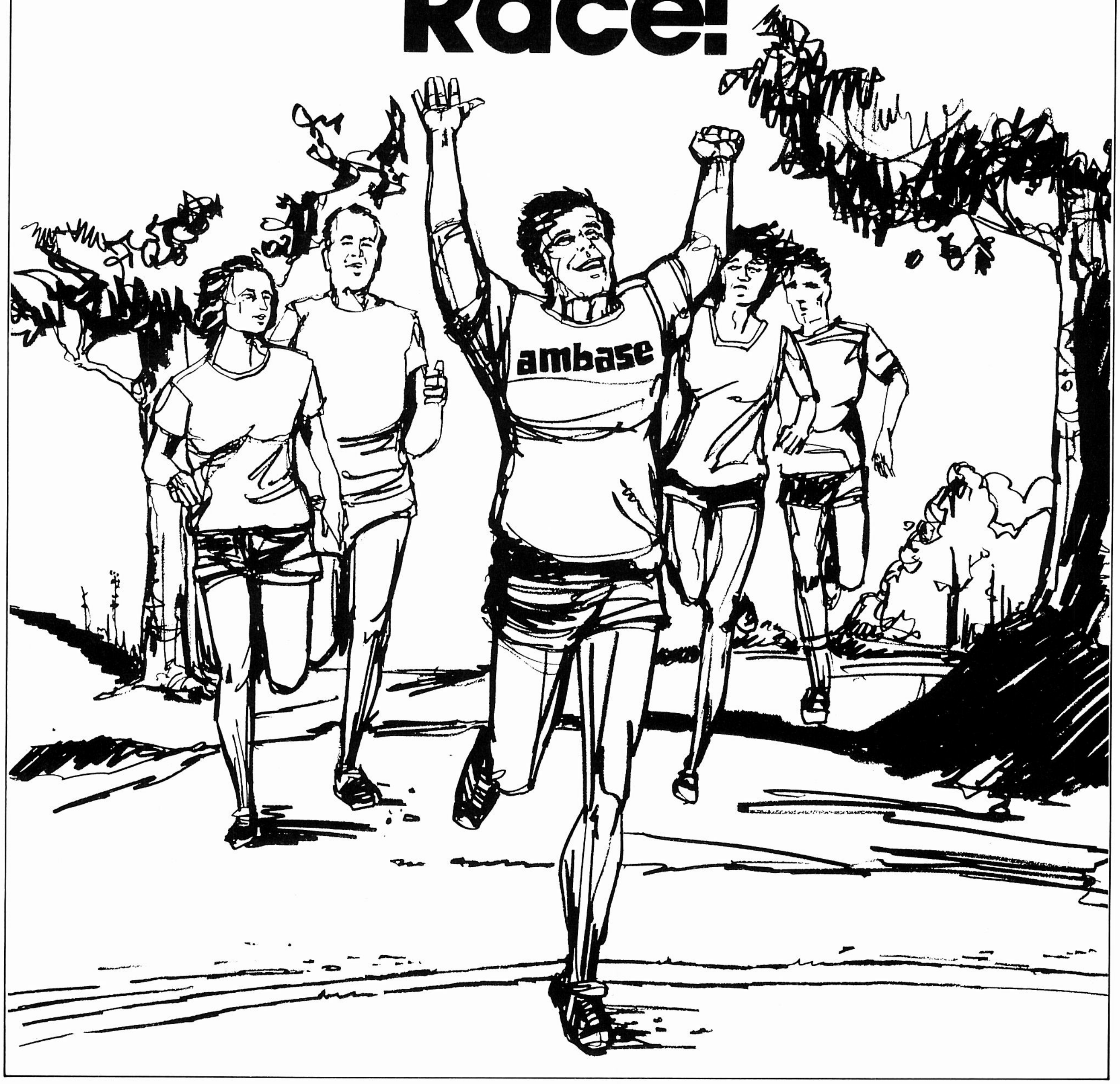
Company ____________

Address ____________

City ____________ State ____________ Zip ____________

Telephone (   ) ____________

Win the Productivity Race!



with **ambase**™ the DBMS that is increasing programming productivity in RSTS shops worldwide from 100% - 900%.

amcor computer corp.

1900 PLANTSIDE DRIVE • LOUISVILLE, KY. 40299 • USA • 502/491-9820
REGIONAL OFFICES • ATLANTA, GA. • SAN JOSE, CAL.

Contents

Coming . . .

- Cook Book "How to Structure Your Disk
- Dave and Carl Performance Measurement and Enhancement
- CNTRL F (Mini SYSTAT enhanced to show file statistics)
- System Managers Notebook
- Disk Directories
- The RSTS Professional Buys a Computer
- More on the Future of RSTS
- The VAX-SCENE
- String Handling Explained
- More . . .

The RSTS Professional Magazine, December 1, 1980, Vol. 2, No. 4. Published quarterly. Single copy price $\$7^{50}$, $\$20^{00}$ per year. Published by M Systems, Inc., 753 Johns Lane, Ambler, Pa. 19002, telephone (215) 542-7008. Send all correspondence and change of address to: Box 361, Ft. Washington, Pa. 19034. Application to mail at second class postage rates pending at Ambler, Pa. 19002.

From the editors ...

While Version 7.0 of RSTS and the Large (new) file option have proved popular among the user community, the problem of small buffers has plagued users with 64 or more ports (even less) on their systems. While we understand that this only happens to 11/70's, and larger ones at that, never the less there is a nagging software problem, made worse in V7.0, that is keeping many of us from achieving the advertised specification for RSTS/E that it will support up to 63 Jobs. No way.

Now, everyone makes mistakes; but most of us try to rectify them as fast as we can. Many of us thought that there would be a V7.1 or V7.01 that would relieve our small buffer crunch. There were rumors about that the code to move small buffers to XBUF or SMBUFF was already in the sources in conditional assembley code waiting to be tested and implemented. Wrong. At the San Diego DECUS the head of RSTS development reported, "no new Version before next FALL". This means that it will be two years, or more, between releases of RSTS. Under some circumstances this would be acceptable, now it is not.

At least two sites I know of have recently reverted to 6C in order to keep going. Some others are at a standstill, plans to expand crushed by not enough small buffers. Machines planned to have 60 terminals are stopped dead at 50. And,"NOT BEFORE NEXT FALL". Of course we now have INDENT, and GIGI and more; but our current house is not in order. I have seen the RSTS SIG wish list, and some of the ideas are great, but surely FIXING the small buffer problem is not a wish; it is REQUIRED.

Before the small systems people jump up and down and say, "but this is only for you big guys", I would like to say that when and if any new version prevents you from doing things you USED TO DO, or wouldn't allow you to do things you were supposed to do, I would ask for immediate relief. This is a serious problem for a substantial portion of RSTS users.

Gentlemen: You have released upon the RSTS community a product with a deficiency in it so glaring that it simply MUST be fixed with all possible speed. It should be a first and foremost priority with all available manpower focused on it. I have great respect for the talents of the developers, I KNOW that a solution can be found and implemented. We NEED IT NOW. Not NEXT FALL. If you didn't hear it in San Diego then you are hearing it here. Lets get on with improving the product, and you can't do that until you FIX what's wrong with it. Anything less is simply unacceptable. Carl B. Marbach

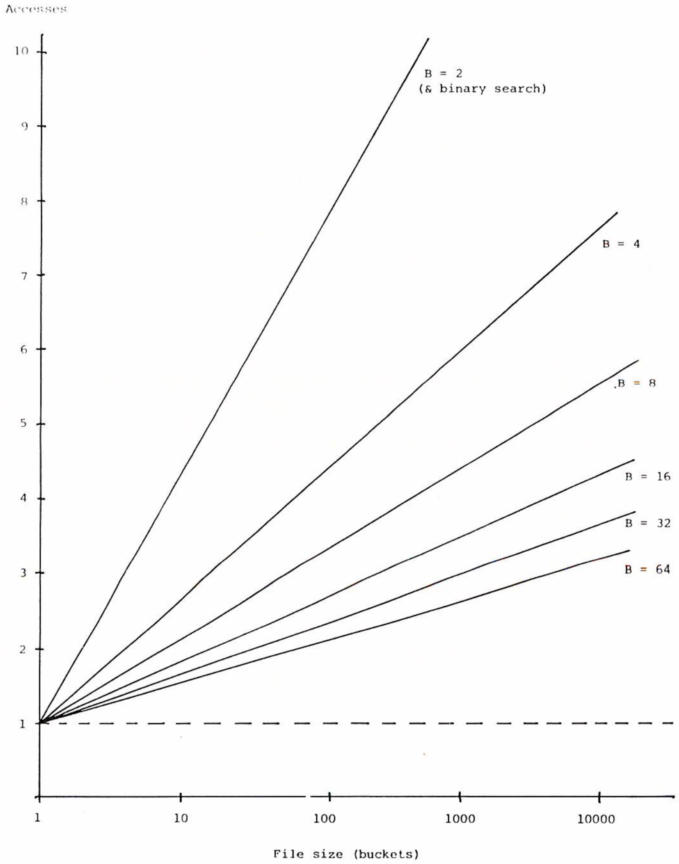


FIGURE 5. ISAM Disk Accesses vs. File Size & Blocking Action

Correction:

RSTS Professional, Vol. 2, No. 3, article "File Structures and Accessing Techniques" by Gregert Johnson (p.18).

Figure 5 was printed incorrectly (see page 23). The correct Figure appears at left. It is reproduced here in the proper size so that readers may paste it in place.

We apologize for the inconvenience.

RSTS PROFESSIONAL*

Pauline Noakes

Editors
R.D. Mallery
Carl B. Marbach
Assistant Editor
Helen Marbach
Controller
Peg Leiby
Subscription Fulfillment
Kathi B. Campione
United Kingdom Representative
Pauline Noakes
RTZ Computer Services Ltd.
P.O. Box 19, 1 Redcliff Street
Bristol, BS997JS
Copy Editors
Marty Grossman
Peg Grossman
Contributors
Scott Banks
Howie Brown
R.C. Cannon
Al Cini
Steven P. Davis
I.F. Dawson
Steven L. Edwards
Dan Esbensen
Jack Gordon
Jeffrey R. Harrow
Dave Kachelmyer
Paul R. Laba
Ted A. Marshall
Bob Meyer
Pauline Noakes
Val Skalabrin
Cartoons
Douglas Benoit
Photographic Consultant
Arthur Rosenberg
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.

RSTS users: You've got it EZ...

Announcing EZSORT V4.0, the first internal sort for Basic Plus 2, Cobol and Dibol.

Easy to Use

EZSORT is linked into your program. No chaining, no funny command files, and no intermediate extraction files.

EZSORT uses any file management system, including: RMS-11K, DMS-500, TOTAL, Record I/O, and User-developed Systems.

EZSORT is flexible: No limit to number or types of keys. Sort by ascending and descending keys at the same time. Maximum recordsize of 16,000 bytes. No maximum filesize.

EZSORT is fast!
Call us for the unbelievable performance details.

Easy to Afford

EZSORT, now available for only $775.00. Multiple CPU licenses and OEM agreements are available.

Easy to Get

Simply call us at (213) 594-9405
or (714) 995-0533.
Delivery is 10 days ARO.
Customer references available.

RSTS users, take it easy, with EZSORT V4.0.

EZSORT from Software Techniques, Inc.

SOFTWARE SPECIALISTS ENGINEERING CONSULTANTS
5242 Katella Avenue, Suite 101, Los Alamitos, CA 90720

LETTERS to the RSTS Pro . . .

Dear Dave and Carl,

On behalf of the PDP-11 Commercial SIG of DECUS UK, I would like to thank you and Al Cini for giving a superb presentation to our User Group on 8th October, 1980 at The Royal Festival Hall in London.

Many of the delegates present at the meeting have asked that their congratulations be passed on to you, together with their thanks for the excellent documentation you provided.

We sincerely hope you will return to London in 1981 to give another seminar to the group, and we thank you most warmly for all the work done by you in the preparation and presentation of the meeting already held.

Many thanks from us all,
Wendy Caine
Pauline Noakes, General Secretary
PDP-11 Commercial SIG, DECUS UK

—

Dear Dave and Carl,

Thanks for the latest 'Professional'. Another bumper issue full of goodies.

It was nice to watch you giving our DECUS UK Commercial SIG seminar. Lots of good, down-to-earth advice and help from people who know because they've done it. May there be more.

If any of your readers know of a need for a couple of months RSTS/E effort in return for a family-of-three holiday in the US next summer, put me at the head of the queue. Come to think maybe you could run camps, and we could all sing campfire songs around the 11/70 console after the bars had closed.

I'll let you know if any of my dreams come true. Meanwhile, back to the sweatshop to cook up another article. It helps if readers say whether they liked them, and why, and what they'd like more of.

Warmest Good Wishes
Steve Holden

Readers please note: the card insert now contains a 'Hot to Cold' article response form. We invite you to rate the articles in the RSTS PROFESSIONAL then simply drop the card in the mail. We'll keep you informed of the results.

—

Dear Dave:

On behalf of all of the SENERUG (Southeastern New England RSTS Users Group) members, I would like to thank you for contributing to the overwhelming success of our first *annual* conference in October. I trust you and Carl had a flight back to Pennsylvania which was smoother than the trip to the airport in Mansfield, Mass.

Enclosed is my (somewhat overdue) subscription for the Professional. I think the magazine can be an extremely effective tool if the "professional" remains part of the spirit as well as the name of the publication. By the way, notably missing from your list of advertiser's in the September 1980 issue was none other than DIGITAL EQUIPMENT CORPORATION. That's right, on page 75 in the classified section (Vol. 2, #3), there is an ad for RSTS/E software consulting in New England, and if you call that number, (617) 895-5090, you will find that it is Digital's New England District Software Services' phone, and comes under the watchful eye of district software manager, Carol Bayley.

Let me close on a serious note by emphasizing that it is in all of our best interests to ensure that we maintain a friendly and professional posture with the management of Digital Equipment Corporation. I look forward to the establishment of a mutually beneficial *free and open* dialogue which will permit the goals of Digital as a Corporation and the needs of its customers and their respective corporations, to become more and more synchronized. Hopefully, once this dialogue is established, we can come up with a suitable mechanism to ensure that the spirit of cooperation and not of antagonism is the true theme of the vendor/customer relationship for the months and years ahead.

Sincerely,
George W. Hallahan, President
George W. Hallahan, Inc.

Thank you for your Delightfully Enlightening Comment, and the ride to the airport.

—

Dear sirs:

Recently I received an invitation to subscribe to your journal. Thank you for the invitation. I will not be able to profit from the offer since I am no longer a user of RSTS. However, there is a way that you might be able to help me.

When I was at Wheaton College we developed a RSTS data management system which we used to build our administrative system. This software subsequently became WISE® (Digital Equipment Corporation). As we used the system we found some weaknesses that could not be fixed without a change in the basic design. Wheaton College was not interested in financing the changes; but since there were a significant number of WISE® users by then who had encountered the same problems, I was encouraged to develop the new system, called WIMS®, myself.

The system is essentially (but not completely) finished. It does indeed have some significant improvements over the data management part of WISE®. Unfortunately I am no longer in a DEC environment and my access to a PDP-11 RSTS system is quite limited, making further development of WIMS® difficult.

I would like to find a reasonable organization that could purchase the rights to WIMS® and complete the steps needed to make WIMS® available. Ideally it would be a software house that could use it as a basis for application packages as well as distributing the WIMS® system itself. The work remaining on it is a small amount of testing (and probably a little debugging), the assembling of the documentation and programs into a marketable package, and promotion.

In addition, there would be customer service, especially guidance in making the best use of such a powerful tool as this, and ongoing support.

If you have contacts with some organizations I could write to, please pass their names and addresses on to me. Thank you for your help and counsel.

Sincerely yours,
Jacques LaFrance, Ph.D.

Dr. LaFrance can be reached at: Wheaton Information Management Systems Consulting (WIMS), 6723 East 66th Place, Tulsa, Oklahoma 74133. Phone: (918) 492-9036.

—

Dear Editors,

We have been running RSTS at our college in Tottenham, North London, since February 1977, when we first took delivery of our PDP-11/34. We have found RSTS to be the ideal operating system for a college environment with its accounting structure, file security and overall 'user-friendly' character. We are a member of the RSTS EduSIG. This is a group of users who have several interests in common. We all have PDP-11's running BASIC Plus and we are all educational establishments. There are some 15 members of the SIG and the numbers seem to increase at every one of our regular meetings. Needless to say, we find that this pooling of our experiences and knowledge leads to a 'collective expertise' from which we all draw benefits — especially the new members who are just starting up. Recently, we have become 'official' by joining DECUS and registering as a Special Interest Group.

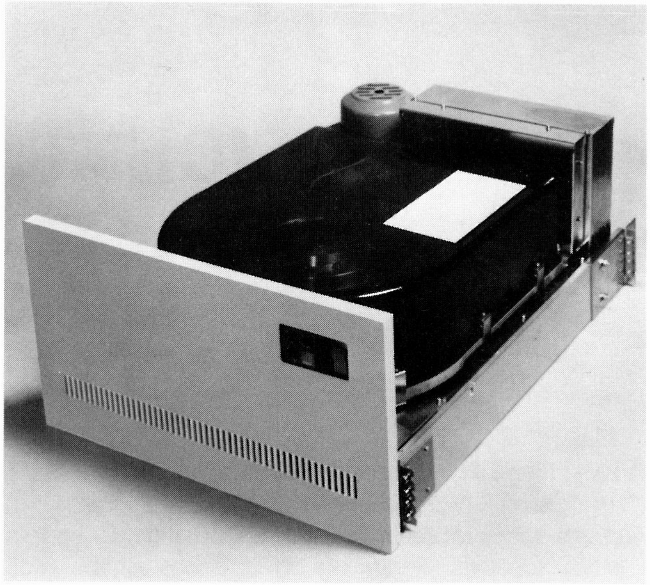
We at Tottenham are a typical member. We use our computer system primarily for teaching, but recently we have introduced word processing and increased the work done for college adminstration by having all student records on disk. All this is done with the minimum of staff — namely, me! I find that my job is a combination of Systems/Applications programmer, DP manager, Technician, Software/Hardware advisary service, etc. Still, the job has an inherent 'omnipotence' when one is 'all things to all men'.

I am turning to the readers of your magazine to help me with some software problems. I'd be grateful for any help or advice on any of them.

1. Apart from the keyboards in the computer suite, we have several others scattered throughout the college. These are connected directly by cables to the distribution panel of the DZ11/DH11's. I can see who is running what by using SYSTAT or VT50PY but in most cases, that isn't enough. I need to be able to view all the interaction that is going on at a particular terminal both from the user and from the system. I want to run a program that, once given a KB number, will then reproduce on my screen, all the input/output taking place at the designated keyboard — just as if I were looking over the user's shoulder. I spoke to the DEC software boffins, and they said it couldn't be done. This seems as good a reason as any for someone out there writing it. Anyone out there willing to accept the challenge?
2. Does anyone know of a documented copy of the source version of the VT52 or VTEDIT TECO macro? Is there a version that uses the DEC ANSI standard escape sequences?
3. We are looking into the possibility of a College Database. Is there a Data Management package written in BASIC Plus? Or RT11? There appears to be something that comes with the RSTS distribution kit called RMS. I have tried wading through the manual, but I'm still in the dark. Is it really as difficult as all that? Is there a simple readable guide to installing it, and writing programs for it?
4. We have several reasonable 'Indirect Command Followers' i.e. programs that read a command file and force them to a keyboard. We have AT and INDIRE of course. We have one written by a student that uses 'command file line numbers' and can jump and loop etc. but what we would like is to:
    a. Have the program detect what the result of a 'forced command' is and make a decision based on it; i.e., if the command file says 'catalogue a disk' and the drive is empty, then it could detect this and branch to a different set of commands.
    b. We can pass parameters into the command strings, but only when the program starts. What we need is an 'interactive' indirect command file follower. When it meets a special command it can prompt and get a string from the keyboard from which it started, then carry on, inserting the string into the appropriate places in the command strings. See what I mean?
5. Does anyone have a program to sort UFD name blockette links so that a CAT will show the files in alphabetical order? (Scott Banks, please note). I have a number of 'general' library disks with many files on few accounts. I've got a program to read the directory and sort the output, but I have to do this every time I load a disk, or after files have been added/deleted.

. . . continued on page 19

Winchester Disk Storage



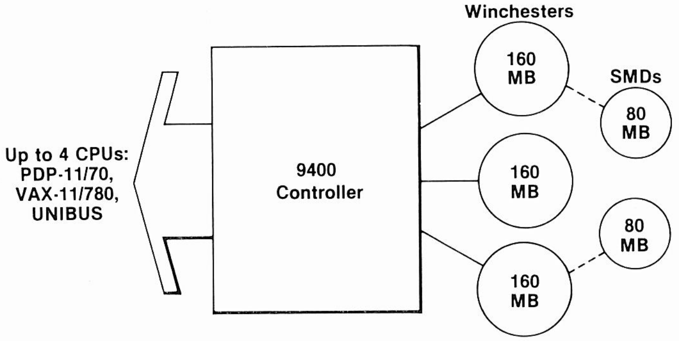
. . . up to 160 Mbytes/spindle, and SMD-compatible too.

Here's Winchester disk storage that delivers more than 10,000 hours of field-tested performance. With fast average access, this 160-Mbyte drive is fully SMD-compatible. That means you can use it with any of our popular 9400 Series system configurations, to provide reliable bulk storage for PDP-11/70, VAX-11/780 or UNIBUS Machines.

Key Features

- Field-tested Winchester technology, with over 10,000 hours MTBF of power-on performance.
- Software transparent in RMOX configurations.
- Operates co-resident with 80-Mbyte SMDs to provide optimal blend of Winchester (non-removable) and SMD (removable) storage media.
- Various 9400 system configurations which interface 4 drives per controller with PDP-11/70, VAX-11/780 and UNIBUS machines.
- Easily interfaceable with Data General's Nova and Eclipse minicomputers, using our single-board 3100 controller.
- Fast access time: 6 ms track-to-track; 27 ms average; 55 ms maximum seek.
- No preventive maintenance required.
- Available with or without power supply.
- Up to 640 Mbytes of rack-mountable storage within a standard 19-inch rack.

Rotational speed	2,964 RPM ± 4%
Recording density	6,475 BPI
Track density	668 TPI
Capacity per track	20,480 Bytes
Transfer rate	1,012 KB/s
Track-to-track access	6 ms
Average access	27 ms
Maximum access	55 ms
Average latency time	10.12 ms
Starting time	40 sec
Stopping time	30 sec
Recording method	MFM
Interface data	NRZ



Winchester non-removable disk storage in co-residence with removable SMD media.

System Industries

In the U.S.: 525 Oakmead Parkway, P. O. Box 9025, Sunnyvale, CA 94086, (408) 732-1650, Telex 346-459. **In Europe:** System Industries U. K., System House, Guildford Road, Woking, Surrey, GU22 7QQ, England, (048 62) 5077, Telex 859124. **Atlanta:** (404) 231-3640 **Boston:** (617) 332-3220 **Chicago:** (312) 642-5456 **Cincinnati:** (513) 771-0075, (513) 874-5503 **Houston:** (713) 497-7224 **Los Angeles:** (213) 557-0384 **New Jersey:** (201) 839-8650 **New York:** (212) 953-0315, (516) 482-6082 **Orange County:** (714) 754-6555 **Rhode Island:** (401) 739-8070 **Washington, DC:** (703) 734-9700 **West Germany:** (06102) 5464/5 **Sweden:** 08-63 62 74

RESIDENT LIBRARIES II
or
Son of Shared Segments

By Al Cini, Computer Methods Corporation

In "A Basic-Plus-2 Programmer's Guide to Resident Libraries" (RSTS Professional, Vol. 2, No. 2), we discussed **Program Logical Address Space** (PLAS), the RSTS/E V7.0 engineering feature which supports — among other things — the RMS Resident Library. By way of review, we recall that "PLAS" is a monitor feature which allows a RSTS/E user a certain measure of control over the **physical** relocation of his or her user job in main memory. The PLAS **directives** by which a user can manage this feature are made available in two ways:

1. PLAS calls are generated transparently by the Taskbuilder when a user creates and links to shared common areas and resident program libraries;

2. A user can execute PLAS directives by writing MACRO programs which issue the PLAS emulator trap (EMT).

In the "Guide to Resident Libraries," we covered Taskbuilder PLAS procedures in detail, politely deferring a discussion of the more flexible (and more complicated) PLAS monitor calls. In this article, we can consider direct programming of PLAS directives in MACRO. We'll discuss each of the PLAS sub-functions, and — to keep things reasonably practical — we'll present a BASIC-PLUS-2 callable set of subprograms which will allow high-level language access to this unusual monitor feature.

The mechanics of Memory Mapping.

Before we proceed, let's take another look at memory mapping under RSTS/E.

We remember that RSTS manages physical computer memory in 1KW segments, while a user job — described by a set of eight mapping registers (APRs) — is organized in 4KW pages (thus the 32KW limit: 8 APRs, each describing up to 4KW of user memory). A 14KW BASIC-PLUS run-time system, as most system managers know, might as well be 16K — a BASIC-PLUS job running under such a run-time system is still limited to 16 KW. (RSTS, of course, will use only 14 KW of physical memory to contain the abbreviated RTS).

As we saw in the previous article, we can use the Taskbuilder to access a resident common area by establishing a physical equivalence between a portion of a BP2 program (defined by a MAP or COM) and a physically distinct region of main memory. This device allows multiple jobs to exchange large volumes of data directly through this "memory window" with very little monitor overhead. Also, if we choose, the Taskbuilder will allow us to access large subroutine libraries through such a "Memory Window" using a much smaller portion of our user tasks. Thus, the RMS Resident Library, which is 23KW in size, "appears" via the **memory resident overlay** device to be only 8KW long to a BP2 program which uses it.

Using PLAS, therefore, a programmer can access (up to) 32KW of memory using (as little as) 4KW of user space. This "user memory extention" PLAS benefit is available from TKB for subroutine libraries, and from the PLAS monitor directives for data.

A complete discussion of these memory mapping principles, and descriptions of each of the PLAS sub-functions, can be found in the **RSTS/E Monitor Directives Manual.** We'll discuss each of the PLAS directives in the context of the BP2 subroutines provided in this article.

The BASIC-PLUS-2 PLAS Subroutines.

The PLAS directives available to MACRO programmers under RSTS/E V7.0 are very straightforward; their implementation as BP2-callable subroutines is a fairly simple exercise. Once a "memory window" has been created using PLAS, however, accessing specific virtual memory addresses using BP2 variables is a somewhat tricky matter. BP2, like most high-level languages, does not provide syntax to "base" program variables on address "pointers." Just the same, BP2 string variables **are** dynamically relocated in memory at run-time and, by manipulating some simple internal descriptors, we can take control of where this relocation takes place. The subroutine which handles this — "FIELD" — directly manipulates the headers of BP2 strings; while this is a simple procedure, please note that a change to BP2 internal data structures in some future release may invalidate this subroutine.

FIELD.

The FIELD subroutine relocates a BP2 character string variable anywhere within user address space. The calling sequence is:

CALL FIELD (ERROR%,BASE.ADDR%,FIELD.LEN%,STR.VAR$)

where:

LEARN BY OBJECTIVES.

Study practical software strategies — uniquely adapted to your environment — right where you live, and on your own terms.

Setting and reaching project objectives demands professional talent. No management plan is complete without an objective skills assessment: the skills you've already got, and the talent you need to build.

Computer Methods Corporation offers common sense training in applied state-of-the-art analysis, design, programming and management methods — on-site for less than you think.*

We've done it for government, industrial and financial organizations with serious commitments to quality software engineering, and we can do it for you.

Use the training to build skills in yourself, or in your project team. If you like, we can work part of your own unique application into a program tailored for you.

* Rates for a 5-day seminar, conducted on-site, begin at $3500 for five students. (Fee is all-inclusive for training conducted within the continental U.S.; overseas pricing available.)

LEARN MORE ABOUT OUR DEC-RELATED CURRICULUM.

BASIC-PLUS(-2) SOFTWARE ENGINEERING STRATEGIES

This 5 day program relates structured coding and design tactics to the unique DEC extended BASIC (RSTS, RSX, VAX) engineering environment.

BASIC-PLUS-2 FOR COBOL PROGRAMMERS

DEC or mainframe programmers who write COBOL can pick up effective BASIC-PLUS-2/VAX BASIC coding habits quickly in this 4 day seminar. The syllabus focuses on functional similarities between the two languages to build new skills on current knowledge.

DEC and RSTS are registered trademarks of Digital Equipment Corporation.

RSTS/E SYSTEMS PROGRAMMING I & II

Two 4 day training programs focus on the effective use of RSTS/E as a software development medium. Covering SYS functions to shared libraries and system directives, these courses are aimed at high-level language programmers and can be offered together or separately.

DISTRIBUTED SYSTEMS PROJECT MANAGEMENT

This 3 day training program addresses the unique problems of building and maintaining distributed applications.

ANALYSIS WORKSHOP

This 2 day session covers interviewing techniques and 'discovery' methods for systems analysts. The focus is on learning to get answers quickly from users, and a video tape analysis of interviewing habits is used to spot problem areas. (Not offered on-site.)

COMPUTER METHODS CORPORATION

SKILLS DEVELOPMENT PROGRAM
P.O. BOX 592, MOORESTOWN, NEW JERSEY 08057
609/778-8440

I want to learn more.

Name ______________________________

Title ______________________________

Company ______________________________

Address ______________________________

____

I'm particularly interested in these training areas:

____

____

____

____

□ Send information on your off-site training calendar.

ERROR%	Returned RSTS/E error code
BASE.ADDR%	Virtual byte base address (**must** be even) at which to relocate the string variable
FIELD.LEN%	Size in bytes of user memory area over which the string variable is to be defined
STR.VAR$	Character string variable to be relocated

Usage notes:

1. Use LSET or RSET to modify STR.VAR$ after a call to FIELD. Standard string assignment will cause BP2 to change the internal descriptor of STR.VAR$ (analogous to usage of the "FIELD" statement).
2. Use CALL; do **not** use CALL BY REF

EXAMPLE:

```
CALL FIELD (ERROR%, 0%, 2%, X$)
PRINT SWAP% (CVT$%(X$))
```

This sequence will display the contents of the first word of user memory.

```
APR%=7%
CALL FIELD (ERROR%, APR%*8192%, 8192%, Y$)
LSET Y$=" "
```

This sequence will "blank out" the 4KW user memory area mapped by APR 7.

The PLAS Directives.

There are six RSTS/E PLAS directives:

ATRFQ	Attach to resident library.
DTRFQ	Detach from resident library.
CRAFQ	Create address window.
ELAFQ	Eliminate address window.
MAPFQ	Map window.
UMPFQ	Unmap window.

Complete functional descriptions of each of these directives can be found on pages 3-163 through 3-190 of your "Monitor Directives Manual." The calling sequences for our corresponding BP2-callable subroutines follow (page references refer to the "Directives Manual").

ATRFQ.

This subroutine allows a BP2 job to "attach" to a shared common area (3-164).

CALL ATRFQ (ERROR%, RESLIB.NAME1%, RESLIB.NAME2%, ACCESS.MODE%, RESLIB.ID%, RESLIB.SIZE%)

where:

ERROR%	Returned error code (p.3-167)
RESLIB.NAME1%	Rad-50 library name to attach (1st 3 chars) (3-165, FIRQB+FQNAM1)
RESLIB.NAME2%	Rad-50 library name to attach (2nd 3 chars)
ACCESS.MODE%	Desired library access: bit 0=1 Read-only access bit 1=1 Read/Write access
RESLIB.ID%	Returned internal resident library code word (save it for later)
RESLIB.SIZE%	Size of attached resident library, in 32.word blocks.

DTRFQ.

This subroutine allows a BP2 job to "detach" an attached resident library (3-174).

CALL DTRFQ (ERROR%, RESLIB.ID%, DETACH.STATUS%)

where:

ERROR%	Returned error code (3-176)
RESLIB.ID%	Internal resident library code word of library to "detach" returned by ATRFQ) (3-175, FIRQB+6)
DETACH.STATUS%	Status of detach operation: bit 14=1 (16384.) Windows were unmapped by this detach.

CRAFQ.

This subroutine creates an address window and (optionally) maps it into a range of addresses within an attached library (3-168).

CALL CRAFQ (ERROR%, RESLIB.ID%, BASE.APR%, WINDOW.SIZE%, MAP.AREA.OFFSET%, MAP.SIZE%, ACCESS.MAP.LENGTH%, WINDOW.STATUS%)

where:

ERROR%	Returned RSTS/E error code (3-173)
RESLIB.ID%	Internal code of attached library
BASE.APR%	The base user APR of the window (1-7)
WINDOW.SIZE%	Desired window size, in 32.word blocks
MAP.AREA.OFFSET%	The offset, in 32.word blocks, from the start of the library where mapping is to begin (3-171, FIRQB+20)

ACCESS.MODE%	Desired access mode: bit 1=1 Write access desired bit 1=0 Read only access bit 7=1 Map the window bit 7=0 Do not map the window
WINDOW.ID%	An internal number identifying the created window (3-172, FIRQB+6)
WINDOW.START%	Starting virtual address of new window (3-172, FIRQB+10)
ACTUAL.MAP.LENGTH%	Length, in 32.word blocks, of area actually mapped (3-172, FIRQB+20)
WINDOW.STATUS%	Map status flags (3-173, FIRQB+22).

ELAFQ.

This subroutine eliminates a specific address window.

CALL ELAFQ (ERROR%, WINDOW.ID%, WINDOW.STATUS%)

where:

ERROR%	Returned RSTS/E error code (3-181)
WINDOW.ID%	Internal identifier of window to eliminate (returned by CRAFQ or MAPFQ)
WINDOW.STATUS%	Status of window after "eliminate" directive (3-180, FIRQB+22)

MAPFQ.

This subroutine will map a created address window onto a specific range of addresses within an attached resident library.

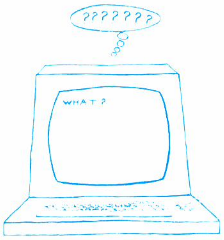
CALL MAPFQ (ERROR%, WINDOW.ID%, RESLIB.ID%, MAP.AREA.OFFSET%, MAP.AREA.SIZE%, MAP.ACCESS.MODE%, MAP.AREA.LENGTH%, MAP.AREA.STATUS%)

where:

ERROR%	Returned RSTS/E error (3-186)
WINDOW.ID%	Internal code for window to "map" (3-183, FIRQB+6)
RESLIB.ID%	Internal code for resident library to "map" (3-183, FIRQB+14)
MAP.AREA.OFFSET%	Offset, in 32.word blocks, from start of resident library where mapping is to begin (3-183, FIRQB+16)
MAP.AREA.SIZE%	Length of area to map in 32.word blocks (3-184, FIRQB+20)
MAP.ACCESS.MODE%	Desired access to mapped area (3-184, FIRQB+22)
MAP.AREA.LENGTH%	Returned size of mapped area (3-185, FIRQB+20)
MAP.AREA.STATUS%	Status of mapped area (3-185, FIRQB+22).

. . . continued on page 66

DEAR RSTS MAN:



DEAR RSTS MAN: After months of effort, I finally convinced the powers at our 11/70 site to purchase version 7.0, with the promise that it would run faster and jump higher than version 6C. Upon completing the SYSGEN the computer ran very slowly; SYSTAT stuttered and bumped pausing often even when it was the only job on the system. Fortunately I have STATs in the monitor and running it showed 1800 characters/sec in and 500 characers/sec out, and no jobs were running! I disabled terminals one at a time and found that when I disabled our Hewlitt Packard Mark Sense reader on DH1 line 6 the problem went away. More experimentation revealed that if I turned the Reader on prior to booting the system everything was all right! Of course Version 6C doesn't have this problem, what great new thing is in Version 7.0 that makes me keep my Mark Sense reader on all the time?

Always On

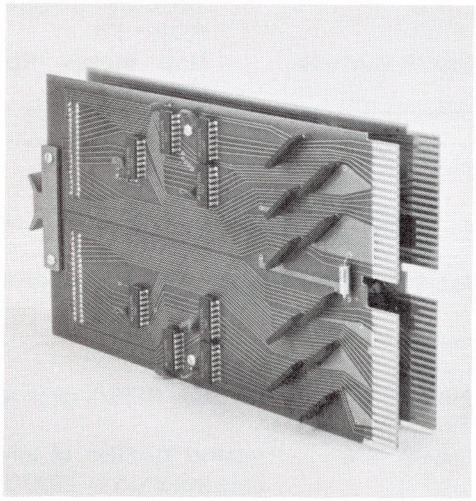
Dear Always On: *The RSTS MAN commends you on your detective work. Large amount of terminal activity can swamp even the best 11/70 bringing the system to almost a complete halt. You are indeed fortunate to have STATS, know how to use them, and disable keyboards (use SPEED 0 rather than DISABLE in UTILTY because once DISABLE'D they can not be reENABLED). I'm not sure why, but one of the''standard features' of V7.0 is BREAK = ↑C. Thats right, type a break key and you'll get cntl C. This feature is dynamite..it can blow up in your face on terminal lines connected to modems who produce breaks or other terminal interfaces who, when their power is turned off, generate a break on the line continuously. I suspect that the hardware of your Mark Sense reader is generating a break when the power is off causing your problem. There is a "feature patch" to disable this new and wonderful "feature". Just apply the patch and your problem will go away; and we can conserve fuel and power again.*

DEAR RSTS MAN: I just added a half-MB bringing my 11/70 to 1½MB. My machine slowed down. Way down. All swapping has ceased, but the machine seems cpu-bound.

Bound Upward

Dear Bound: *I notice that you patched the cache replacement time down to 1 second. Change it to ten seconds. You will note a great change. Then apply the first-fit memory patch for even better results.*

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



WAR and PEACE

In the Beginning there was IBM . . . then came the PDP-8 . . . and the war began . . .

IBM "batch" vs DEC "interactive"

The user who saw the merits of both was little aided by either.

Problem Need: a reliable, high thruput interconnect which both sides find easy to use.

Problem Solution: a PDP-11 front-end processor with Software Results Corporation's HASPBOX™ software.

PEACE at last!

HASPBOX is one of a number of advanced Software Solutions for the VAX and PDP-11 markets from the Data Communications Specialists . . .

SOFTWARE RESULTS CORPORATION

1229 West Third Avenue
Columbus, Ohio 43212 USA
614/421-2094 TWX 810 482 1631

PDP-8, PDP-11 and VAX are Registered Trademarks of Digital Equipment Corp.

USING RSTS/E AS A DEVELOPMENT SYSTEM FOR RT-11 AND RSX-11M PROGRAMS

By Ted A. Marshall and Jack Gordon, Data Processing Design, Inc., Placentia, California

ABSTRACT

RSTS/E has been very useful to us as a development system for programs set-up to run under RT-11 and RSX-11M. This paper describes our experiences developing MACRO-11 programs under these conditions. The limitations and problems with this approach and useful shortcuts are included in this discussion.

INTRODUCTION

Our company has specialized in RSTS/E software (primarily a word processing system called WORD-11) for several years. We have just recently begun to migrate WORD-11 to, and produce new software for, other PDP-11 operating systems, mainly RSX-11M and RT-11. We have a large 11/70 running RSTS/E version 7.0 which has been used for inhouse accounting and outside timesharing in addition to RSTS/E program development. We also have an 11/34 which is shared between RSX-11M, RT-11, and generation of software for our customers (under RSTS/E), and a PDT-150 dedicated to RT-11. Because of the limited capacity of the 11/34 and PDT, and also the expertise of our programming staff with RSTS/E, we are now using the 11/70 RSTS/E system for development of MACRO-11 software for all three operating systems. This paper discusses the usage of RSTS/E as a general PDP-11 MACRO development system.

This paper refers specifically to RSTS/E version 7.0 which emulates version 3 of RT-11 and version 3.1 (approximately) of RSX-11M.

WHY USE RSTS/E?

There are a number of reasons to develop programs under RSTS/E. Perhaps most importantly, RSTS/E is a full time-sharing system with full protection. On a properly set-up RSTS/E system, a macro programmer can edit, assemble, and test programs on the same system running the company's accounting, "canned' programs for layman users, and tasks for other programmers without worrying about crashing the system or destroying another user's program. Also, a program in a run loop will not prevent other users from running programs.

Also, RSTS is designed for time-sharing. Our experience has shown that RSTS/E is generally more efficient than RSX-11M and the larger DEC PDP-11 operating systems for this type of operation.

There are also disadvantages to developing foreign programs on RSTS/E. RSTS/E is not a real-time system. A job cannot have direct access to device registers or be directly connected to interrupt vectors. Neither can it run in kernel or supervisor mode.

In addition, the emulation of monitor calls provided by RSTS/E is not complete. For instance, the emulation of RT-11 monitor calls includes only a subset of those on the single-job monitor.

However, the emulation is good enough that RT-11 and RSX-11M utility programs (including MACRO, the linker or task-builder, and so on) run on RSTS/E without modification (except for re-task-building of RSX-11M programs). This excludes programs such as PIP that do very system specific operations. At any rate, a program can at least be created, assembled and linked on RSTS/E, no matter what it does.

THE SYSTEMS

RSTS/E is a fully protected, non-multi-tasking, application independent, "friendly" time-sharing system. The monitor and all jobs are protected from the actions of other jobs. There is no way for a normal job to read from or write to the memory of another job or the monitor, the device registers or the vectors.

RSTS/E is not multi-tasking as is RSX-11M. Each logged-in user is connected to one job and each job can only execute one program at a time. There are also detached jobs that are not connected to any terminal, but each of these is basically independent.

RSTS/E is generally acknowledged to be the friendliest of the DEC PDP-11 operating systems. This can be understood from the fact that it evolved from a small multi-user BASIC-only system for educational users. But it is also a full multi-language time-sharing system usable on machines ranging from an 11/34 with 96KW to an 11/70 with 4 megabytes. And for general program development and application usage, our experience has shown RSTS/E to be generally more efficient than RSX-11M and the larger DEC PDP-11 operating systems.

RSX-11M is an application tuned, multi-terminal, real-time, multi-tasking system. It is often described as an "application engine". That is, RSX-11M is often used to run a single application and can be highly tuned for that application. RSX-11M is a full multi-tasking system that can be run on an 11/23 or larger machine with or without memory-management. A terminal may be connected to any number of programs (tasks). Tasks can freely spawn other tasks. But RSX-11M does not have a full time-sharing scheduler. It is strictly priority driven.

RT-11 is a single-console, unprotected, single job or foreground-background, real-time system that can run on any PDP-11. Although multiple terminals can be supported, only one can directly interact with the monitor command processor. The others can only be used by running programs.

RT-11 can be generated for a single job, foreground and background jobs, or with the latest version: foreground, background, and six 'system' jobs (reserved for the error logger, queue manager, and such). The scheduler is strictly priority driven with each job having a fixed priority. Also, unlike the other two systems, all jobs are permanently core-resident.

PROGRAM CORE IMAGES

There are several important factors in looking at the compatability between these three systems. The most important of these are program layout in main memory and monitor calling conventions.

The main memory layout of a RSTS/E job is shown in figure 1. Because RSTS/E is a mapped system, the monitor, device registers and vectors are not in the job's addressing space. Thus, the program can use the full 32KW addressing space. Beginning at the top of 32K and allocated down from there is the segment known as the Run-Time-System. This is a sharable, generally read-only segment. Its common uses are for language processors, run-time support libraries and shared programs. It also is used for the RSX-11M and RT-11 emulation code. This segment is required in all jobs, except that under RSTS/E version 7.0, if the job is running under the RSX-11M Run-Time-System, the Run-Time-System can be made to "disappear" with the monitor doing the emulation.

Beginning at location zero and growing up from there is the user segment. This is a non-sharable read-write segment that contains the job's context. Most of the lowest 1000 (octal) bytes are reserved for use by the monitor for servicing the job. This segment is used for non-sharable programs (including RSX-11M and RT-11 programs), variables, I/O buffers, and the program stack. The program can dynamically change the size of this segment.

The space (if any) between these two segments is inaccessible except that addressing windows may be created to access portions of segments called Resident Libraries. These are very similar to those under the other two operating systems.

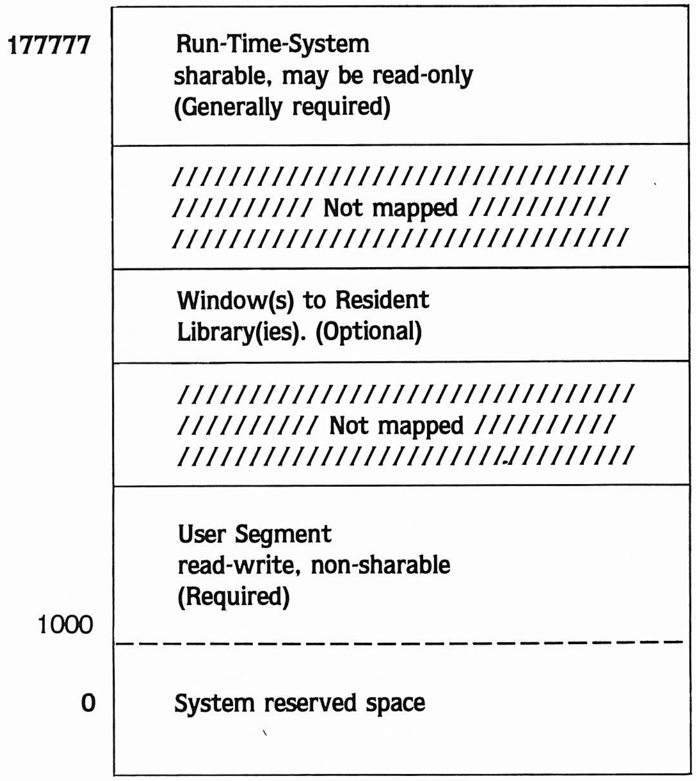


Figure 1 - RSTS/E Program Layout

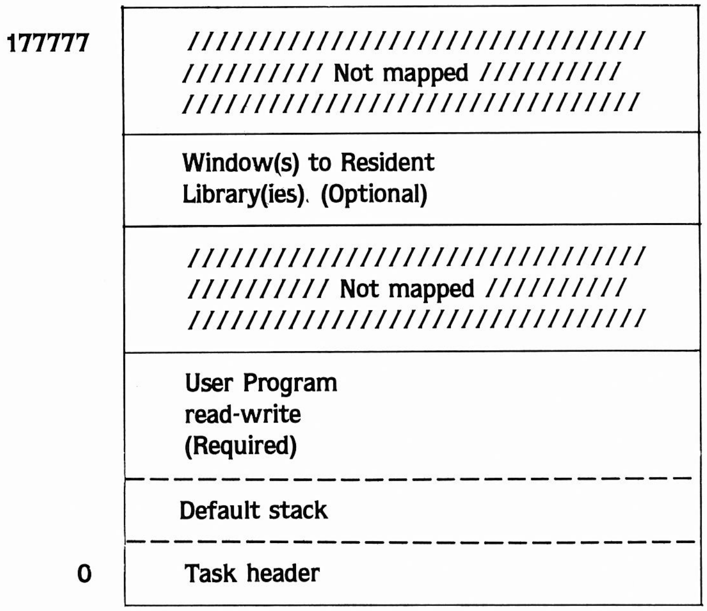


Figure 2 - Mapped RSX-11M Program Layout

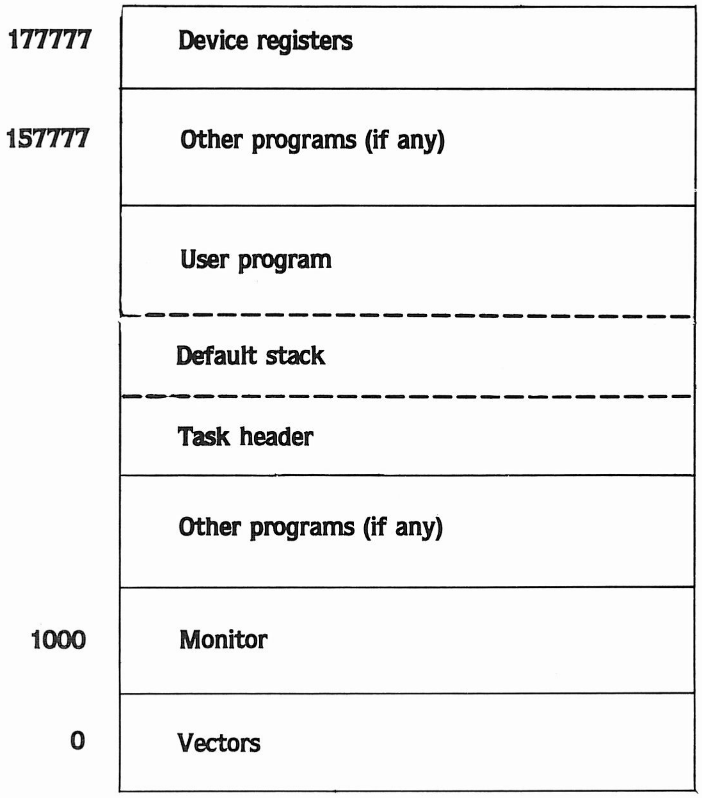


Figure 3 - Unmapped RSX-11M

The memory layout of a task on a mapped RSX-11M system is shown in figure 2. The program begins at location zero and grows upward in memory. At the very bottom is the task header which is used by the monitor to keep task related information and to communicate with the task. The remainder is freely available to the program. Figure 3 shows the layout of a task on an unmapped RSX-11M system. The difference is that the base of the task is not at location zero and the task's memory is surrounded by the memory for other tasks and the monitor.

The major differences between a task image under mapped RSX-11M and a program image under RSTS/E is that under RSTS/E the system reserved space at address zero is larger and the highest available address is lower (because of the Run-Time-System and any constraints on the user's job size made by RSTS/E). Thus an RSX-11M task can be fit into a RSTS/E job space by simply re-task-building for a different program base (however, an RSX-11M program may not fit because of the increased space used for overhead). The RSTS/E version of SYSLIB.OLB, the system library file, takes care of any references to locations in the header. Variables required by the emulator are fit into face locations within the RSTS low-core system area.

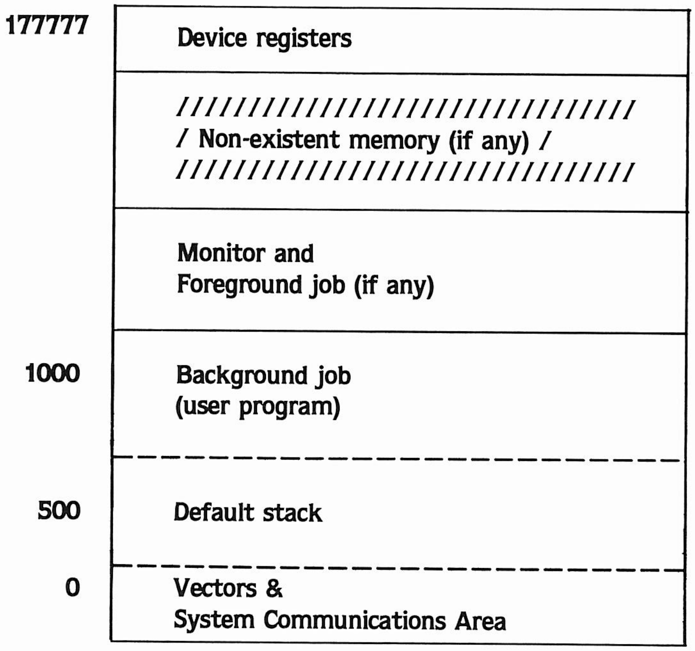


Figure 4 - RT-11 Program Layout

The memory layout of an RT-11 background job is shown in figure 4. The lowest 500 (octal) bytes contain the system vectors and the system communications area. The systems communications area is used to pass information between the monitor and user job. The monitor, device drivers and the forground and systems jobs are placed above the background job's memory.

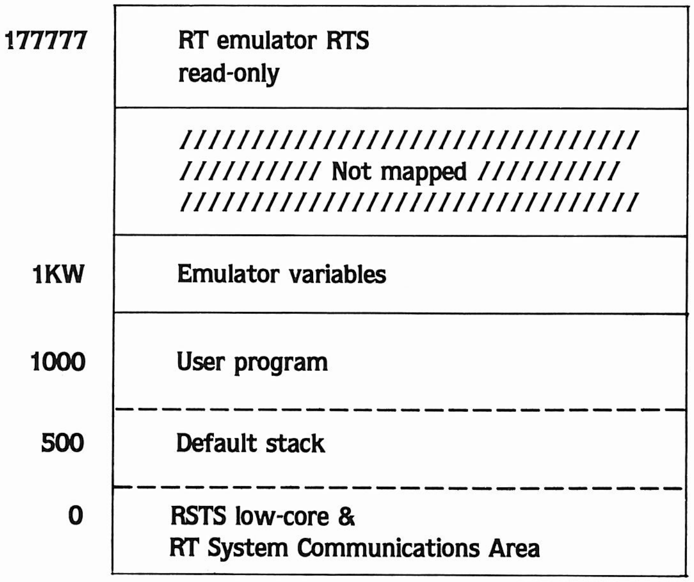


Figure 5 - RT Emulator Program Layout

The major conflict between this and the RSTS/E layout is the use of locations 0 through 777 (octal). Fortunately, (or perhaps by design) the system communications area fits into a free portion of RSTS/E lowcore. Also, the area from 500 through 777 can be relocated while the RT-11 emulator is in control. Therefore, this area is available as in real RT-11. The emulator requires about 1KW of memory for variables and the moved portion of RSTS/E low-core. This is placed immediately above the top of the user's program. Therefore, it is vital that the program allocates all space that is needed or uses the SETTOP monitor call to allocate additional space. Figure 5 shows the emulator job layout.

MONITOR CALLS

Monitor calls to RSTS/E are done through the Emulator Trap (EMT) instruction. The EMT instruction contains a value in its low byte, ranging from 0 through 377 (octal). These values specify the major function of the monitor call for RSTS/E. Information is passed and returned with the calls through two areas in low core and other core areas pointed to by portions of these. These are called the File Request-Block, or FIRQB (pronounced Furk-bee), and the Transfer Control-Block, or XRB. All registers and condition codes are unchanged over these monitor calls.

The exact action of an EMT depends on several factors. If the EMT is located in the Run-Time-System, it is executed by the monitor. An RTS can be declared to have a "prefix" EMT which controls the actions of an EMT located in the user segment. If no prefix is defined, EMT's in the user segment are executed by the monitor unless it is an unused code in which case it is passed to the RTS. If a prefix is defined but not used, all user segment EMT's are passed to the RTS. If an EMT is prefixed by the prefix EMT, it is processed as if there where no prefix defined. The prefix EMT otherwise has no effect.

In RSX-11M, all monitor calls are done with the EMT 377 instruction. Data is passed and returned through the stack, locations in the task header, and the "C" condition code. Because EMT 377 is not a valid RSTS/E call and there is no prefix EMT for the RSX-11M emulator, this call is passed directly to the emulator RTS when running under RSTS/E. Most calls directly to the RSX-11M monitor are emulated, but calls passed to an RSX-11M ancillary control processor (ACP) for file operations are not emulated. However, these calls are generally done through the file control system (FCS) which is linked into the task. The RSTS/E system library file contains a version of FCS that uses RSTS/E calls. Thus, as long as the task does not call an ACP directly and does not make certain RSX-11M calls, primarily for real-time processing, it will run under RSTS/E only requiring a task build with the proper SYSLIB.OLB file.

Monitor calls in RT-11 use EMT's with varying codes like RSTS/E. Data is passed using R0 to point to the transfer area. This is achieved by using the emulator with a prefix EMT code of 377 (which is not used in RT-11). This allows all other EMT's in the user segment to be passed to the emulator. Thus, providing that certain monitor calls (primarily real-time and foreground-background) and direct access to the device registers and the monitor are avoided, RT-11 background programs will run under the emulator without modification. RSTS/E does not attempt to emulate the environment of foreground programs.

Information on exactly what calls are emulated can be found in the RSTS/E System Directives Manual, # AA-D748A-TC, Chapter 5, for the . . . . RSX-11M emulator and in the RSTS/E V7.0 Release Notes, # AA-5246C-TC, ... — Chapter 8, sequence 22.1.1, for the RT-11 emulator.

MISCELLANEOUS INFORMATION

A major restriction of RSTS/E is that all input/output operations are synchronous. That is, the job is suspended until the operation is completed. Thus, RSX-11M and RT-11 "read or write without wait" operations are emulated as "read or write with wait" and the "wait for I/O complete" produces no operation. Thus, these operations will generally work correctly.

The disk file system layouts are different on all three systems. Thus, disk packs are not transportable between them without the use of special utility programs. There is a RSTS/E utility, called FIT, that will read or write RT-11 disks. However, there are no programs for reading or writing Files-11 disks. Magnetic tape, DECtape and DECnet are directly compatible between RSTS/E and the other two.

The monitor command languages are significantly different between RSTS/E and RSX-11M. This is mainly because of the different underlying concepts of the systems. The command languages are somewhat similar between RSTS/E and RT-11, because both are modeled on TOPS-10. Also, because the command processors under RSTS/E are built into the Run-Time-Systems, the commands for the emulator have been made similar. However, RSTS/E does not support the DEC Command Language. In all cases, some re-training is required.

TABLE 1. RT-11 Emulator Variables

Offset	Size	Label	Description
0	2	SPCBLK	Special block number for read/write.
0	2	PPN	Account number.
2	2	RWMSBS	Most significant byte of block number for read/write.
2	2	PROT	Protection code.
4	2	MODE	Open/create mode.
6	2	CLUSTP	Cluster size for create.
10	2	POSITN	Position for create.
12	1	CRMSBS	Most significant byte of file size for create.
13	1	NOTRNS	Flag: Don't translate logical device name.
66	2	NOCTLC	Flag: ignore ^C.
70	2	USERCC	^C trap.
72	2	FPADDR	FPP traps.
74	2	TRAPAD	Traps through vectors 4 and 10.

As noted earlier the RT-11 emulator keeps about 1KW of variables just above the user program. Location 54(8) contains the address of the first of these variables. Table 1 shows the most interesting of these. The addresses listed are relative to the contents of 54. Locations SPCBLK through NOTRNS are loaded with values for RSTS/E file operations allowing these to be used with the RT-11 monitor calls.

These are all one-shot. Locations USERCC through TRAPAD contain the addresses of trap routines for the user program.

A program should be assembled and linked or task-built using the programs from the target system (i.e. use MACRO.SAV to assemble an RT-11 program, MAC.TSK for an RSX-11M program, and so on). Copies of these programs are provided with the RSTS/E distribution but some of these have been modified or have had some options disabled. However, the assembler, linker and librarian from version 4.0 of RT-11 and the assembler and librarian (but not the task-builder) from version 3.2 of RSX-11M are fully operational under the emulators. In fact, the RT-11 version 4 assembler is significantly faster than the one provided with RSTS/E.

CONCLUDING REMARKS

The emulators on RSTS/E are useful in allowing program development for RSX-11M and RT-11 on a medium-to-large timesharing system without worry of crashing the system or otherwise interfering with the other users. The program can be fully developed on RSTS/E, leaving only the final testing (or all testing in the case of a few programs) under the target system. This is particularly helpful where there is a large general usage system and a few small target systems.

REFERENCES

. Digital Equipment Corporation, RSTS/E V7.0 . . . Release Notes, DEC # AA-5246C-TC, September, 1979.

. Digital Equipment Corporation, RSTS/E System . . . Directives Manual, DEC # AA-D748A-TC, May 1979.

. Digital Equipment Corporation, RSTS/E V7.0 . . . Mon/Init MCRF (source microfiche), DEC # . . . AH-2658G-SC, 1979.

. Digital Equipment Corporation, RSTS/E V7.0 RSX . . . RTS, Lib MCRF (source microfiche), DEC # . . . AH-2659G-SC, 1979.

. Digital Equipment Corporation, RSTS/E V7.0 RT . . . RTS, Lib MCRF (source microfiche), DEC # . . . AH-D089G-SC, 1979.

. Digital Equipment Corporation, RT-11 Software . . . Support Manual, DEC # AA-H379A-TC, March 1980.

. Digital Equipment Corporation, RSX-11M System . . . Logic Manual, DEC # AA-5579A-TC, November 1978.

LETTERS to the RSTS Pro . . .

. . . continued from page 6

Finally, when are you going to bring RSTS Professional out more frequently? We need all the help that we can get.

Yours sincerely,
Dennis M. Ellis
Dept. of Environmental Health & Science
Tottenham College of Technology

Can any of our readers help Mr. Ellis?
The RSTS Professional will have to remain a quarterly until we can get all the help we need.

—

Dave—

Your participation in the impromptu session where you pinchit, and the "Carl & Dave Show" were the highlights of the San Diego DECUS.

Thanks so much,
Dirk Fitzgerald
Community Computer Services, Inc.
Auburn, New York

—

Dave & Carl—

Just a follow-up concerning Figure 5 in my article. (*File Structures and Accessing Techniques*, Vol. 2, #3, p.18). The true Figure 5 is enclosed. Perhaps it's worth printing in the next issue.

Really enjoyed your contributions to our Symposium. It was great to have all those high quality people up at the front of the room. Hope to see you again soon.

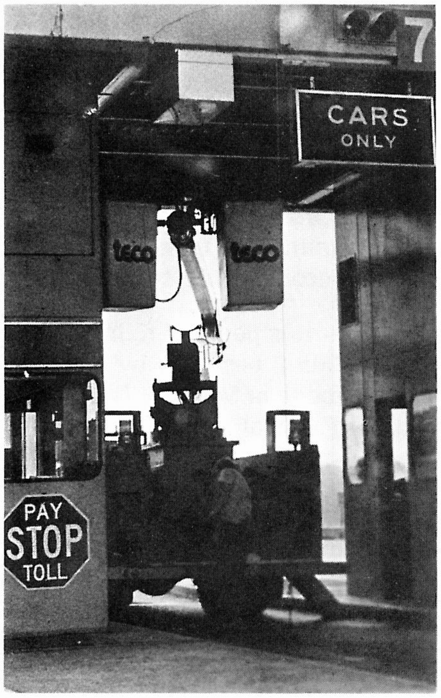
Greg Johnson
IIRI International, Inc.
Providence, R.I.

We apologize for the printer's error. In Greg's article, what is shown to be Figure 5 (appearing on page 23) is really Figure 7 repeated. You'll find the true Figure 5 on page 4 of this issue. It is printed in the proper size so that readers may paste it in place.

—

DO YOU REMEMBER THIS?

(Photo contest(?), RSTS Professional Vol. 2, #3, p.75.)



[No one has gotten this right yet!]

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

A RSTS Professional T-shirt is on its way to the following readers who had nerve enough to answer September, 1980's photo contest.

Dear RSTS Professional:

Re: How TECO? Why TECO, p. 75, Sept. 1980.

How? They drove there in their truck.

Why? Something's wrong with the toll booth.

You see, these obviously must be the kind employees of the Tampa Electric Company (TECO!) at the entrance to the Sunshine Skyway bridge — you know, the one that the ship ran into. I must admit, however, that paying toll for TECO is a novel, but rotten, idea.

Chris Thomas
Rose-Hulman Institute of Technology
Terre Haute, IN

P.S.: If Mark gets a T-shirt, I have to have one too!

Did Mark get a shirt after the mess he made?! If that's the case, you deserve one also. (Besides, yours was the first incorrect answer we received.)
See RSTS Professional, Vol. 2, #3, p.33.

—

Gentlemen:

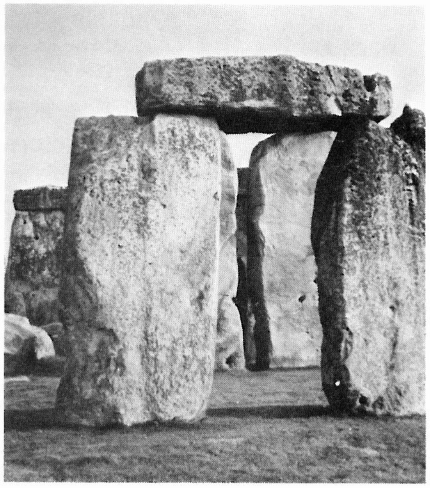
The photo captioned "How TECO? Why TECO? A prize if you tell us?" is a picture of the Tampa Electric Company working on a toll booth.

A prize?

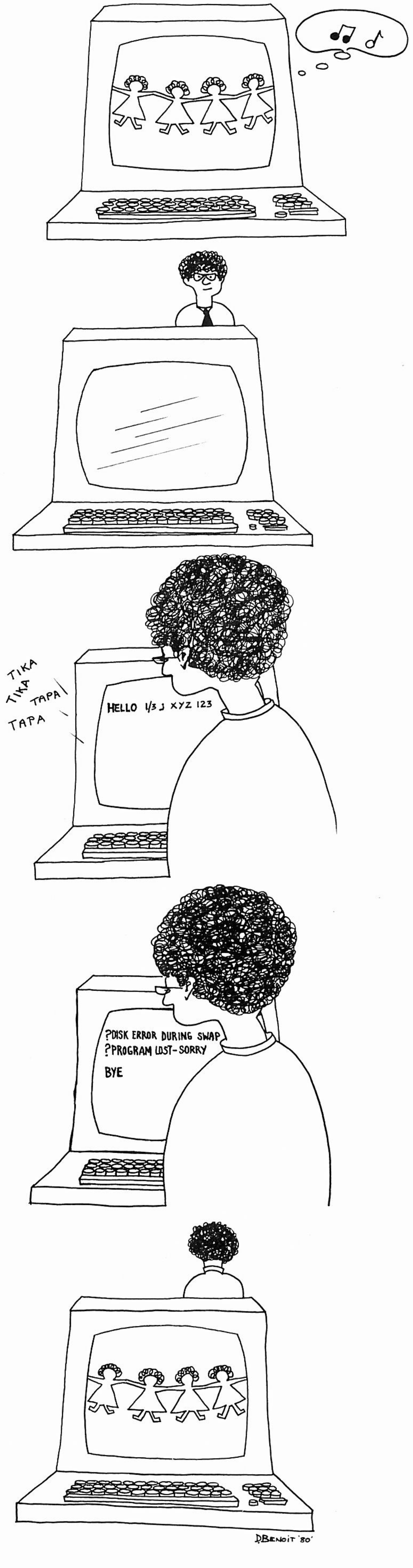
Sincerely, James Allan
Grinnell College, Grinnell, IA

Wrong! But . . . okay, James, a prize. But, we'd like to know how you could answer (?) and be sincere at the same time.

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



WAS THIS THE FIRST COMPUTER?



Basic-Plus 2 And MACRO

By Steven P. Davis and Steven Edwards, Software Techniques, Inc.

1.0 Abstract

This presentation will center on the use of assembly language subroutines/subprograms while programming in Basic-Plus 2. From the concept of threaded code to the use of threaded routines in the subroutines themselves. Also discussed will be typical applications as well as applications the speakers have done themselves. In addition, the structure of the OTS workspace will be explained and how the user may take advantage of it. This document addresses itself to the advantages and methods of interfacing Basic-Plus 2 with MACRO. Although it deals specifically with the RSTS/E operating system, it should be useful to any Basic-Plus 2 programmer.

2.0 Disclaimer

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

3.0 Why Basic-Plus 2 and MACRO

The easiest answer would be to say, "Because it's there?'. However, this has scarcely justified anything but mountain climbing.

Writing programs in a high level language like Basic-Plus 2 allows rapid development of applications, while writing programs in an assembly level language like MACRO allows the utmost in efficiency and flexibility. The combination of Basic-Plus 2 and MACRO can be a synergistic relationship whereby the resulting program can be developed in a reasonable time frame and still be very efficient and flexible.

3.1 Enhanced Subroutine Usage

Breaking a single large Basic-Plus 2 program into a series of smaller Basic-Plus 2 subroutines promotes the coding of small, easy to manage routines. The programmer can then analyze each routine to determine which routines would benefit from being coded in MACRO (and also which are within his ability).

3.2 Efficiency & Overhead

This of course does not suggest that Basic-Plus 2 is inefficient. What it does imply (hopefully) is that many things may be coded in MACRO that do not incur the wrath of the Basic-Plus 2 OTS. These subroutines may be tailored to operate as quickly as possible with as little overhead as possible.

3.3 Informational Purposes

Many functions of the operating system are only available on the assembler level. This includes internal monitor information that may be obtained through MACRO-level subroutines that might otherwise require privilege or considerable overhead if written in Basic-Plus 2.

3.4 Functionality

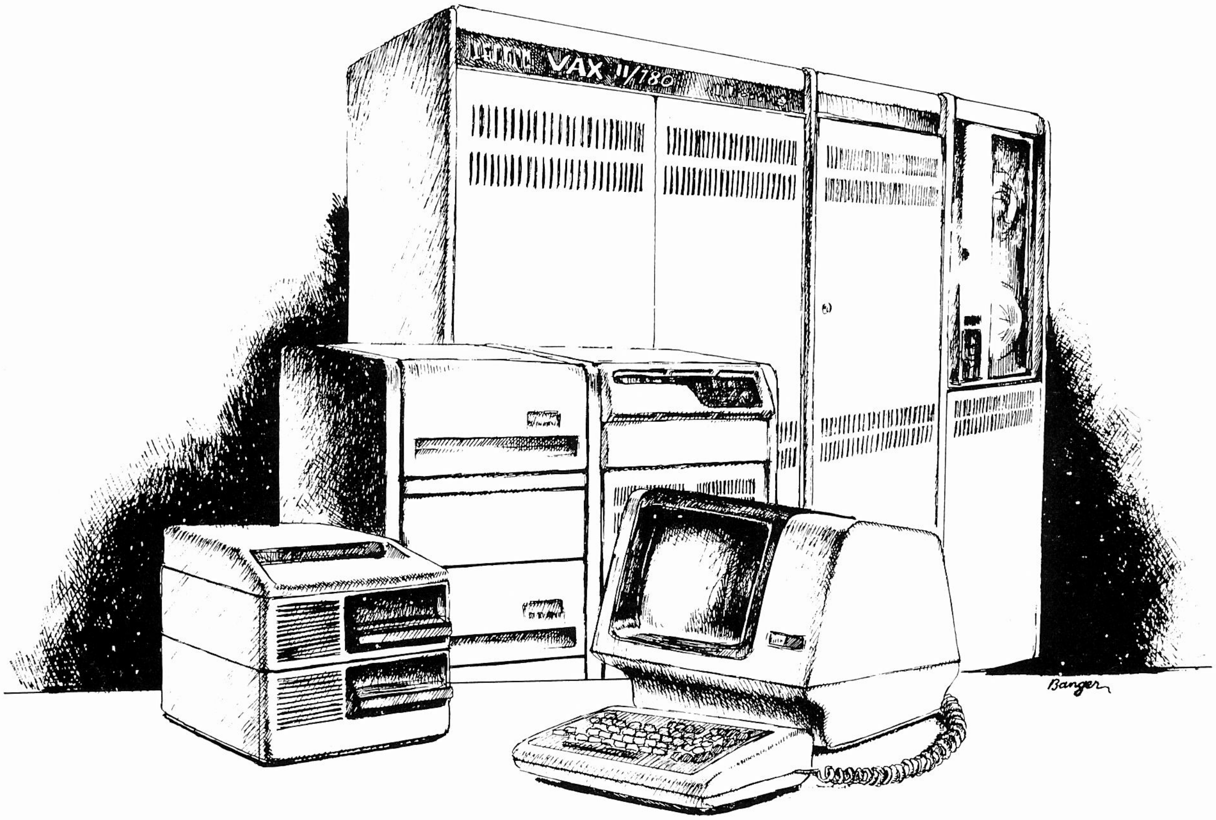
MACRO has several features that can contribute to efficient and flexible programming. MACRO offers an extension to the powerful features of Basic-Plus 2 that is uncompromised. Essentially any feature of the operating system can be employed and taken full advantage of even though that feature may not be readily available within Basic-Plus 2 itself.

With the release of the Executive Directives Manual it has become evident that many useful functions of RSTS/E are not implemented within the compiler. As people explore and find applications for these functions, the use of "hybrid programming" in commercial software developments should become quite common.

3.4.1 Variable Initialization — It is possible to initialize a set of variables in a Basic-Plus 2 program by using an overlaying program section coded in MACRO. This allows for easy manipulation and initialization of selected variables without re-compilation.

3.4.2 Variable Number of Arguments — Unlike a Basic-Plus 2 subprogram, a fixed number of arguments is not required. When control is passed to the MACRO subroutine, the number of arguments is passed in the first byte of the argument list. Basic-Plus 2 does not require that the subroutine use all passed arguments. This allows you to pass a variable number of arguments in the CALL statement.

WORD PROCESSING



PDP-11 AND VAX

MULTI-USER

Written entirely in MACRO-11 machine language, you can have up to 90 people using the full power of WS-11 word processing simultaneously without impairing the performance of data processing.

EASY TO USE

Color coded commands are clearly labeled on the keyboard. Menus display all choices on the screen. No technical language or cryptic commands to remember. The screen immediately shows all changes you make to your document. No surprises.

FROM PDT TO VAX

WS-11 can be used on the full PDP-11 family using its own operating system, RSTS/E, CTS-500, RSX-11M or VAX/VMS. Transition from one system to another can be made without any change to your documents.

POWERFUL FEATURES

Just a sample: Right margin justification, Centering, Search and replace, Libraries of stored text, Simultaneous editing and printing, Support for a wide range of printers including Xerox, NEC, Qume, Sanders and DEC.

$2000-$3500

WS-11

NDS

NORTHWEST DIGITAL SALES
668 S. Sunset Avenue
West Covina, CA 91790
(213) 960-2895

This flexibility permits the use of optional parameters in the subroutine. One example would be a subroutine that sets terminal characteristics (i.e. ESC SEQ, PRIVATE DELIMITER, etc.) and optionally sets the terminal width to a specified value.

```
     CALL SETTTY(ERR%)

          - or -

     CALL SETTTY(ERR%, WIDTH%)
```

FIGURE 3-1. Example of Variable Argument CALL

When the WIDTH% argument is specified, the MACRO subprogram can detect its presence and act accordingly.

3.4.3 Multiple Entry Points — MACRO subroutines facilitate the use of multiple entry points. This allows several similar subroutines to exist in a single object module, sharing both code and data.

An example would be the already mentioned SETTY subroutine. It is desirable to have the ability to reset the terminal characteristics to those set prior to the SETTY call. An additional entry point, CLRTTY, can be added to reset the terminal. When SETTTY is called, it saves the current settings in a common data area. The CLRTTY call is then executed to restore the terminal to its previous state using the saved information.

3.4.4 Extended Functionality Of Basic-Plus 2 — Two examples of extending the functionality of Basic-Plus 2 are modifications to the STP$ thread (STOP statement) to allow GOTO line number and RESUME line number, and modifications to the LIN$ thread (module used to set the pointer to the current line number) to do timing/tracing on a line-by-line basis.

3.4.5 Access to the Basic-Plus 2 OTS Workspace – Through MACRO subroutines you can interface with the Basic-Plus 2 OTS workspace.

3.4.6 Resident Library Support — Resident library support is currently not an available feature within the compiler itself, however, through MACRO subroutines you have access to the complete set of monitor directives that allow resident library mapping, unmapping, and window control.

A typical application might be the use of a resident library to transfer data between two interactive tasks. For example, a single task might control all access to a database, while an additional task handles all terminal handling. The terminal handler would make a request to the database manager for a specific block of data using the SEND/RECEIVE capabilities of the RSTS/E monitor. It is not desirable to transfer the data to the terminal task with a SEND/RECEIVE transfer, or even through an intermediate disk file. The data may be transferred through a read/write resident library that both the tasks are mapped to. This eliminates the overhead of SEND/RECEIVE and disk file handling.

3.4.7 Full RMS functionality — Although the authors have not had the time or the need to explore the benefits of this aspect, the RMS manuals allude to functions which are only available to the MACRO programmer.

4.0 How Does Basic-Plus 2 Work?

Although this document does not deal directly with the operation of Basic-Plus 2 itself, it is necessary to know a limited amount about that operation to make full use of its interface with MACRO.

The Basic-Plus 2 compiler, (a large part of which is written in Basic-Plus 2), accepts a program source file and compiles the program statements into threaded code, performing some optimization during the process.

4.1 Threaded Code

Basic-Plus 2 produces what is known as threaded code. It is the structure of the code that prompts the name. A list of the threaded routine addresses and the arguments used by those threaded routines are created in a sequential order by the compiler. Then an indirect jump auto-increment call (JMP @(R4)+) is used to start the execution of the threaded routine. A "threaded routine" is a module that performs some function on an argument list.

Threaded code differs from in-line subroutine code conceptually in that threaded code transfers control from one program module to another without returning to the 'mainline' program.

Threaded code provides a more space efficient means of transfer between routines. Below in Figure 4-1, is a short program that demonstrates the structure of threaded code. A threaded routine, ADD, is used to add two numbers together. It should be noted that the"JMP' instruction requires only one word for address specification, whereas a "JSR' subroutine call would normally occupy two words of storage. Also, execution of the 'JMP' instruction requires less than half the time of the"JSR' instruction. This also allows for use of stack (SP) for argument transfer without the need to correct for subroutine calls.

4.2 The Compiler Threads

By using the /MAC option of the compiler, one can see how this thread list is built, From this it can seen that interfacing your MACRO subroutines with the Basic-Plus 2 object threads is just a matter of determining what arguments it requires. This makes available to the programmer any function required to successfully interface with Basic-Plus 2 on even the most sophisicated level.

Figure 4-2 is a simple string assignment Basic-Plus 2 program. Using the COM/MAC option of the compiler, you can see how the code is generated for this program.

Figure 4-3 is a segment of the output generated by the compiler. It demonstrates how the thread list is assembled for execution.

```
SUM:      .WORD   0                ;RESULT OF ADDITION

START:    MOV     #THREAD,R4       ;SET START ADDRESS
          JMP     @(R4)+           ;START THE THREAD

THREAD:   .WORD   ADD              ;ROUTINE TO EXECUTE
          .WORD   10.              ;FIRST ARGUMENT
          .WORD   15.              ;SECOND ARGUMENT
          .WORD   SUM              ;DESTINATION
          .WORD   EXIT             ;AND EXIT

ADD:      MOV     (R4)+,@2(R4)     ;ADDRESS OF DESTINATION
          ADD     (R4)+,@(R4)+     ;AND PLACE THE SUM
          JMP     @(R4)+           ;AND CONTINUE

EXIT:     HALT                     ;AND STOP (KLUNK)

.END      START
```

FIGURE 4-1. Example of Threaded Code

```
1000      TEST$ = "STR" + "ING"
1010      END
```

FIGURE 4-2. Simple String Assignment Program

```
L1000:   LIN$      ,1000           ;  #1000
         MOS$MM    ,$PDATA+16      ;  "STRING"
                    $STRNG+0       ;  TEST$

L1010:   LIN$      ,1010           ;  #1010
         END$
```

FIGURE 4-3. Generated Thread List

5.0 Subprogram Calling Conventions And Linkage

The Basic-Plus 2 program transfers control to the MACRO subprogram via the CALL statement. The Basic-Plus 2 compiler generates the code needed to establish the argument list at run-time as well as the code to transfer control from the thread list to the program.

Note: The Basic-Plus 2 CALL statement will allow you to pass up to 25 arguments to a MACRO subroutine, whereas the Basic-Plus 2 SUB statement will allow only 8 arguments.

5.1 Argument List Structure

Basic-Plus 2 uses an argument list to pass arguments to MACRO subprograms. When control is passed to the MACRO subprogram, register 5 (R5) points to the address of the argument list, as shown in Figure 5-1.

5.2 CALL Statements

There are (currently) two forms of the CALL statement: CALL() and CALL by REF(). The two statements pass integer, real (single-precision), and double value (double-precision) arguments in the same way. They differ in the manner they pass string and array arguments. The R5 argument list passes data formats as follows:

Integer The R5 argument list contains the address of the word holding the integer value.

Real The R5 argument list contains the address of the high-order word for the double-precision value.

String When CALL is used, the R5 argument list contains the address of a two word string header. The first word of this header is the starting address of the first byte of the string. The following word is the string's length in bytes.

When a CALL by REF statement is used, the R5 argument list contains the address of the first byte of the string. The string's length is not available with this form of the CALL statement.

Array When CALL is used, the R5 argument list contains the address of the second word of the array header. The array header contains the subscript information and the address of the first array element.

When CALL by REF is used, the R5 argument list contains the address of the first element in the array. Array header information is not available.

NOTE: Additional information on the data formats contained within the argument list can be found in Appendix D of the Basic-Plus 2 User's Guide.

```
R5  ______________________________
                                  |
       -------------------------
      | Undefined   | # of Args |
       -------------------------
      | Address of Argument #1  |
       -------------------------
      | Address of Argument #2  |
       -------------------------
      |            .            |
      |            .            |
      |            .            |
       -------------------------
      | Address of Argument #N  |
       -------------------------
```

FIGURE 5-1. Argument List Format

5.3 Accessing The Argument List

What follows are a series of code examples to illustrate common methods of accessing the argument list after Basic-Plus 2 has transferred control to the subroutine.

Refer to Figure 5-2 for the source Basic-Plus 2 statements that the examples make use of.

It is often advantageous to check the number of arguments passed to you by the caller. Figure 5-3 gives an example of how to perform this check. Argument checking is the same for the CALL() and the CALL by ref() statements.

Integer access is the simplest of the three methods presented here. Since the integer in the call in Figure 5-2 is the first argument we can find the pointer to the integer by examining the second word in the argument list. This is shown two ways in Figure 5-4.

Floating point values can be accessed in much the same manner as integer values. However, floating point values require either two or four words of storage depending on the precision with which the code was compiled. Floating point access is shown in Figure 5-5.

```
1000    CALL TEST0( INT%, FLT, ST$)
1010    CALL TEST1 BY REF( INT%, FLT, ST$)
```

FIGURE 5-2. Sample Basic-Plus 2 CALL Statements

```
TEST0:: CMPB     (R5),#3.        ;ARE THERE THREE ARGUMENTS?
        BNE      ERROR           ;NOPE, GO TO ERROR ROUTINE
                 .
                 .
                 .
```

FIGURE 5-3. Checking The Number Of Arguments

```
TEST0:: MOV      2(R5),R0        ;GET THE POINTER TO THE
                                 ;INTEGER
        MOV      (R0),R0         ;AND NOW THE INTEGER

        - or -

TEST0:: MOV      @2(R5),R0       ;R0 HAS THE VALUE OF THE
                                 ;INTEGER
```

FIGURE 5-4. Integer Access

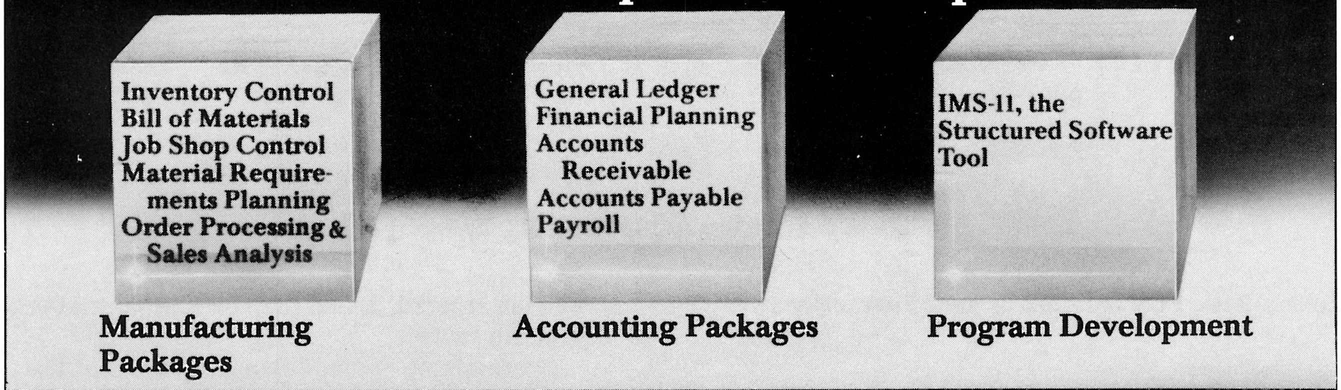
```
TEST0:: MOV      4(R5),R0        ;R0 POINTS TO THE FIRST
                                 ;WORD OF THE FLOATING POINT
                                 ;NUMBER
```

FIGURE 5-5. Floating Point Arguments

Award winning software packages for your DEC PDP-11

(operating under RSTS/E)

ICP Million Dollar Award and DEC AIP Gold Stars demonstrate our products' acceptance.



Manufacturing Packages

Tighter control over spiraling inventory costs, recognizing production backlogs *before* they develop and improved customer service are just some of the benefits of this set of packages. Some system highlights: multiple inventory costing methods, automatic rollup and recosting of bills, labor and work center utilization reports, shop load requirements and on-line "bucketless" MRP runs.

Accounting Packages

Our on-line, interactive packages provide you with immediate and *always* up-to-date accounting data. At a moment's notice you can verify customer credit, get account agings, pay vendors, and produce financial statements. Some advanced system features: multidivisional and multibank reporting, auto-cash application, variable dunning notices, A/P foreign gains/losses, budgeting, comparatives and G/L report writer.

Program Development

As a direct result of our staff's combined 100+ years of RSTS/E experience, the development of our applications under IMS-11 insures that all our software conforms to the same rigid development and documentation standards. As a stand-alone programming tool, development time of new systems can be cut up to 50%. Features include: programming standards, record I/O, keyed access, linked list and other data management functions, screen control, IMS sort utilities, menu and report directors, and JCL generation language.

Call (617) 489-3550 or write:

ims INTERACTIVE
MANAGEMENT
SYSTEMS
375 Concord Ave., Belmont, Mass. 02178

AUTHORIZED digital COMPUTER DISTRIBUTOR

Accessing string arguments is a bit more involved but is not complicated by any means. Figure 5-6 is an example of how to retrieve the pointers to a string argument when the CALL statement is used.

With the CALL by REF(), statement just the string start address is passed. The string length is not available. Figure 5-7 shows how to retrieve the start address of the passed string when the CALL by REF() statement is used.

```
TEST0:: MOV     6(R5),R0         ;R0 CONTAINS THE ADDRESS OF
                                 ; STRING HEADER NOW
        MOV     (R0)+,R1         ;R1 NOW HAS THE START ADDRESS
                                 ; OF THAT STRING
        MOV     (R0),R2          ;R2 NOW HAS THE LENGTH OF
                                 ; THAT STRING
                .
                .
                .
```

FIGURE 5-6. String Arguments

```
TEST1:: MOV     6(R5),R0         ;R0 NOW POINTS TO THE FIRST
                                 ; BYTE OF THE PASSED STRING
                .
                .
```

FIGURE 5-7. String Argument CALL by REF()

6.0 Using Basic Plus Modules In Your Subroutines

The Basic-Plus 2 support modules may be used within your MACRO subroutines to perform functions that require direct interfacing to the Basic-Plus 2 OTS. This enables you to manipulate strings, perform data type conversions, and force error conditions.

Any function you may want to emulate can be done by: 1) compiling the Basic-Plus 2 source statement using the /MAC option of the compiler, 2) examining the thread list created, 3) and then emulating that sequence in your subroutine.

By examining Figure 4-3 one can see that the thread routine that creates the string TEST$ is MOS$MM. It can be seen that the routine requires two arguments in the thread list. The first being the pointer to the source string header, and latter being the pointer to the destination string header. Using this information we can then re-write strings from our MACRO subroutine. Figure 6-1 is an example of the use of this routine in MACRO.

```
                 .
                 .
SOURCE: .WORD    SRCSTA          ;SOURCE STRING START ADDRESS
        .WORD    SRCLEN          ;SOURCE STRING LENGTH

DEST:   .WORD    DSTSTA          ;DESTINATION STRING START
                                 ; ADDRESS
        .WORD    DSTLEN          ;DESTINATION STRING LENGTH
                 .
                 .
        MOV      #10$,R4         ;START ADDRESS
        JMP      @(R4)+          ;START THE THREAD
10$:    .WORD    MOS$MM          ;THREAD ROUTINE
        .WORD    SOURCE          ;SOURCE STRING HEADER ADDRESS
        .WORD    DEST            ;DESTINATION STRING HEADER
                                 ; ADDRESS
        .WORD    20$             ;WHERE TO COME BACK
20$:             .
                 .
```

FIGURE 6-1. Example of MOS$MM String Routine

7.0 The Object Time System (OTS) Workspace

The OTS contains Basic-Plus 2's internal pointers and data structures. The structure of the OTS can be found on your Basic-Plus 2 distribution medium in a file called PRE.MAC. It describes the offsets of various internal structures as well as defining what these structures contain.

Included in the many bits of information you have access to in the OTS are: A pointer to the OTS ($AOTS), the current module name (NMPTR), the"on error goto' address (ONERGO), the current error number (ERRNUM), the current module name (NMPTR), the "on error goto" pending flag (EPEND), the pointers to the module name where an error occurred (ERN1, ERN2, ERN3), etc.

7.1 $AOTS

This is the offset that holds the pointer to the OTS work area. This value is currently defined as 52(8) (fifty-two octal). At run-time this location contains the pointer to the start of the OTS work area. Once the pointer to the OTS is retrieved, accessing the information available becomes a matter of examining the proper offsets (as described in PRE.MAC).

Figure 7-1 is an example of accessing the OTS pointer and using it to access some offset in the OTS.

7.2 NMPTR

This is currently offset 2 decimal.

This is a pointer to three sequential ascii words of the current module name. Figure 7-2 is an example of how to access this information.

7.3 ONERGO

This is currently offset 72 decimal.

This holds the"on|error goto" address for the line number specified. It is possible in a MACRO subroutine to temporarily change this address to intercept any Basic-Plus 2 errors. Figure 7-3 shows how this can be done.

```
                    .
                    .
$AOTS    =          52

         MOV        $AOTS,R0          ;GET THE POINTER TO THE OTS
         MOV        OPSYS(R0),R1      ;USING R0 AS THE BASE
                                      ;USE THE OFFSET TO GET THE
                                      ;OPERATING 'SYSTEM
                    .
                    .
                    .
```

FIGURE 7-1. How To Access The OTS Pointer

```
$AOTS    =          52
MNPTR    =          2.

         MOV        $AOTS,R0          ;POINT TO THE OTS
         MOV        MNPTR(R0),R1      ;R1 POINTS TO THE MODULE
                                      ; NAME
                    .
                    .
                    .
```

FIGURE 7-2. Accessing The Current Module Name

```
$AOTS    =          52
ONERGO   =          72.

         MOV        $AOTS,R0          ;POINT TO THE OTS
         MOV        ONERGO(R0),SAVADR ;SAVE THE CURRENT ADDRESS
         MOV        #INTRCP,ONERGO(R0) ;INTERCEPT ANY ERRORS
                    .

SAVADR:  .BKLW      1                 ;RESERVE A SLOT FOR OLD
                                      ; ERROR ADDRESS

INTRCP:             .
                    .
```

FIGURE 7-3. Intercepting Basic-Plus 2 Errors

7.4 ERRNUM

This is currently offset 94 decimal.

When an error occurs during run-time the value of the error code is placed at this offset in the OTS.

7.5 ERLNUM

This is currently offset 96 decimal.

When an error occurs during run-time the value of the line number is placed at this offset in the OTS.

7.6 ERLPTR

This is currently offset 98 decimal.

After an error has been trapped, a pointer to the start address of that line number and the line number itself is placed at this offset in the OTS. Figure 7-4 demonstrates how to retrieve this line number.

7.7 ERN1, ERN2, ERN3

These offsets are contiguous and start at offset 134 decimal.

These three locations contain the module name where the error occurred as three consecutive ascii words.

```
$AOTS     =        52
ERLPTR    =        98.

          MOV      $AOTS,R0             ;GET THE OTS POINTER
          MOV      ERLPTR(R0),R1        ;POINT TO THE LINE NUMBER
          MOV      (R1)+,R2             ;R2 HAS THE LINE NUMBER
          MOV      (R1),R3              ;R3 HAS THE ADDRESS
                                        ;WHERE THE LINE STARTS
                   .
                   .
```

FIGURE 7-4. Retrieving The Error Line Number

8.0 Functional Applications

Many of functions discussed have been put to use by the authors. Below is a sampling of those applications.

8.1 System or Application Specific Data

An application we have found quite useful is interfacing each program of an application to a MACRO subroutine. In the MACRO subroutine is a PSECT containing the client name, installation specific data (operating system, hardware configuration, location of data files), serial numbers, expiration dates for demo packages, etc. Figure 8-1 demonstrates how this can be accomplished.

```
910     MAP     (INIT)
                 Exp.Date%                   ! Expiration Date
                ,Site.Name$=25%              ! Site name
                ,Site.Address$=25%           ! Site address
                ,Site.Address.2$=25%         ! Site address extended
                ,Site.Zip.Code$=9%           ! Site zip code
                ,Site.State$=2%              ! Site state


.PSECT  INIT,RW,D,GBL,REL,OVR

EXPDAT::.WORD   0                            ;EXPIRATION DATE
SITNAM::.ASCII  "Customer Site                 " ;SITE NAME
SITADR::.ASCII  "12345 Tape Drive              " ;SITE ADDR
SITAD2::.ASCII  "Newport Beach                 " ;SITE ADDR 2
SITZIP::.ASCII  "90780    "                  ;SITE ZIP
SITSTA::.ASCII  "CA"                         ;SITE STATE
```

FIGURE 8-1. Overlaying Data Through MAPS

We have also incorporated the use of the task-builder's ability to produce symbol tables to interface with DEC's ONLPAT automated patch facility. By using the MAKSIL program supplied on your RSTS/E distribution medium you can produce symbolically patchable tasks to enable the patching of certain features within a software package. Figure 8-2 shows a sample run of the program MAKSIL to generate an output SIL (Save Image Library) that can be patched by ONLPAT.

```
RUN $MAKSIL

MAKSIL  V7.0-07 RSTS V7.0-07 Timesharing
Resident Library name? PROGRA
Task-built Resident Library input file <PROGRA.TSK>? <CR>
Include symbol table (Yes/No) <Yes>? <CR>
Symbol table input file <PROGRA.STB>? <CR>
Task Image SIL output file <PROGRA.SIL>? <CR>
PROGRA built in 16 K-words, 118 symbols in the directory
PROGRA.SIL renamed to PROGRA.SIL<104>

RUN PROGRA.SIL
```

FIGURE 8-2. Running MAKSIL

8.2 Screen Compilation

In any screen oriented application, a large number of screens are needed. The Basic-Plus 2 code needed to position and display a screen token is about 13 words of thread space. For a busy screen it would be easy to consume 500 words of thread space. The Basic-Plus 2 code needed to construct all of the screens for an entire application can easily consume several K-words of memory. This thread space estimate does not include the actual data to be displayed, just the code generated by the Basic-Plus 2 compiler to concatenate the various literals and variables into one string so that the screen can be displayed with a single write request.

If the screens are constructed by the MACRO assembler at assembly time there will be no overhead in the Basic-Plus 2 program to construct the screen. To ease the task of generating screens within our own programs, we have developed a screen compiler, written in Basic-Plus 2 and MACRO, that takes an ascii text file that contains the screen image and generates a MACRO file ready to be assembled.

8.3 Echo Block Mode Simulation

This goes hand-in-hand with the screen application. Almost any application program needs to accept input from the user in a highly controlled fashion. We have accomplished this with a MACRO subprogram. The subprogram positions the cursor and accepts the user input. The subprogram has many validation options and input modes. This eliminates the need for the programmer to code these functions in his program.

The subprogram validates dates, insures valid numeric amount and performs editing functions on the string similar to the CVT$$ or EDIT$ functions in the compiler itself. It also has the capability to display an appropriate error or help message to the user on a selected line on the screen.

We have found that this subprogram has saved us countless man-hours in application development time.

8.4 A One-Shot Spool Call Interface

Currently the one-shot spool monitor call (UU.SPL) is not supported by Basic-Plus 2. This feature has been implemented in a MACRO subprogram capable of performing the call for the requesting program.

C H A N G E S ? ? ? ? ? ?

Are you changing addresses? Please let us know so you won't miss one issue of the *RSTS Professional.*

FORMER	**PRESENT/Near Future**
Name ____________________	Name ____________________
Address ____________________	Address ____________________
City/State ____________________	City/State ____________________
Zip ____________________	Zip ____________________

8.5 Date Utility Routines

Date handling is a thorn in the side for most programmers. To eliminate this "pain" we have incorporated a full complement of date utility CALLs to handle all date functions. Using the RSTS/E internal date construct we have a set of routines that translates string-to-integer and integer-to-string dates.

8.6 Setting Terminal Characteristics

It is often necessary to have specific terminal characteristics set for a terminal to ensure proper operation of software. We found it highly desirable to have the programs themselves perform this function at run time. We also wanted the ability to reset the terminal to its previous characteristics. Thus was born the SETTTY and CLRTTY calls described earlier. We have the SETTTY call save the current state of the terminal in a small buffer in the module, then set escape sequence, no stall, and CTRL/C as its private delimiter. Also, optionally, the width. When the CLRTTY call is made, the terminal is returned to its original state.

8.7 Internal Sort Capability

One of the most frequent requirements of an applications program is that of sorting. Each application sorts a different type of data, often in complex sequences. The only way to handle this (without writing a special sort routine each time) has been to put the data you want sorted into an intermediate file and 'chain' to a general purpose sorting program.

Through the use of MACRO interfaces to Basic-Plus 2, we were able to develop a complete sorting system which is fully resident within the Basic-Plus 2 applications program. This sort system is versatile enough to handle any type or volume of data likely to be encountered. Because it interfaces directly with Basic-Plus 2, it is extremely easy to use. And, since it uses MACRO, it is very fast.

Appendix A

Sample MACRO Subprogram

```
TITLE   FATAL,<PRINT FATAL ERROR MESSAGE>,01,29-OCT-80,SPD

; WRITTEN BY: STEVEN P. DAVIS
;
; COPYRIGHT (C) 1979, 1980
; SOFTWARE TECHNIQUES, INC.
; LOS ALAMITOS, CA 90720
;
; THIS SOFTWARE  IS  BEING  PROVIDED FREE OF CHARGE TO THE NORTH
; AMERICAN DECUS ORGANIZATION  AND  MAY  BE COPIED ONLY WITH THE
; INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE, OR ANY
; OTHER COPIES THEREOF, MAY  NOT  BE  PROVIDED OR OTHERWISE MADE
; AVAILABLE  TO  ANY  OTHER PERSON EXCEPT FOR NON-COMMERCIAL USE
; AND  TO  ONE  WHO  AGREES TO THESE LICENSE TERMS. TITLE TO AND
; OWNERSHIP  OF  THE  SOFTWARE SHALL  AT  ALL  TIMES  REMAIN  IN
; SOFTWARE TECHNIQUES.
;
; THIS INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE WITHOUT
; NOTICE AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY SOFTWARE
; TECHNIQUES.
;
; THIS SOFTWARE  IS  UN-RELEASED  AND SOFTWARE TECHNIQUES HAS NO
; COMMITMENT TO SUPPORT IT AT THIS TIME, UNLESS STATED ELSEWHERE
; IN WRITING.


 .SBTTL  SET UP BP2 OFFSETS IN OTS

 .EQUATE $AOTS,  52                  ;POINTER TO BP2 OTS WORK AREA

 .EQUATE ERRNUM, 94.                 ;ERROR NUMBER
 .EQUATE ERLPTR, 98.                 ;POINTER TO CURRENT LINE
 .EQUATE ERN1,   134.                ;THE ERROR MOUDLE NAME (ASCII)

 .SBTTL  SOME CONSTANTS

 .EQUATE CR,     15                  ;CARRIAGE RETURN
 .EQUATE LF,     12                  ;LINE FEED
 .EQUATE WORK,   200.                ;WORK SPACE FOR STRING

 .SBTTL  MESSAGE MACRO

 .MACRO  MOVMSG  TXT
         TMPORG  TEXT
 .NLIST  BEX
 $$$     =       .
         .ASCII  TXT
 $$$$    =       .-$$$
 .LIST   BEX
         UNORG
 .IF     EQ      $$$$
 .ERROR  ;NULL LENGTH STRING
 .ENDC
         CALL    60$,R4
         .WORD   $$$
         .WORD   $$$$
 .ENDM   MOVMSG
```

```
;+
; FATAL  - PRINT FATAL ERROR MESSAGE
;
; CALL: CALL FATAL
;
; BACK: "??Unexpected error #N occured at line N in "XXXXXX"
;       "?RSTS ERROR MESSAGE
;       "?Please write this error down and report it"
;
;       BLOWS ECHO CONTROL!!!
;-
.ENABL  LSB
.ENABL  LC

        DEFORG  FATAL

        SUB     #WORK,SP        ;GET SOME STRING WORK SPACE
        MOV     SP,R5           ;POINT TO THE STRING
        MOV     $AOTS,R3        ;POINT TO BP2 OTS
        MOVMSG  <"??Unexpected error #"> ;START UP THE MESSAGE
        MOV     ERRNUM(R3),R1   ;GET THE ERROR NUMBER
        CALLX   NUM,R4,<0>      ;CONVERT THE NUMBER
        MOVMSG  <" at line ">   ;MORE OF THE MESSAGE
        MOV     ERLPTR(R3),R1   ;POINT TO THE OFFENDING LINE
        MOV     (R1),R1         ;AND GET IT
        CALLX   NUM,R4,<0>      ;CONVERT THAT LINE NUMBER
        MOVMSG  <' in "'>       ;NOW SET FOR MODULE NAME
        MOV     #ERN1,R1        ;MOVE THE OFFSET IN R1
        ADD     R3,R1           ;POINT TO THE NAME
        MOV     #6.,R2          ;SET THE MAX LENGTH
10$:    MOVB    (R1)+,R0        ;GET THE CHAR
        CMP     #' ,R0          ;HIT A SPACE YET?
        BEQ     20$             ;YES, THAT'S ALL THEN
        MOVB    R0,(R5)+        ;PUSH IT IN
        SOB     R2,10$          ;AND KEEP IT UP
20$:    MOVMSG  <'"'<CR><LF>>   ;FINISH IT OFF
        CALLX   SETFQB          ;SET UP THE FIRQB
        MOVB    #ERRFQ,(R0)+    ;SET FUNC TO RETURN
        MOVB    ERRNUM(R3),(R0) ;SET THE ERROR NUMBER
        CALFIP                  ;AND DO THE CALL
        CLR     XRB             ;CLEAR TO MARK END
        MOV     R0,-(SP)        ;SAVE THE START OF TEXT
30$:    TSTB    (R0)+           ;CHECK FOR NULL
        BNE     30$             ;GO TILL NULL IS HIT
        DEC     R0              ;IGNORE THE NULL
        SUB     (SP),R0         ;R0 HAS LENGTH NOW
        MOV     R0,50$          ;SET LOCATION
        MOV     (SP)+,40$       ;SET LENGTH
        CALL    60$,R4          ;AND MOVE IT IN
40$:    .WORD   0               ;RESERVE A WORD
50$:    .WORD   0               ;RESERVE A WORD
        MOVMSG  <<CR><LF>>      ;START OUT THE LAST MESSAGE
        MOVMSG  <"?Please write this error down and report it">
        MOVMSG  <<CR><LF>>      ;AND ALL DONE NOW
        SUB     SP,R5           ;R5 NOW HAS THE LENGTH
        CALLX   SETXRB          ;SET THE XRB
        MOV     R5,(R0)         ;THE LENGTH
        MOV     (R0)+,(R0)+     ;IN BOTH
        MOV     SP,(R0)         ;THE LOCATION
        .WRITE                  ;WRITE IT OUT
        ADD     #WORK,SP        ;GIVE BACK THE WORK
        RETURN                  ;BACK FROM WINCE WE CAME

; MOVE A MESSAGE IN BUFFER

60$:    MOV     (R4)+,R0        ;GET THE LOCATION
        MOV     (R4)+,R1        ;AND THE LENGTH
70$:    MOVB    (R0)+,(R5)+     ;START THE MOVES
        SOB     R1,70$          ;MOVE IT IN
        RETURN  R4              ;AND GO BACK

.DSABL  LSB
.END
```

APPENDIX B

Sample RUN Using FATAL.MAC

```
>RUN $MAC.TSK
MAC>FATAL=COMMON,FATAL
MAC>NUM=COMMON,NUM
MAC>^C
>RUN $LBR.TSK
LBR>LB:SA/CR
LBR>LB:SA=FATAL,NUM
LBR>^C
>RUN $BASIC2.TSK

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


Basic2

NEW TEST

Basic2

1000      ON ERROR GOTO 19000
1010      INPUT "Force which error "; ERR%
1020      CALL "$FRCER" BY REF(ERR%)
1030      STOP
19000     CALL FATAL
19010     RESUME 32767
32767     END

SAVE

Basic2

COM/OBJ

Basic2

DSK LB:SA/LB-LB:BP2COM

Basic2

BUI TEST

Basic2

TKB @TEST
>RUN TEST
Force which error ? 5
??Unexpected error #5 at line 1020 in "TEST"
?Can't find file or account
?Please write this error down and report it
>
```

GOOD NEWS FROM SENERUG '80

By Howie Brown, Information Systems Inc.

Just over a year ago began the SENERUG experiment...RSTS users independent of the DECUS umbrella, providing mutual support and marketplace awareness for themselves. It looked viable then. Today we report the success of the First Independent RSTS Users Seminar and Annual Meeting, held October 22-23 at the Sheraton-Mansfield, Mansfield, MA.

This was, first and foremost, a conference by, of and for users. Questions about Brand X add-on memory, for example, were welcome and fully answered, since without marketplace information, including pricing, we can't do our jobs. Our speakers list reflected the considerable reservoir of accumulated experience that users both well-known and obscure have to offer. The entire event was put together by volunteer users who each put a little love into their task. And finally, and significantly, two days of technical interaction, including two notable lunches, cost each attendee under $100. (A little too cheap, maybe next time we'll make some money out of it.)

Attendance: about 70, with all six New England states represented. The Long Distance Award went to Rob Frobhoedt of Scientific Atlanta.

DAY 1. Two day-long sessions. Tom Considine (Considine Computing) and Gerry Tolman (consultant) moderated "Optimizing RSTS Performance". Gerry Tolman opened the session with an overview of performance measurement. Dave Mallery (Nationwide Data Dialog, RSTS Pro) ran down hardware considerations...should I get a DH or a D-zede ? (He caught a bad accent at the UK DECUS). What problems will I run into with a "foreign" RM drive? Following a detailed walk through a SYSGEN dialog by Gerry Tolman, Tom Considine laid out a number of reasonable procedures to control disk traffic and ease a major bottleneck of RSTS systems. Disregarding Tom's promise not to "talk dirty", Dave Mallery offered an esoteric procedure followed only by junkies (including my own shop) to center and pre-extend your UFD's. Seriously behind schedule, we broke for lunch.

Meanwhile, it was PASCAL Tutorial day next door, moderated by Mark Lifland (MBL Associates). The morning session was exactly that . . . an introduction for the newcomer.Pete Kaczowka (TSC) walked us through a step by step overview of the language, no specific compiler. A complete set of slides, each illustrating a component of the language, comprised a text for the talk, and a set of handouts mirrored the slides. PASCAL plusses: extensive structuring, efficiency when fully compiled. Minuses: weak string handling and file handling services.

Lunch. Roast beef, peas, Carl Marbach (Teachers Service Organization, RSTS Pro) and Dave Mallery, keynote speakers. Weak jokes and good feelings. We're the wave of the RSTS future. "If I'm not for myself who will be?" asks Carl, quoting a medieval theologian.

Afternoon in the RSTS room. Tom Griffiths (Evans, Griffiths and Hart) has a meaty handout: "Machine Language Under RSTS/E". What's to be gained . . . when and what to re-code. He speaks from EGH's considerable experience in implementing application software as run-time systems. We get into garbage collection as a RSTS problem . . . what it is, what to do about it.

Now all five experts take to the speakers table and two brave users put their $STATUS output on the projector. Free consulting for two systems from some of the best in the field: get rid of those multiple public disks . . . too much FIP . . . you need an 11/44. Arguments among the panel and reaction from the rest of us . . . an electric atmosphere and a memorable session.

PASCAL afternoon. Applications experience under RSTS. Jim Smith of Greater Lawrence Vocational and Technical High School reports having totally converted his system to PASCAL, and estimates a impressive speed improvement. Jeff Stulin (Computer Software Consultants) concentrates on the Structured aspect of the language. The Structure is the language's great advantage over COBOL, BASIC, DIBOL. Structure becomes a programming convenience : build a procedure library and simply append them to the program root; unlike other languages, variable-names are handled in such a way that they don't need to be consistent between main program and procedure. As the session ends people from the RSTS Performance Session come looking for PASCAL handouts, grumbling about not being able to be at both sessions.

DAY 2. The Business Meeting opens with a membership debate: consultants without their own installations previously denied voting privileges, want the right to voting membership. A dozen volunteers to modify the by-laws. Two new At-large seats on the Board are contested; the newly-elected Board has representation from Massachusetts (Ken Graves, Kendall Co. and Tony Bossi, Agar Supply), Connecticut (Larry Shatsoff, Bridgeport Hydraulic) and Rhode Island (Monica Collins, Geo. Mann Co, Bob Grund, Greenwood Credit Union, Howie Brown, Information Systems).

Rella Hines, Executive Director of the Hewlett-Packard General Systems Users Group, is our guest speaker. We have some things in common, she says; independence of the vendor, representation of the user interest. We have some differences; no shadow of a DECUS to contend with, and the HP base is considerably smaller. The SENERUG potential is impressive; she points to an estimated 15,000 DEC machines in New England alone. Food for thought for SENERUG.

Lunch and awards. Ken Graves is surprised by a plaque presented to him for his efforts as Seminar Coordinator. Thanks go out to everyone who's helped us through our first year. And a framed Resolution, now hanging on the appropriate wall, was presented to Rich Johnston and Herb Yarborough, Unit Managers of the East Providence Branch Office of DEC, for "a clear improvement in . . . problem areas" identified by SENERUG members at a February 1980 meeting with DEC personnel, and for the "successful efforts of the Branch to provide outstanding customer support" during 1980.

Thursday afternoon saw two simultaneous panels geared to marketplace awareness: Hardware, moderated by Larry Singband (Merchandata, 'The Alternative to DBMS', RSTS Professional, Sept. 1980), and Software, moderated by Monica Collins and Dennis Thibeault (Commercial Union).

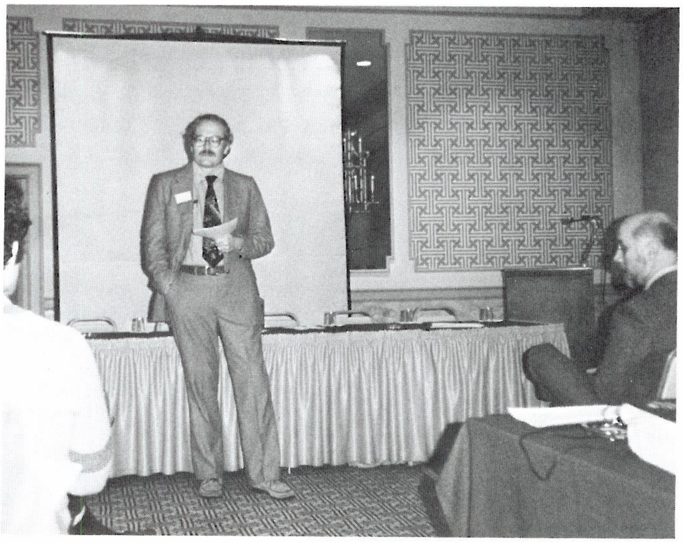
Hardware session highlights: Bill Okerman of Tektronics wows the crowd with a cassette-driven demo on standalone graphics machine. The screen is filled with business graphs, drafting applications, mapping, games and others. Chris Banus of Nordata introduces us to the world of multi-vendor systems which they assemble, burn in and deliver. He explains Nordata's selection criteria of peripherals; refers to third-party service alternatives available through Tymshare and CDC. Multi-vendor peripherals are also the subject of Pat West, J&J Associates; he introduces Louis Perez, Centronix printers, and Bob Otten, Minicomputer Technology disk controllers. Representatives of Monolithic Memory set up a display and answer questions about their MOS compatible memory.

Software highlights: Larry Shatsoff has done some homework regarding sources of RSTS-compatible software; he refers us to useful reference publications. Eric Mootheart, Data Processing Design, gives us a view of WORD-11, and the discussion gets into detailed cost-effectiveness and usability. Inevitably, we spill over into SAVER. We are introduced to Page Soltveldt, DPD's New York staff. Greg Johnson (IIRI, 'File Structure and Accessing Techniques', RSTS Professional, September 1980) delivers an overview of data management techniques. He is followed by presentations from Meredith Gilbert, Cincom, TOTAL; and Brian Boyle, Interactive Management Systems, IMS-11.

SUMMARY. Most everyone turned in evaluation forms. The seminar wasn't perfect; speakers were uneven; not everything started on time. Overall, the people who came were happy with the experience: comments ranged from "good" to enthusiastic "excellent". Especially indicative of success was the high proportion of people willing to help with the next one. Proceedings of SENERUG '80 may become available; let us know if you want to receive them.

What's next? The most likely directions seem to be: 1) expanding our hardware and software panel sessions into a full-blown annual multivendor trade show; and 2) taking our technical sessions and making them available to more people, that is, help make them happen outside of New England. Interested? write SENERUG, PO Box 3043, Pawtucket, RI 02861

SENERUG 1980



TOM GRIFFITHS

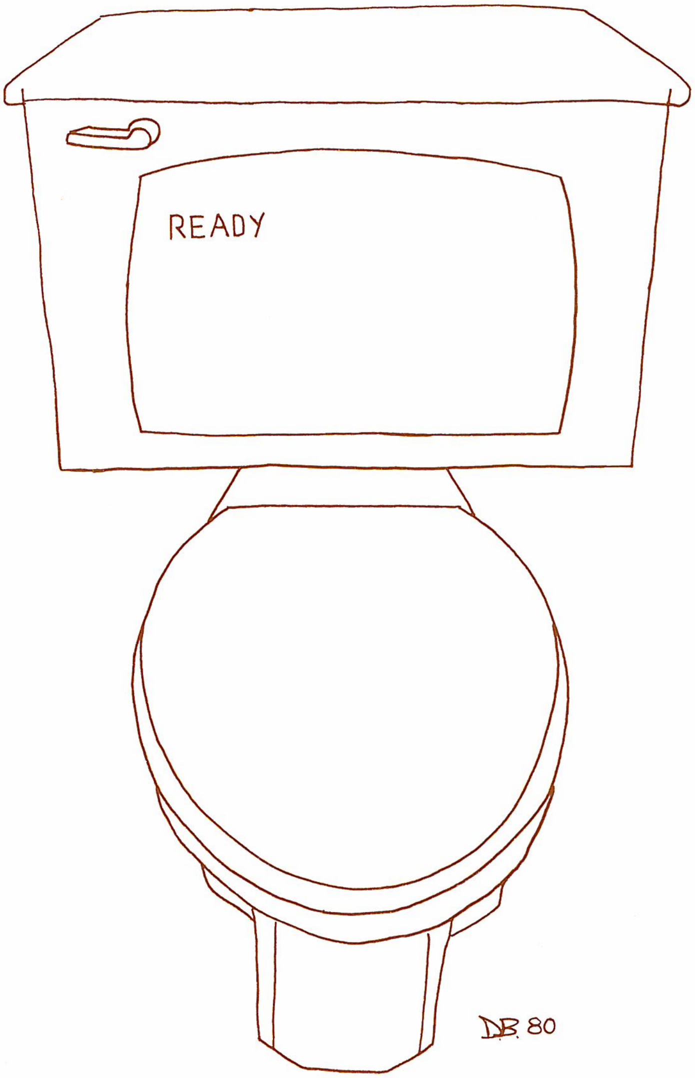
HOW TO JOIN DECUS

Send for a membership application to one of the addresses listed.
State that you want to join the RSTS SIG.
Tell them that you received this information in the ***RSTS Professional.***

Australian Chapter
DECUS AUSTRALIA
P.O. Box 384
Chatswood
NSW 2067
Australia

European Chapter
DECUS Europe
P.O. Box 510
12, Av. Des Morgines
CH-1213 Petit-Lancy 1/GE
Switzerland

U.S. and
CANADIAN CHAPTERS and others
DECUS
MR2-3/E55
One Iron Way
Marlboro, MA 01752
U.S.A.



A STANDARD FORMAT FOR PROGRAM DIALOGUE

By R.C. Cannon, British Aerospace, Kingston, Surrey, England

Abstract

For easy use by inexperienced users, programs can obtain the values of parameters by a self-explanatory dialogue of questions. The paper describes a dialogue structure that uses default values and various terminators to minimise the number of questions answered.

Consistent format and simplified programming is ensured by using a set of standard Input functions to perform all dialogue I/O and error handling.

INTRODUCTION

Most programs require the values of parameters to be input by means of commands and switches, or by a dialogue of questions and answers.

Command Driven Programs

It is necessary to have extensive documentation and regular experience in using them. Recovery from keyboard errors is often poor, usually requiring a lengthy command to be retyped. The default values are not obvious and must be remembered or looked up, although if the default values are used the commands can be made compact. They are not normally suitable for use by inexperienced or casual users.

Dialogue Driven Programs

With well designed questions this type of program can be very easy to use, even by persons with little experience. Often minimal documentation is required as the questions are self explanatory and the default values for the replies can be printed with the question. Recovery from keyboard errors is very good as only the question being answered needs to be repeated. Unfortunately the experienced user will find the long print out of questions tedious, particularly when he knows he wants to use the default values for most of the replies.

The number of questions to be answered can be minimised by structuring the dialogue of questions into blocks and using the type of terminator used with the replies to control the flow through the blocks of questions. A further improvement is obtained by enabling the user to save his own default values so he can use them later. A standard format is ensured and programming effort minimised by using a set of standard INPUT functions.

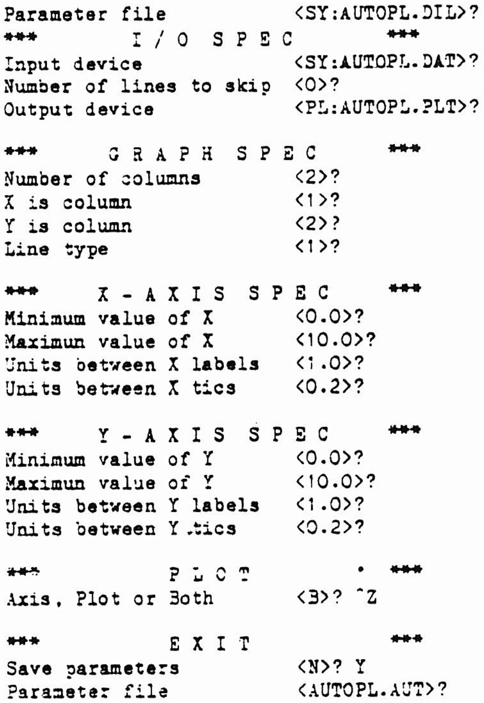
Structured Dialogue

The questions are divided into a number of blocks, each block asking questions on related parameters. A block should start with its heading and for ease of use the number of questions in a block should be limited to about six. Each question prints its default reply between angle brackets i.e., < default > . Normally when the reply to a question has been accepted it becomes the new default.

The first question of the dialogue is for the parameter file, the default to this gives the standard default values. The last questions give an option to save the current default parameter values in a file; this file can then be used with a subsequent run of the program, thus all the default parameter values can be preset for a task.

Typical Block Structure

Part of the dialogue for a graph plotting program (AUTOPLot) is show to illustrate a block structure.



```
Parameter file              <SY:AUTOPL.DIL>?
***        I / O  S P E C          ***
Input device                <SY:AUTOPL.DAT>?
Number of lines to skip     <0>?
Output device               <PL:AUTOPL.PLT>?

***      G R A P H  S P E C        ***
Number of columns           <2>?
X is column                 <1>?
Y is column                 <2>?
Line type                   <1>?

***     X - A X I S  S P E C       ***
Minimum value of X          <0.0>?
Maximun value of X          <10.0>?
Units between X labels      <1.0>?
Units between X tics        <0.2>?

***     Y - A X I S  S P E C       ***
Minimum value of Y          <0.0>?
Maximun value of Y          <10.0>?
Units between Y labels      <1.0>?
Units between Y .tics       <0.2>?

***            P L O T         .   ***
Axis, Plot or Both          <B>? ^Z

***            E X I T             ***
Save parameters             <N>? Y
Parameter file              <AUTOPL.AUT>?
```

Use of Terminators

Now that the default values can be used on most questions a means is required to skip past them, and to return to a question if modification is needed.

Six terminators are defined as having a consistent action to control the flow through the questions.

1. LF (Line Feed) — Skip to the start of the next block.

2. CR (Carriage Return) — Ask the next question.

3. ESC (Escape or Alt) — Ask previous question, or skip back to the start of the previous block if on the first question of a block.

4. FF (Form Feed) — Skip to the last block or skip several blocks. Usually to a question that starts the program executing a task with the parameters set.

5. CTRL/Z — Jump to Exit, usually the parameter saving question.

6. CRTL/C — Jump back over several sections, repeated CTRL/Cs will abort.

By using a standard philosophy all programs are very similar to run. The inexperienced user can go through all the questions with CR, use ESC to go back to change replies and exit with CTRL/Z. If it is known that all the defaults are correct then the parameter file question can be terminated with FF to skip past all other questions. As the user becomes more experienced he will quickly become fluent at using all the terminators to minimise the number of questions answered.

Figure 1 shows the block structure and the effect of the terminators for an AUTOPLOT program. This is very easy to use to plot almost any tabular data on a Textronix plotter. It was initially written to emulate the AUTOPLOT facility of the Hewlett Packard 2648A graphics V.D.U.

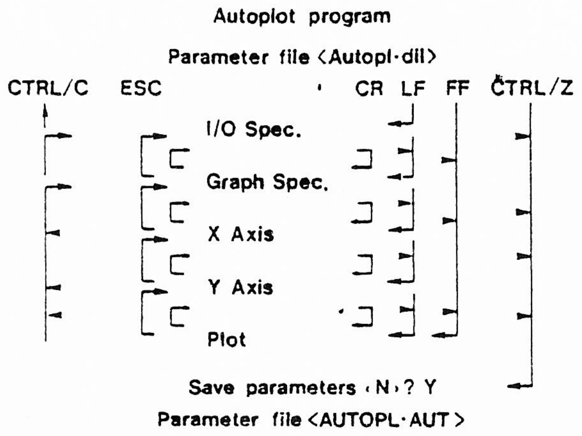


FIGURE 1. Block Structure Example

Input Functions

All questions and replies are handled using a set of standard input functions to ensure a consistent format and minimise programming.

There are a number of different sets of functions, written in BASIC+ for use with RSTS/E, to suit different applications.

1. INPUT.BAS — For simple programs with only a few questions, saving of parameters is not normally required. The question text and default value are passed in the call to the function.

   REPLY$ = FNINPUT$("Question", "Default")

   For updating the default, REPLY$ is used for "Default".

2. INPUTA.BAS — For programs with a large number of questions, with the option for the user to save his defaults, the questions and defaults are stored in string arrays in a virtual array file. The defaults are copies to another array before being used.

   REPLY$ = FNINPUT$(QuestionNo.%)

   This version updates the default to the value input. The question number is the array element number for the question and default.

3. INPUTE.BAS, INPUTF.BAS — A version of each of the above for use with direct cursor addressing V.D.U.'s where the questions are static on the screen, and Echo control can be used. Row and column numbers are required as extra parameters in the call or the question array.

Each set of functions includes the following functions which validate the reply as necessary; an error message is printed and the question asked again if there is an error. The set used by INPUTA is as follows:

```
REPLY$   = FNINPUT$(QuestionNo.%)            !Char. string
REPLY    = FNINPUT(QuestionNo.%)             !Floating point
REPLY%   = FNINPUT%(QuestionNo.%)            !Integer
REPLY%   = FNINPUTC%(QuestionNo.%, "CHOICE")
```

The first character of the reply must be one of those in "CHOICE", and the function returns the position of it within "CHOICE". This version can also validate the first n characters, where n is the length of the default reply.

The questions and standard default values are in the virtual arrays ITEXT.$( ) and INDFLT( ), the defaults are copied to the array IDFLT.$( ) for use by the input functions. All types of defaults are stored as a character string.

Two functions to print error messages are included to ensure a standard format:

```
ERROR% = FNERRMES%(ERR)   !System message
ERROR% = FNERRTXT%("PROGRAMMERS MESSAGE")
```

The INPUT functions set the variable ILEN.% to the length of the actual string of characters input and ITERM.% according to the terminator used.

Terminator	CTRL/C	CTRL/Z	LF	CR	ESC	FF
ITERM.%	-28	-11	1	2	3	4

If a function is called with ITERM.% set to 5 then the functions returns the default value, without any question or reply, to minimise the code required to initialise variables to the default values.

To avoid conflict, all variable names start with 'I' and contain '.' (which is the last character if a variable is used outside the function). Temporary variables take the form T.. or T.n, n is 0 to 9.

Coding

The dialogue should be designed to avoid processing data between the input of parameters; this simplifies the code. Each call of the INPUT function is followed by an ON ITERM.% GOTO statement to pass control to the appropriate line number. CTRL/Z and CTRL/C are handled by trapping the 'Subscript out of range' error caused by the negative value of ITERM.%.

Programs with only a moderate amount of dialogue are best served by the set of functions in the file INPUT.BAS: the question text and the default value are function parameters. There is minimal programming overhead.

Programs with a large number of questions should use the INPUTA.BAS set of functions and include code to save and restore the users parameters. Programming effort is minimised by using a standard code to carry out these actions.

The size of the standard code files is minimised, to reduce the time taken to append them, by storing a version with all comments deleted in a library account. A TECO macro is used to delete comments.

CONCLUSIONS

Both experienced and casual users have found that programs using a structured dialogue controlled by the terminators are very easy to use with little reference to documentation. Saving of the default parameters enables a task to be repeated rapidly.

Using a standard set of INPUT functions with standard parameter saving and restoring code ensures a consistent format with the minimum of new code.

The examples show an implementation in BASIC+ for use with RSTS/E but the philosophy is applicable to any other operating system and language.

APPENDICES

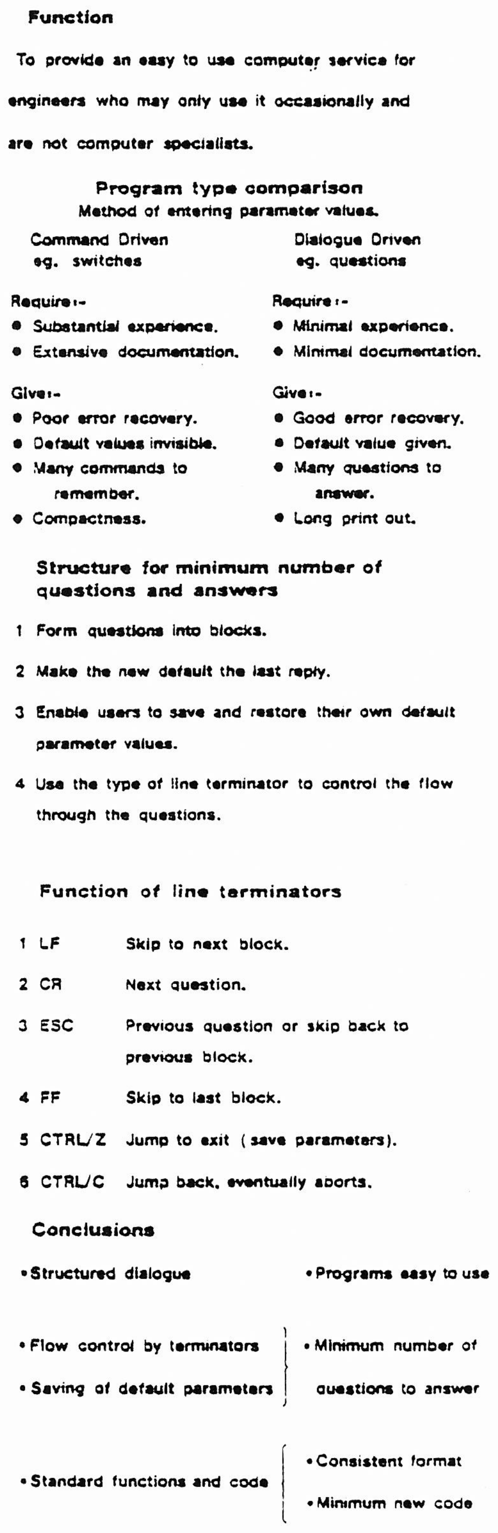
A — Extracts from the presentation slides.

B — Standard code for initialisation, saving and restoring parameters together with the set of functions forming INPUTA.BAS.

C — A program to set up a file containing questions and standard defaults.

D — A TECO macro to delete comments from programs.

APPENDIX A
EXTRACTS FROM PRESENTATION SLIDES



Function

To provide an easy to use computer service for engineers who may only use it occasionally and are not computer specialists.

Program type comparison
Method of entering parameter values.

Command Driven eg. switches	Dialogue Driven eg. questions
Require:-	Require:-
• Substantial experience.	• Minimal experience.
• Extensive documentation.	• Minimal documentation.
Give:-	Give:-
• Poor error recovery.	• Good error recovery.
• Default values invisible.	• Default value given.
• Many commands to remember.	• Many questions to answer.
• Compactness.	• Long print out.

Structure for minimum number of questions and answers

1 Form questions into blocks.

2 Make the new default the last reply.

3 Enable users to save and restore their own default parameter values.

4 Use the type of line terminator to control the flow through the questions.

Function of line terminators

1 LF	Skip to next block.
2 CR	Next question.
3 ESC	Previous question or skip back to previous block.
4 FF	Skip to last block.
5 CTRL/Z	Jump to exit (save parameters).
6 CTRL/C	Jump back, eventually aborts.

Conclusions

•Structured dialogue	•Programs easy to use
•Flow control by terminators •Saving of default parameters	•Minimum number of questions to answer
•Standard functions and code	•Consistent format •Minimum new code

APPENDIX B
DIAPRO.BAS AND INPUTA.BAS STANDARD CODE

```
2       !DIAPRO Framework for DIAlog PROgrammes.
10      EXTEND
101 !A general program containing the basic code   &
! for initialisation, saving and restoring         &
! parameters; used with Input functions in INPUTA  &
! The Questions and standard default parameter     &
! strings are in a virtual array using channel     &
! 10, see line 910., channel 9 used to save.       &
!                                                  &
! ITEXT.$                         INDFLTS          &
! (0%)   Parameter file           SY:DIAPRO        &
! (1%)      E X I T                                &
! (2%)   Save parameters           N               &
!                                                  &
! The progam name with extension .DIL is used for  &
! the dialog and standard defaults file. The first &
! first three letters of the program name are used &
! as the default extension for the users parameter &
! file.                                            &

910 DIM #10%, ITEXT.$(40%)=64%  !Question text     &
            , INDFLT$(40)=64%   !Std. defaults     &
\   DIM       IDFLT.$(40%)      !Current defaults  &

1100                                               &
! ++++++++++FORM PROMPT AND DEFAULTS++++++++++++++ &

1110 ONERROR GOTO 19000            !Std. errors    &
\ IDFLT.$(0%),SYDFLT$="SY:DIAPRO.DIL"!Dilogue file &
\ OPEN SYDFLT$ FOR INPUT AS FILE 10%               &
                 ,MODE 8192%       !Read only      &
\ PAREXT$= "DIA"                   !3 chr name     &

1120 ITERM.% = 2%                  !Cancel init dflts &
\ DEFALT$ = FNINPUT$(0%)           !Ask defaults   &
\ GOTO 32700 IF ITERM.% < 0%       !^Z, ^C    &
             OR ITERM.% = 3%       !or ESC exits   &
\ ITERM%= ITERM.%                  !For after dflts &
\ ITERM.%=5%                       !Set defaults   &
\ IF DEFALT$ =CVT$$(SYDFLT$,3%) !If standard       &
   THEN IDFLT.$(I%)=INDFLT$(I%) FOR I%=0% TO 40%   &
   ELSE DEFALT$=DEFALT$+"."+PAREXT$!Add extension  &
    UNLESS INSTR(1%,DEFALT$,".")!If not one        &
\ OPEN DEFALT$ FOR INPUT AS FILE 9% !Users dflts   &
\ FOR I%=0% TO 40%                 !Each dflt      &
\  INPUTLINE #9%, T..$             !               &
\  IDFLT.$(I%) = CVT$$(T..$,5%) !Strip term.s      &
\ NEXT I%                          !               &
\ CLOSE 9%                         !               &

2000 !+++TYPICAL CODE+++++++++++++++++++++++++++++ &
  ON ITERM.% GOTO 2100,2100,2100,3000,2110         &
!                   LF   CR   ESC  FF  Init.       &

2100  T..%=FNINPUTH%(4%)           !Header text    &
\ T..$=SYS(CHR$(6%)+CHR$(-7%))     !Enable ^C trap &

2110  P1.1=FNINPUT(5%)                             &
\ ON ITERM.% GOTO 2200,2120,2100,3000,2120         &

2120  P1.2%=FNINPUT%(6%)                           &
\ ON ITERM.% GOTO 2200,2200,2100,3000,2210         &
```

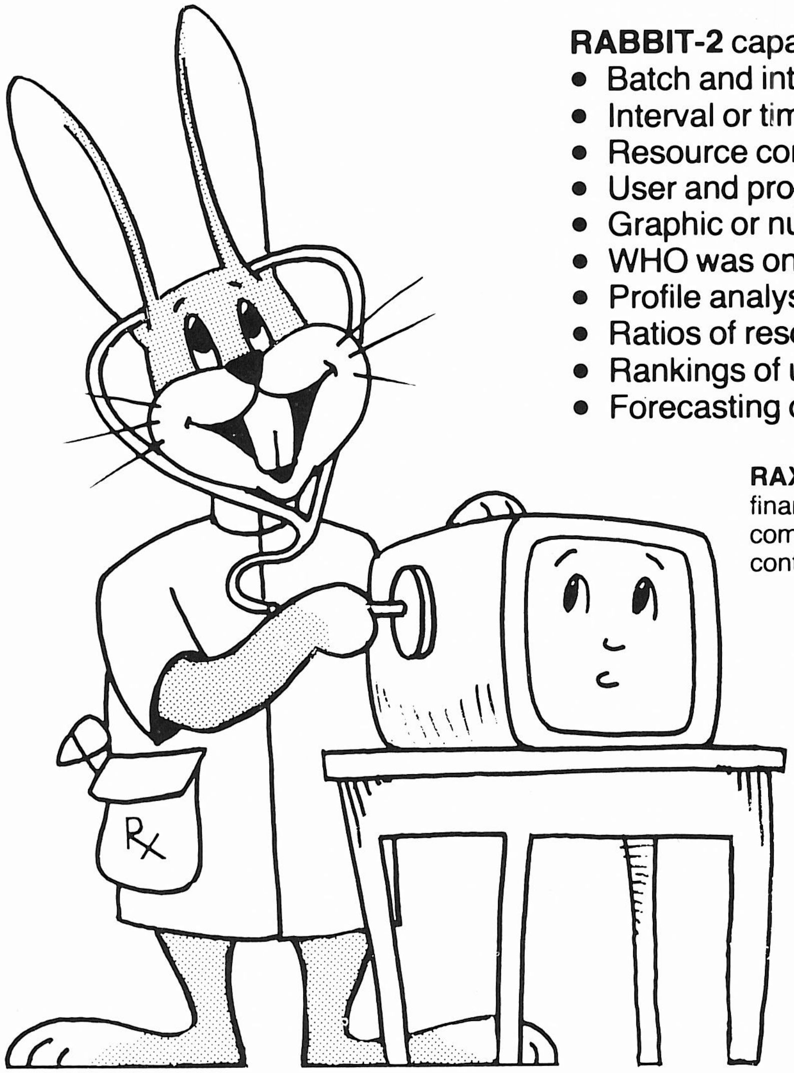
```
2200 ! Next block
2210 ! More dialogue
2500 ! Last block
2560 LAST%=FNINPUT%(30%)                           &
\  ON ITERM.% GOTO 3000,3000,2500,3000,9100        &

3000 T..%=FNINPUTP%("Processing",12%)!             &
\ GOTO 2100                     !Back to first qn. &

9000 !++++SAVE DEFAULT PARAMETERS+++++++++++++++++
9010 T..%= FNINPUTH%(1%)         !Exit Header      &
\ T..%   = FNINPUTC%(2%,"YN")    !Save wanted ?    &
\ GOTO 32700 IF ITERM.% < 0%     !^Z or ^C         &
\ GOTO 9030 IF T..% = 2%         !No               &

9020 T..$ = FNINPUT$(0%)         !File name        &
\ GOTO 32700 IF ITERM.% < 0%     !^Z or ^C         &
\ T..$ = T..$ + "."+PAREXT$ UNLESS                 &
        INSTR(1%, T..$, ".")     !Add extension    &
\ OPEN T..$ FOR OUTPUT AS FILE 9%!                 &
\ PRINT #9%, IDFLT.$(I%)         !                 &
   FOR I% = 0% TO 40%            !Cpy dflt         &
\ CLOSE 9%                       !                 &

9030 ON ITERM.% GOTO 32700, 32700, 2100, 32700     &
                                 !ESC back
9100 !+++Restore ITERM.% after setting dflts++++++ &
  ITERM.%= ITERM%                !Restore trmtr.   &
\ GOTO 2000                      !To 1st header    &

19000 !++++++++++++ ERROR HANDLING ++++++++++++++++ &
        RESUME 19010                               &

19010 GOTO 9000 IF ITERM.%=-11% !^Z, save param.   &
\ IF ITERM.%=-28%               !^C, jump back     &
   THEN IF ERL > 2110 GOTO 2100 !Past first questn &
   ELSE               GOTO 32700!or exit           &

19100 ERROR.%=FNERRMES%(ERR)     !Message for error &
\ GOTO 1120 IF ERL=1120          !Par. file restore &
\ GOTO 9020 IF ERL=9020          !Par. file save   &

19990 GOTO 32700                 !Exit             &

20200 !+++++++++++++++++++++++++++++++++++++++++++ &
! INPUTA  INPUT FUNCTIONS,ARRAY PARAMETERS         &
!                                                  &
!  INPUT functions with text and defaults being    &
! stored in arrays.  ITEXT.$(TEXT%) and IDFLT.$    &
! (TEXT%) hold the prompt text and default value,  &
! TEXT% is the parameter to the functions. The     &
! default is set to the latest value I/P unless    &
! the first character of the default string is a   &
! space. All types of defaults are stored as       &
! character strings.                               &
!                                                  &
! LIST OF FUNCTIONS                                &
!                                                  &
! FNINPUT$(TEXT%)      Input string                &
! FNINPUT(TEXT%)       Input real no. and validate &
! FNINPUT%(TEXT%)      Input intg. no. and validate &
! FNINPUTC%(TEXT%, DFLT$) Input a character string &
! and validate,the list of valid strgs is in DFLT$ &
! the length of string validated is the length of  &
! the default.  The function returns the position  &
! of the input string within DFLT$, ie. 2 if it    &
! is the second valid string.                      &
!                                                  &
! FNINPUTH%(TEXT%)     Print LF and header format. &
! FNINPUTP%(TEXT$,TYPE%) Prints TEXT$ on the       &
! keyboard, TYPE% is the line spacing, the tens is &
! the number of LF,s before the text and the units &
! the number of CR LF after the text.              &
```

SYSTEM PERFORMANCE ANALYSIS FOR VAX AND RSTS/E USERS

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

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

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



RABBIT-2 capabilities include:

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

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

RAXCO INC.

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

```
! FNERRMES%(TYPE%) Prints the system error message &
! corresponding to TYPE%, prints a bell and blank  &
! line before unless 0%. prints a blank line after &
! unless negative.  Returns ABS(TYPE%)             &
!                                                  &
! FNERRTXT%(TEXT$) Prints a bell and blank line,   &
! TEXT$ and a further blank line. Returns -1.      &
!                                                  &
! FNINPUTS$(DFLT$, TYPE%) Called by other input    &
! functions.Function to get a string of characters &
! from the keyboard. The parameters required are   &
! DFLT$ which is the default and TYPE% which is    &
! the type of validation required - see below.     &
!       The function returns a value in ITERM.%    &
! according to the terminator used, the values     &
!       are :-                                     &
!       ITERM.% Terminator      TYPE%   Validation &
!                                                  &
!       1%      <LF> <CR> <NL>  0%      None       &
!       2%      <CR> <LF>       1%      Integer    &
!       3%      <ESC>           2%      Real       &
!       4%      <FF>                               &
!                                                  &
! The function itself is equal to the input string &
! ILEN.% is the length of the input string (with   &
! terminators stripped). If the function is        &
! entered with ITERM.% = 5 then no input is        &
! solicited and the function returns the default.  &
! Used for setting initial val's of user variables &
!                                                  &
! +++++++FNINPUTS$(Default$, Validation%)+++++++++ &
  DEF* FNINPUTS$(DFLT$, TYPE%)        !
20202 FNINPUTS$,T..$ = RIGHT                       &
  (DFLT$, 2% AND ASCII(DFLT$) = 32%)!Strip space   &
                                    !Init default  &
\ IF ITERM.% <> 5% THEN             !Not init      &
   PRINT ITEXT.$(TEXT%);" <";T..$;">";!Print prmpt &
\  ON ERROR GOTO 20215              !Set error     &
\  INPUTLINE T..$                   !Get reply     &
\  PRINT IF POS(0%)                 !Neat          &
\  ITERM.% = ASCII(RIGHT            !              &
          (T..$,LEN(T..$))) AND 7% !Get last chr   &
\  T..$ = CVT$$(T..$,5%)            !Tidy-up       &
\  T..  = VAL(T..$) IF TYPE%        !Get val ?     &
\  T..% = T.. IF TYPE% = 1%         !Get val ?     &
\  GOTO 20217 IF TYPE%=1% AND T..<>T..%!Chck vals  &
\  ITERM.% = 1% UNLESS ITERM.%      !<LF><CR><NL>  &
\  ILEN.%=LEN(T..$)                 !Lngth of I/P  &
\  IF ILEN.% THEN FNINPUTS$ = T..$+ ""!That's it   &
\   IDFLT.$(TEXT%) = CVT$$(T..$,8%) !Strip ldg sp  &
        UNLESS ASCII(DFLT$) = 32%   !Update deflt  &

20210 ON ERROR GOTO 19000   '       !Reset error   &
\ FNEND                                            &

20215 RESUME 20217 IF ERR=52% OR ERR=51% !Ill. no. &
\ ITERM.% = -ERR                    !Return error  &
\ ILEN.%= 0%                        !No input str  &
\ RESUME 20210                      !Jump to end   &

20217 T..% = FNERRTXT%("%Illegal number")!Err.mess &
\ GOTO 20202                        !Start again   &
! ------------END OF FNINPUTS$-------------------- &
```

```
20220 !+++++++FNINPUT(Array element no.)++++++++++ &
  DEF FNINPUT$(TEXT%)           !Input String with &
  = FNINPUTS$(IDFLT.$(TEXT%),0%)!specified defalt  &

20225 DEF FNINPUT%(TEXT%)       !I/P Integer with  &
  =VAL(FNINPUTS$(IDFLT.$(TEXT%), 1%))!specifd dflt &

20230 DEF FNINPUT (TEXT%)       !Input f/p Number  &
  =VAL(FNINPUTS$(IDFLT.$(TEXT%), 2%))!specifd dflt &
! ------------END OF INPUT FUNCTIONS-------------- &

20240 !+++++FNINPUTC%(Array el. no%,valid chr.$)++ &
  DEF FNINPUTC%(TEXT%, DFLT$)!Array el.,valid chrs &

20245 T..$ = IDFLT.$(TEXT%)       !Temp. default   &
\ T..$  = RIGHT(T..$, 2%                           &
         AND  ASCII(T..$)=32%)    !Strip any space &
\ I.DFLT%        = LEN(T..$)      !Lngth to match  &
\ T..$ = CVT$$(LEFT(FNINPUTS$     !Upper case,     &
       (CHR$(32%)+T..$            !no dflt update  &
       , 0%),I.DFLT%),32%)        !len. dflt to mach &
\ T..$  = T..$+SPACE$                              &
     (I.DFLT%-LEN(T..$))          !Pad lngth of dfl &
\ T.., T..% = (-1.+INSTR          !Where is the    &
      (1%, DFLT$, T..$))          !reply, must be on &
           /I.DFLT% +1%           !on a boundry    &
\ GOTO 20250 IF T..%                               &
     AND T..% = T..               !It's there      &
\ T..%  = FNERRTXT%                                &
     ("invalid input")            !Oh no it isn't  &
\ GOTO 20245                      !Try again       &

20250 FNINPUTC% = T..%            !FN=postn of T..$ &
\ IDFLT.$(TEXT%) = T..$ IF ILEN.%  !Load new deflt &
  UNLESS ASCII(IDFLT.$(TEXT%))=32%!If allowed      &
\ FNEND !------END OF FNINPUTC%------------------- &

20270 !+++FNINPUTH%(Array el. no%)++++++++++++++++ &
  DEF FNINPUTH%(TEXT%)                             &
       =FNINPUTP%("***    " + ITEXT.$(TEXT%)      &
       +  "    ***",11%)                           &
! --------END OF FNINPUTH%------------------------ &

20280 !++++FNINPUTP%(Text$,Leading/trailing blnk%) &
  DEF FNINPUTP%(TEXT$, TYPE%)           !Text$,blnk% &
\ PRINT  STRING$(TYPE%/10%,10%);TEXT$;!Ledng blnk  &
\ TYPE% = TYPE% - (TYPE%/10%) * 10%   !Trling LF,s &
\ PRINT STRING$(TYPE%-1%,10%)IF TYPE% !Trlng blnks &
\ FNEND !-----End function FNINPUTP%-------------- &

20290 !+++++++FNERRMES(Errno%)+++++++++++++++++++ &
  DEF FNERRMES%(TYPE%) = FNINPUTP%                 &
   (CHR$(7% AND TYPE%<>0%)        !Print bell      &
    + RIGHT(SYS(CHR$(6%)+CHR$(9%)                  &
    + CHR$(ABS(TYPE%))),3%)       !Err messag      &
    , (10% AND TYPE%<>0%)                          &
    + (2% AND TYPE%>=0%))         !for format      &
    + ABS(TYPE%)                  !err no.         &
! -------END OF FNERRMES%------------------------- &

20295 !+++++++FNERRTXT%(Error text$)+++++++++++++ &
  DEF FNERRTXT%(TEXT$) = FNINPUTP%                 &
  (CHR$(7%)+TEXT$, 12%) - 1%      !Value           &
! -----------END OF FNERRTXT%--------------------- &
! --------END OF INPUT FUNCTIONS----------------- &

32700 !++++++EXIT PROGRAM+++++++++++++++++++++++++ &
        CLOSE 10%                                  &

32767   END
```

APPENDIX C
DIALOG.BAS PROGRAM

Sets up questions and defaults virtual array file.

```
2       !DIALOG
3       !DIALOGUE AND DEFAULT FILE BUILDING PROGRAM
4       !FOR INPUTA
10      EXTEND
110     !Input data is a list giving the element   &
!       number, prompt and default for each        &
!       element required to be set; each item      &
!       must be terminated by ESC or be on a new   &
!       line.  The output is a string virtual      &
!       array with a string length of 64           &
!       the defaults follow all the prompts.       &
!       The input file must have an extension .DIA &
!       and the output has an extension .DIL.      &
!                                                  &
!            Typical input list                    &
!       0                       Element number     &
!       Parameter file          Prompt             &
!       Parameter file default  Default            &
!       1                                          &
!       First prompt                               &
!       First default                              &
!       2                                          &
!       Second prompt                              &
!       Second default                             &
!                                                  &
!       Each element no. can be preceded by blank  &
!       lines or comment lines starting with '!'   &
!                                                  &
!            Output file is compatible with        &
!       a DIM statement of form:-                  &
!       DIM #ch.,ITEXT.$(n) = 64%,                 &
!       INDFLT$(n) = 64%  where n is the array     &
!                         max subscript specified  &

900     ON ERROR GOTO 19000
910     DIM IDFLT.$(100)        !Upto 100 questions
920     DIM #2%,                !Prompt and        &
         ITEXT.$(32767) = 64%   ! default text     &

1100    INPUT "Program name"; FILNAM$
1110    OPEN FILNAMS + ".DIA"                      &
                 FOR INPUT AS FILE 1%
1120    OPEN FILNAMS + ".DIL"                      &
                 FOR OUTPUT AS FILE 2%
1130    INPUT " Array max subscript", ARRSIZ%
1140    IDFLT.$(I%), ITLXT.$(I%) = ""              &
        FOR I% = 0% TO ARRSIZ%  !Null arrays
1220    INPUT #1%,I$
1225    I$ = CVTS$(I$,2%)       !Strip spaces      &
\       GOTO 1220 UNLESS        !Test for blank    &
        LEN(I$) AND             !or comment line   &
                  ASCII(I$) <> 33%                 &
\       I%=VAL(I$)                                 &
\       INPUT LINE #1%, T..$                       &
\       ITEXT.$(I%) =           !Strip term.       &
                 CVTS$(T..$,5%) !and parity        &
\       INPUT LINE #1%,T..$ .   !Default           &
\       IDFLT.$(I%) =           !Strip term.       &
                 CVTS$(T..$,5%) !and parity
1270    GOTO 1220               !Input next        &

1300    ARRSIZ% = ARRSIZ% - 1%  !Start of default  &
                                !in array
1310    ITEXT.$(ARRSIZ% + I%) =                    &
                IDFLT.$(I%) FOR I% = 0% TO ARRSIZ%
1400    GOTO 32700                                 &
19000   !ERROR HANDLING
19100   RESUME 1300 IF ERR = 11% AND ERL = 1220    &
\       PRINT "Last element was :-";I$             &

32700   CLOSE 1%,2%                                &
\       ON ERROR GOTO 0                            &

32767   END
```

APPENDX D
DELCOM.TEC TECO MACRO

DELetes COMments from BASIC+ programs.

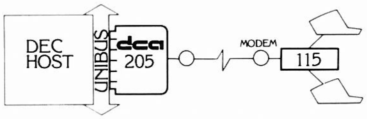
```
!DELCOM.TEC
Deletes comments and trailing space/tab from BASIC+
extend mode programs .Call from TECO by EI DELCOM$$!
:@S/10  EXTEND/"U @A/
NO LINE:-
10      EXTEND
COMMENTS NOT DELETED.
/@O/END/' L OUC OUD!Cont line and del. comment flgs!
@UQ/"'/
!NEXT! .US  .U2        !Reg for start of line and
                                  2nd quote, LF flag!
L -1A-10"E -1UL -1UA R!Befor LF,store LF & cntn flg!
  -1A-13"E  OUL  OUA R'!Befor CR, clr LF & cntn flg!
<2,-1:@S/^ES/ "U O;' -D> 2,-1:@S/&/ "S -1UA R'
             !Delete trailing space back past &, flag!
 .UE                                       !End line!
QD "N OL @O/COM/'
<Q2 US  QE U1  QE U2  QE U3  QS J
 !Load 1st, & 2nd ", and shriek pointers,start line!
Q,:@S/^EGQ/ "S .U1'  Q,:@S/^EGQ/ "S .U2'  QS J
                             !Find ",s, back to start!
Q,:@S/!/"S .-1U3' Q1-QE; Q1-Q3-1;>
               !Next iteration if no shriek before "!
Q3 J <2,-1:@S/^ES/ "U O;' -D QE-1 UE > .U3
                    !Back past space & tabs from end!
::@S/!/"S (Q,:@S/!/   QL) "N -1 UD'' Q3J
    !store continuation flag if 2 shriek or LF only!
!COM!QA"E OUD'
 .,QE K .UE OL QC "E  "N
 .-QE"E QA"E @I/!/''''
OL .-QE "E QA "N K @O/SAME/''
QAUC                                 !Flag cont. line!
L !SAME! Z-. "E P' Z "G @O/NEXT/'
               !next page if end, next line if more!
EX
!END!C$$
```

Don't connect remote terminals to your Unibus*-based computer

UNTIL YOU'VE READ ABOUT DCA'S SYSTEM 205 UNIBUS-INTERFACE STATISTICAL MULTIPLEXOR.

A DZ11 emulator and stat mux in one.
That's what DCA's System 205 gives you. In fact, a single DCA 205 acts as a host-end multiplexor for PDP-11, VAX or 2020 computers while eliminating up to 16 DZ11 modules!

Supports up to 128 remote terminals.
With a 205 and another DCA statistical multiplexor at the remote end, you can support up to 128 remote asynchronous terminals. Any DZ11's already in use for remote-site service can be transferred to local service.



An easily-affordable point-to-point datacomm network.

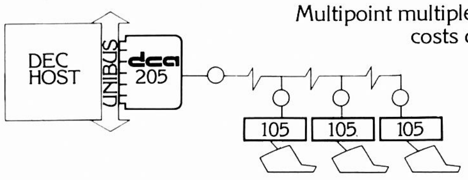
Requires just a single Unibus slot.
That's right, the 205 only uses one Unibus hex slot, freeing up from three to fifteen additional slots. Costly and unsightly cables are eliminated, too.

Superb response time.
The 205 is, in effect, a DZ11 and statistical multiplexor in one integrated unit. So the character delay that normally occurs between a DZ11 and a multiplexor is eliminated. Result: Your terminal users will enjoy a crisper echo.

Statistical multiplexing.
Our 205 gives you all the benefits of DCA's statistical multiplexing, including full ARQ error control that maintains data integrity, network transparency, reduced CPU overhead — and more.

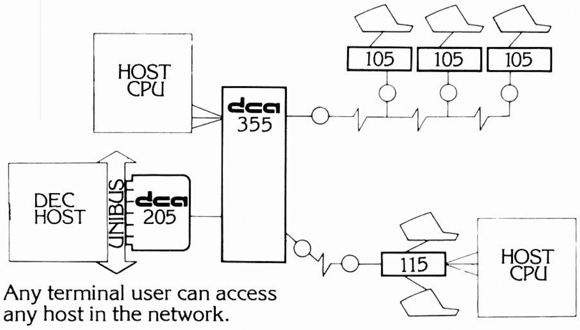
Multipoint multiplexing.
This isn't a promise, it's a fact: DCA's customers have been multidropping several remote terminal locations to a single telephone line since 1979. And multipoint multiplexing with the 205 lets you enjoy the lowest cost communications possible from Unibus-based DEC computers.

Multipoint multiplexing lowers costs dramatically.



Reliable, proven performance.
Like our other statistical multiplexors, the 205 is a field-proven, reliable system. In 1974, DCA introduced statistical multiplexing — we've been the technological leader ever since!

Unparalleled growth potential.
For networks that are bound to grow, DCA is the clear-cut choice. Because complete system compatibility lets you start with a 205 today, and grow into a larger, more powerful network tomorrow. And DCA users enjoy the lowest-cost network growth in the industry!



Any terminal user can access any host in the network.

Excellent cost efficiency.
There is no better value for DEC users — the 205 will dramatically reduce your datacomm costs while providing high throughput and reliable operation. Interested? Send in the coupon or call DCA at 404/448-1400 today.

□ Send me more information on DCA's System 205.
□ Please have a sales representative call.

Name__________

Title__________

Company__________

Address__________

City______ State______ Zip______

Telephone__________
(Area) (Number) (Ext.)

dca
ENGINEERED
TO EXPAND

Digital Communications Associates, Inc.
Dept. S-2 135 Technology Park/Atlanta
Norcross, GA 30092

* Unibus, DEC, DZ11, PDP-11, VAX and 2020 are trademarks of Digital Equipment Corp.

THE "SLAM" SYSTEM

Security Lock and Monitor System

By I.F. Dawson, Supervisor, Hardware/Software Support, Technical Services Division
MacMillan Bloedel Limited, Vancouver, B.C., Canada

ABSTRACT

A Security Door Monitoring and Control System interfacing a SCHLAGE Model 708 Door Controller with a PDP-11 running under RSTS/E. Identification badges are presented at access control points and the system unlocks the door at the point after validating the identification number, door location and time of day.

1. SYSTEM OVERVIEW

A. General Description

This computer system consists of a set of programs designed to execute on a PDP-11 computer running under the RSTS/E Operating System. It was written in BASIC-PLUS Extend Mode and has also been converted to BASIC-PLUS 2.

A Model 708 Door Controller, manufactured by Schlage Electronics, 1135 East Arques Avenue, Sunnyvale, California, is connected to a PDP-11 on an asynchronous terminal line. The Model 708 can have up to 8 sensors connected to it, which read command keycode data from identification badges as they are presented to the sensors. The Model 708 transmits sensor location and keycode data received from the sensors to the central computer, which in turn transmits lock control commands to the Model 708, which are interpreted to determine the lock/unlock action to be taken at the sensor in question.

The control program runs under RSTS/E in a normal timeshare mode, but may be locked in memory and/or have a higher than normal priority, depending on the individual installation's wishes. Of course, the requirement to swap when not locked in memory, and the priority relative to other time-share users on the system, will determine the speed with which a door will be unlocked when an identification badge is presented to a sensor.

In addition to the main control program, there are other programs which permit adding, changing and deleting entries on a master file of valid badge holders, listing of the master file, creating updating and printing of the system control file, and listing and condensing of the activity log file.

The main control program may be started automatically by commands in the RSTS/E start-up control file, and stops automatically when the RSTS/E "Shutup" utility program is run, or when the number of logins is reduced to one.

B. Method of Operation

When the main program is in normal operation, a badge holder approaches a locked door and holds his identification badge within several inches of the sensor, which may be mounted externally on a wall or window, or may be imbedded within a wall out of sight. The Model 708 sweeps each sensor every few milliseconds, and receives a signal from a badge when it comes within proper proximity of the sensor. The signal is sent to the Model 708, which then composes a message in a specific form, identifying the sensor location and the identification recorded within the badge. The PDP-11 program at all times has a "read" outstanding on a hard-wired line connected to the Model 708. When a signal arrives, the program determines that:

- the badge number appears in the master file of valid badges;
- the badge number is authorized at the location of the sensor;
- the badge number is permitted access at this particular time of day.

After these checks have all been satisfied, a signal is sent to the Model 708 instructing it to unlock the door at the location of the sensor. In normal operation, on a PDP-11/70 with 256KW of memory, using an RP04 system pack and an RP06 pack to hold the data files, with 30-35 jobs running, with the program locked in memory and running at priority zero, the door is unlocked within 2 seconds of the badge being presented to the sensor. If the program is not locked in memory, response time may be as long as 3-4 seconds, depending on system activity.

If the badge number being checked fails to pass all the tests mentioned above, a different signal is sent to the Model 708 instructing it to disable the sensor in question for approximately 6 to 8 seconds. After this time, that sensor is re-enabled by the Model 708, and normal scanning resumes. In this way, an invalid badge will not continually send invalid requests to the program, which could cause it to monopolize the PDP-11 CPU.

If the PDP-11 program fails to respond to the signal sent by the Model 708 within approximately 7 seconds, for whatever reason, (ie: CPU down, program failure or heavy CPU load), the Model 708 will go into "Fail Soft" mode. In this mode, the Model 708 will send a "Timed Unlock" command (described later) to any sensor at which a badge has been presented if it is a member of a unique set of badges identified to the 708 as belonging to this installation. No reference is made to the second part of the identification on the badge, which is unique to each badge, nor is any reference made to the time of day or sensor location. This capability may be enabled or disabled for each sensor independently. Therefore, access to very critical areas may be totally prohibited when the system is in the "Fail Soft" mode, in which case access would only be possible using a special, non-duplicatable key.

2. HARDWARE/ SOFTWARE

This set of programs operates on a PDP-11 computer, running under the RSTS/E Operating System, (V7.0 or later). The programs are written in both BASIC-PLUS and BASIC-PLUS 2 (V1.6).

The BASIC-PLUS version starts executing at a size of 13KW.

The BASIC-PLUS 2 version is task built against the 4K word "BP2COM" Run-Time System, and starts executing at 19K words.

The Model 708 is connected to the PDP-11 on a dedicated, full-duplex asynchronous line on either a DH11 or DZ11. The line speed may be up to 9600 baud.

3. SLAM SYSTEM DETAILS

This computer system is designed to permit an installation to specify that certain identification badge holders will be permitted to enter secure areas at identified times of the day. Each badge may be given access to up to 8 doors. Access to each door may be restricted to any time range between 00:01 A.M. and 24:00 P.M., independently for each badge. All accesses to certain specified doors are logged for later reporting, and accesses to other doors are only logged between specified time periods.

Each door is designated by the installation as either "critical or "non-critical". When a badge is presented to a sensor at a "critical" door, and proper identification is confirmed, the door is unlocked for a definite period of time, (settable by switches on the Model 708 for each sensor, at either 1, 4, 10, 15, 20, 30, 45 or 60 seconds). After that time, the Model 708 will relock the door. This is referred to as a "Timed Unlock". If the door is "non-critical," the door will be unlocked for an indefinite time after a certain time of day, (installation settable), such as 8:30 A.M. This is referred to as an "Indefinite Unlock." Thereafter, the sensor will not recognize a badge in its proximity until after the Model 708 has been instructed to relock the door by the control program. This occurs at some time also set by the installation, (ie: 5:00 P.M.) and is known as a "Definite Relock."

In addition, all signals received from "critical" doors are logged to a disk file which identifies the sensor location, badge number, date, time and whether the access attempt was successful or not. This information is only recorded for "non-critical" doors before the indefinite unlock is done for the day, or after the relock has been done for the night. Finally, access attempts at non-critical doors on weekends and on up to 10 installation-identified holidays are treated as if they were access attempts at critical doors. That is, the "unlock" is for a timed period, and all access attempts, successful and unsuccessful, are logged.

The access control program may be temporarily "suspended" by the computer operator from the system console only. This is a simple procedure, accomplished by "attaching" to the "detached" access control program and typing CTRL/C.

which interrupts the running program. At this time, the operator has the option of "suspending" operation of the system for five minutes, or shutting the system down entirely. The "suspend" option has been designed to allow for a future enhancement which will monitor sensors on doors which do not have badge readers, but merely sense when certain doors have been left open. This could apply to a fire exit for example. The program would report an open fire door to a central secuirty office which is manned 24 hours a day. In order to avoid false alarms when it is necessary to leave a monitored door open for a few minutes (ie: janitorial access), this facility will permit the program to ignore any warning signals sent by the monitored door sensor during the time the program is susppended. The program automatically resumes normal operation five minutes after being suspended by the computer operator. Activity at other doors during "suspend" operation will put the Model 708 into "Fail Soft" mode, which will be automatically reset by the SLAM program when the 5 minutes have expired.

Another program is used to add, change and delete records on an indexed master file. When a badge has been lost or stolen, or when an employee leaves the company or is no longer permitted access to a secure area, his record on the master file may be flagged as inactive. This "inactive" flag may be reset to active if the situation returns to normal, or the master record may be removed entirely, depending on the wishes of the company's Security Administrator. At present, the indexed master file is capable of holding 512 identification badge numbers. In addition, the Security Administrator may request a listing of the present contents of the security master file at any time. The Data Caching facility of RSTS/E V7.0 is utilized on this file to assist in providing good response times at the sensors.

Access log files are maintained by month, and reports may be requested for any day or range of days in any month, including the current month and day, for the following activities:

- all log file records;
- all invalid access attempts made;
- all access attempts, (successful or unsuccessful) at any designated door or doors;
- all attempts (successful or unsuccessful) by any badge or badges identified to the system;
- all 'exceptional activities' such as:

  —system startups
  —system shutdowns
  —'non-critical' door indefinite unlocks
  —'non-critical' door relocks
  —Model 708 'Fail-Soft' mode, and recovery
  —program 'suspend' and 'resume'
  —'chained restart'
  —invalid access attempts.

The only restriction on printing log file records is that records of the most recent activity are not actually written to disk until the disk buffer in memory is full. Depending on the system activity, this may represent some period of time. (Each block of the log file contains 39 log records). If a full and complete list of the current day's activity is required immediately, the system may be shutdown and re-started, since this action will cause the last (partial) log file buffer to be written to disk.

When the system starts up again, it appends new log data to the end of the current month's log file. If this activity occurs many times during the month, the log file may contain many partially filled data blocks, making the file larger than is actually necessary. A program is supplied which will "condense" any log file (except that for the current month), in order to eliminate this wasted disk space.

There is also a program which is run when the system is installed, and is used to specify installation-unique information, such as:

- the "keyboard number" to which the Model 708 is attached;
- the time of day after which the non-critical doors are to be unlocked;
- the time of day after which the non-critical doors are to be relocked;
- the "critical" door numbers (if any);
- the Julian day numbers of up to 10 days which are designated "holidays" and are to be treated the same as weekends for the purpose of treating "non-critical" doors as "critical" doors;
- the Project/Programmer number and logical disk name where system files and programs are to be stored;
- a RSTS/E Keyboard to which start-up and error messages may be broadcast (usually KP0:);
- an indication of whether the control program should "lock" itself into memory or not.

All of this information is contained in a control file which is stored on a disk and under an account specified by the user.

Network Processing on RSTS/E

By Val Skalabrin, President, Data Node, Inc.

Captain Grace Hopper said it best.

"When the log is too big for your elephant, don't breed a bigger elephant. Use two elephants."

That concept of parallel processes is at the heart of Network Processing. The definition of network processing and a description of its future, however, requires some groundwork in how systems like RSTS/E operate and what affects total performance.

RSTS/E is a superb system — flexible, powerful, stable, etc.

It wasn't always that way.

RSTS/E began its life as the creation of Nathan Teicholtz at DEC. Developed by Nathan essentially on his own initiative with system time borrowed as he could, this new thing first appeared as a BASIC only system.

DEC's main competition in that market area was then the HP 2000 system. As a matter of fact, the HP2000 was beating the bejeemies out of DEC in multi-terminal sales. We had the TSS/8, Edusystem 20, and Central Engineering was developing a multi-user BASIC (no disk files) on the 11.

Nathan's new BASIC was BASIC-PLUS; magnificent; his operating system was logically device independent (the Resource Sharing of RSts).

We named it BTSS for Basic TimeSharing System. Rick Merril preferred an all FOCAL system. A trademark search, however, showed BTSS already in use. Aren't we all glad of that.

David Ahl, I believe, came up with RSTS for Resource Sharing Timesharing System.

Notice an important thing about DEC.

DEC makes great computers.

But the system the market really wants came from a dedicated individual with an idea — not Central Engineering.

That's primarily a very real lack of marketing ... marketing expertise to outline the product and marketing's strength within DEC to tell Central Engineering what to do. That was in the old days. Of Course.

The first RSTS (no E) systems running on the 11/20 used so much core for monitor that only one job slot was left — and the software was new and being patched and expanded constantly.

Crash, crash.

Ted Sarbin implemented a famous "whoops" patch to RSTS at Delaware Schools that would restart the system immediately upon each crash. The users would suddenly see "Please say HELLO" as the first indication of a crash.

RSTS became RSTS/E and eventually stable with the introduction of the 11/40 and 11/45 and extended memory.

There has since been an enormous amount of development of RSTS/E by Digital (if enough people buy them, DEC notices) with a subsequent attention to performance.

Performance has essentially come with better hardware and some system changes, but primarily programmer education. A poor programmer can make RSTS/E run slower than the winner of the watermelon eating contest.

(It must be pointed out that RSTS/E is still head and shoulders above any competitor's similar system.)

This programmer education we all are familiar with and it's the Record I/O, integers only, contiguous and clustersize, etc., programming "techniques."

Very similar to the tailor who solves your ill fitting suit problem by saying: "Lean to the left, scrunch your head onto your right shoulder, shorten you left arm by an inch ... there! See what a fit."

There is a performance limiting factor built into the type of computer system RSTS/E is, and all our techniques are attempts to extend that limit a little bit.

The limiting factor built into RSTS/E (and all like systems) is also its strongest selling point — it is a do-many-jobs-at-once system.

The limit appears because as we attempt to run more and more jobs on the system the jobs begin to "interfere" with each other and require more and more system management.

A poorly written program "interferes" more by consuming more compute time than ideal and being larger than ideal or creating more "garbage" than ideal which in turn brings in the system management utilities more often for swapping and cleanup.

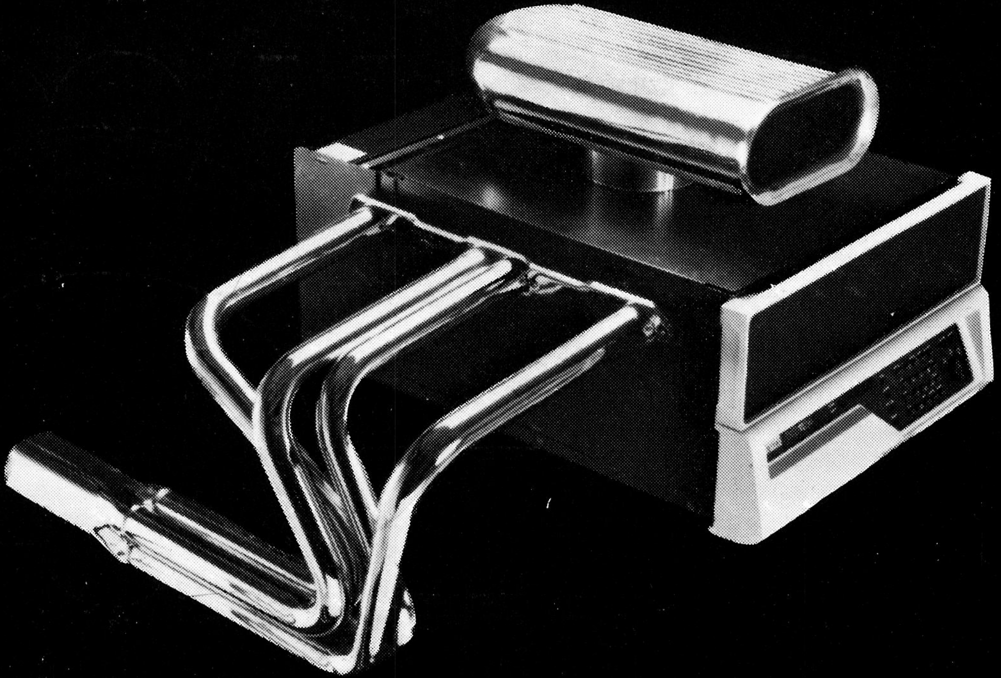
Good (lean, scrunch, shorten) programming practices can delay the onset of this limit, but it will show up.

In a traditional system, Network Processing is based upon placing a microcomputer in the terminal, downline loading (or from local storage) the program **and splitting** the normal program functions between the mainframe (RSTS/E) and the microcomputer so that:

1. the central system can do its central thirg for many, many users simultaneously
2. the user gets responsive, good performance
3. the total system performance goes up by a factor of 5 to 10 within RSTS/E's job max limit.

We come now to heresy, blasphemy and odorous statements.

RSTS/E SUPER CHARGER



- **ZERO TO 60** data node's add-on kit allows any RSTS/E system to support up to sixty terminals with excellent response times.
- **DROP IN KIT** completely transparent to existing applications, the data node add-on kit includes Node Central software providing data base and network interfacing for RSTS/E, and plug-in-and-go microcomputer boards for your VT100s (or ours).
- **HIGHER PERFORMANCE** you can offload your programs onto the microcomputer in the terminal and experience up to an 8-to-1 reduction in memory requirements per job on the PDP-11.
- **BETTER MILEAGE** the superior data management of Node Central, the super quick data retrieval and sorting,and the offloading of computation onto the microcomputer terminals allows the data node boosted system to achieve transaction rates of as much as 10,000 to 20,000 transactions per hour.
- **LOW COST** the small monthly license fee of Node Central, and the low cost the VT100 microcomputer boards, combined with the much higher transaction rates of the data node boosted system are much more cost effective than the high cost and turmoil of going to a larger system (if you can).
- **TEST DRIVE** our special introductory offer gives you the opportunity to try the data node add-on kit with a 30 day, money back guarantee.

data node

FOR DETAILS on the data node add-on kit, our special introductory offer, and the full blown data node I system, fill out this coupon, clip it to your letterhead, and mail to:

Marketing Manager, Service Products Group
Data Node, inc.
432 Toyama Drive Sunnyvale, Ca. 94086

NAME ____________________ TITLE ____________________

FIRM ____________________ ADDRESS ____________________

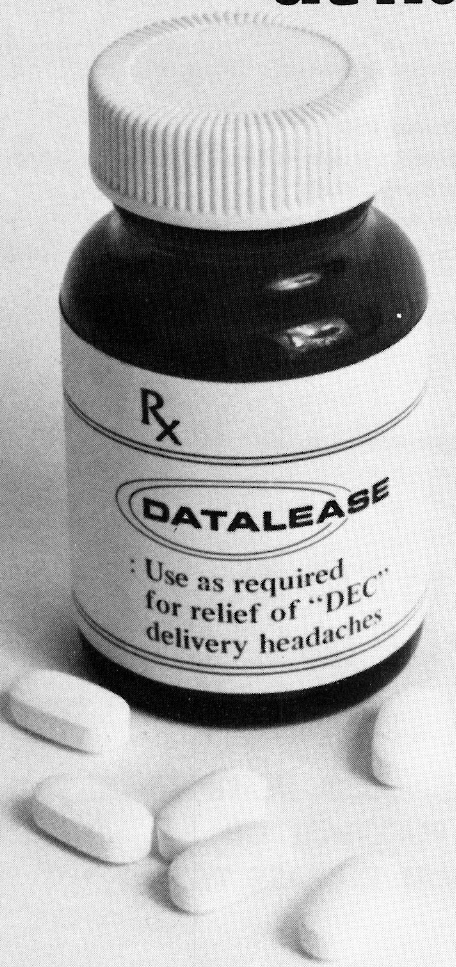
CITY ____________________ STATE __________ ZIP __________

RSTS/E and PDP-11 are registered trademarks of Digital Equipment Corporation

DEC RM03*& RM05*

Disk Capacity Headaches

Why wait for the RM03 and RM05? When we can deliver the same drive at half the price NOW!



COMPARE DELIVERY

DATALEASE MEMORY SYSTEMS can deliver and have your disk sub-system on-line in less time than it takes DEC* to process an order. (Typically within 30 days.)

COMPATIBILITY

DATALEASE MEMORY SYSTEMS offers disk sub-systems that emulate the RM02, RM03, RP04, RP05, and RP06. The sub-systems are completely software transparent and best of all will run standard diagnostics.

NATIONWIDE SERVICE & SUPPORT

DATALEASE MEMORY SYSTEMS provides installation and maintenance nationwide through the Engineering Services Division of CDC. Our own highly experienced systems analysts supervise every installation and do not step out of the picture until you are completely satisfied.

For Fast Relief . . .
Call toll free 800-854-0350

DATALEASE R

DATALEASE MEMORY SYSTEMS
2770 East Regal Park Drive
Anaheim, California 92806
In California (714) 632-6986

***RM03 & RM05 are registered trademarks of Digital Equipment Corporation.**

There is a funny set of traditions and code of conduct in the community of computer people (you and me).

We do things this way because that's the way they are.

Why daddy?

Because the Great Programmer made them that way.

How do we know that, daddy?

That's how they're written in the Book of Structured Venn Diagrams and Bit Patterns.

But, daddy —

Shush child.

We (you and me) are a funny lot.

We are very technical, but we take as faith that the early designs of computers **and their legacy** are good, necessary or inevitable and rarely do we ever take a child's unprejudiced view towards them.

We have central computers — big muscular brutes — many (some at least) terminals.

But what really **has** to be central?

Data.

Data?

Data and some data services.

Some data services?

Well, a lot of data services.

In Network Processing at Data Node we've split a program-plus-operating-system into two parts:

1. **Node Central** — a set of run-time systems and utilities on RSTS/E that perform all the data management functionality a program normally requires (macro ISAM, macro sort, macro communications, dictionary DBMS, etc.)
2. **Node Basic** — a much extended version of Microsoft Basic with VT100 screen handling verbs and Node Protocol verbs built into the language so you may OPEN FILE (local) or OPEN NODE:FILE (host), locate, lock, unlock **logical** records, perform relational retrievals, sort, etc.

If there is interest in the reader community for greater detail on how the split is made, please write the editor and say so. We're willing to give away some of the ten man years spent so far.

For those of you planning to do your own down-line loading, etc., with RSTS/E, save a buncha problems and set:

STALL
XON
FORM
TAB
NO FILL
NO DELIMITER

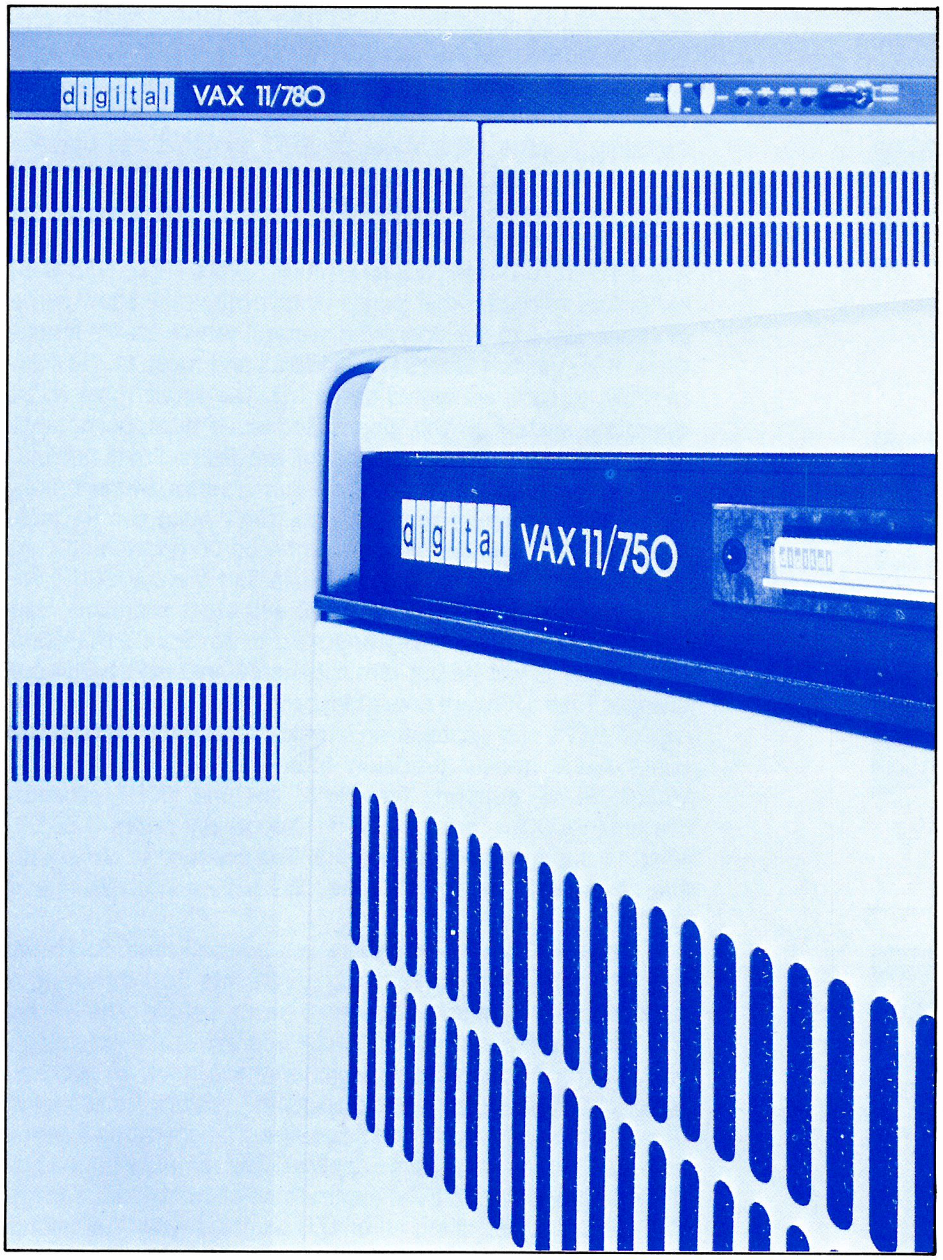
Love from Data Node.

About the author: Val Skalabrin is president of Data Node, Inc., and admits in the past to:

Working for and loving DEC
Founding the old Digital Equipment Business Users Group in DECUS
Authoring Giant/8 and Giant/11
Introducing "Little Red Hen" at the Willow Pond kitchen
Gandy dancing on the Alaska Railroad
Being old
Not going to DECUS in San Diego.

The VAX-SCENE

Number 1 (RSTS PROFESSIONAL, Vol. 2, No. 4) December 1980

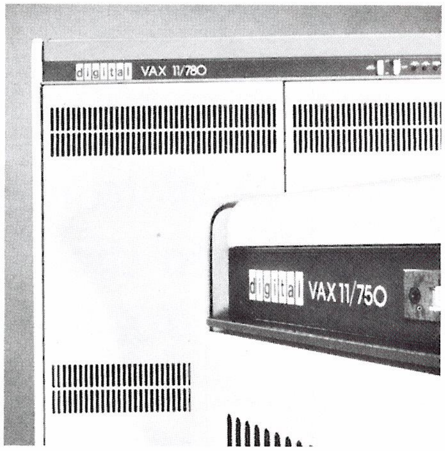


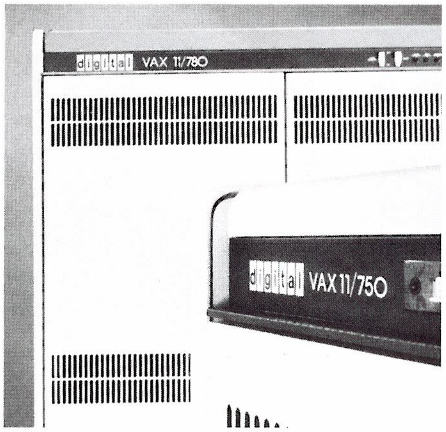
INSIDE:

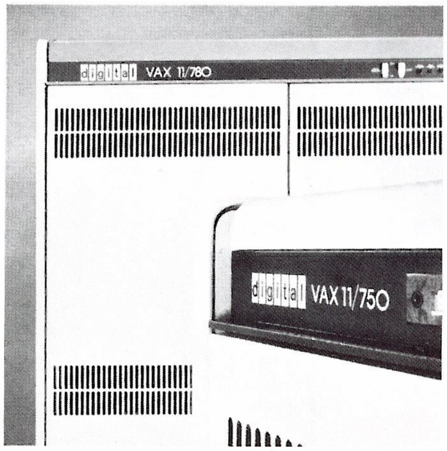
□ Welcome to the VAX-Scene

□ WHAT IS VAX

Welcome to the VAX-Scene!







Welcome to the VAX-SCENE! This is the first in a continuing and expanding section on VAX systems. We have thought long and hard about this section and how it could or should fit into the context of the RSTS PROFESSIONAL. VAX is here to stay, and we believe that RSTS is here to stay also; in fact these two systems will be the cornerstone of DEC commercial efforts in the 80's and maybe beyond. Why is VAX here? Where does RSTS fit? We don't have all the answers but here are some guesses formed after long hours of thinking, and many probing questions asked of DEC and other people.

There is clearly a limit in the amount of operating system that can fit into the 16 bit addressing space. Note that I consider the limiting factor here to be the operating system, NOT the application programming space. Most of us have run out of room (MAX MEMORY EXCEEDED AT LINE 3404) at one time or another, but if you are not using RMS it should be controllable. Ah! RMS. Perhaps that also doesn't fit under the 16 bit architecture; is it the operating system? Yes and no. Certainly a large percentage of RSTS systems can operate very happily without RMS (they did for year!). RMS can and does run under RSTS but let me take a bold step and suggest that if you realy need those kind of data management services a 16 bit computer will be limiting. There are of course alternatives to RMS; other pages of this magazine show some of them. Back to the operating system; where are its limitations. It supports 11/23's (??), 11/34's and most 11/44's admirably. In fact, we would argue it is the single most loved operating system in the world. But when it supports large 11/70's we begin to bump some of the limits: Small buffers, CPU bound managing large (\1MB) memories, 96 port monitors too big, unable to Gen Stats, can't build for 64 jobs, etc. These are monitor limitations based on addressing constraints of the 16 bit architecture. In San Diego (DECUS) we discussed how to get around these and other problems, and the answers are very tricky and hard to do. Should that kind of software effort be put into this hardware? Isn't hardware cheaper than software now? Maybe it is realistic to define the top of RSTS and go back and make the middle better; we could have: double precision integers, CALL statements, BASIC PLUS support for RMS, in line SORT, Xecute statements (X$="print 10/3" ® Xecute X$ prints 3.3333), KMC-11 support, and much more. The question is: Where do the developers spend their time? We have a Wish list—let's get to it.

Once we believe that there is a natural limit to 16 bit machines we can move on to . . . . . 32 bits thus removing a large number of restraints forced on us before. The 32 bit machine is of course VAX (Virtual Address space extended). Now the operating system designers have room to operate. Interestingly there is only one operating system for the VAX opposed to several on the 11/xx series. The operating system now has to satisfy real time applications, compute bound applications, and commercial applications all at the same time. The complexity of doing all of this explains why it is taking years of development time for the VAX/VMS (Virtual Memory System) operating system to mature.

Well there you have it. And here it is. VAX. You will need to know more about it and we'll try to bring it to you, so . . . . heeeerrrreeee'ssss VAX.

WHAT IS VAX

There are two Vax processors currently announced. They are the VAX 11/780 and the VAX 11/750. The VAX 11/780 was the first VAX announced and consists of a 32 bit processor, console, main memory, cache memory, synchronous Backplane Interconnect, one (up to 3) unibus adapters, up to 4 massbus adapters, and optional Floating point Accelerator.

The VAX 11/780 console is an LSI-11 computer with 16KB of memory and an 8K ROM which contains diagnostics, Boot, and console routines. Remote diagnosis via the RDC is also accomplished when a modem and DAA are connected to the console. Console commands control the "switches" of the VAX. The console also has its own floppy disk subsystem.

The main bus of the VAX is the Synchronous Backplane Interconnect (SBI). This is the high speed bus of the VAX and the Unibus, Massbus, and memory adapters all connect to this bus (Fig 1). The SBI has a cycle time of 200 nanoseconds and can transfer 32 bits each cycle. Transfers use two cycles to actually transfer 64 bits at a time. This bus is really a time-division multiplexor between all devices attached to it. Additionally it will handle and arbitrate priorities (one device can go first). In some respects this is the VAX unibus and continues the cycle from the PDP-8's Omnibus, the PDP-11's Unibus, and now the VAX SBI.

The main Memory of the 11/780 is MOS RAM's with a cycle time of 600 nanoseconds. Up to 2 Memory controllers can be attached to the SBI each accessing 4MB of memory for a total system memory of 8MB. Data is fetched from main memory 64 bits at a time (two SBI cycles) and cached in the processors internal memory systems. This includes a main memory cache, an address translation buffer, and an instruction lookahead buffer.

There is an 8KB write through memory cache. The cache is similar to the cache available for the 11/34, and the 11/44. Cache hit rate is reported to be >95%.

In order to maintain peripheral compatibility with the PDP-11 line the VAX 11/780 utiltizes adapters which interface with the SBI. The Unibus Interface connects all devices other than the high speed disk drives and magnetic tapes. All standard Unibus devices can plug into the unibus adapter. The adapter enables the processor to read and/or write the peripheral controller registers. The adapter has an address translation map which translates the 16 bit addresses to a 30 bit memory connect address. The map provides direct access to system memory for NPR (Direct Memory Access; DMA) devices. In order to make efficient use of the memory interconnect bandwidth, the Unibus adapter provides buffered direct memory access data paths for up to 15 nonprocessor request (NPR) devices. Each of these channels has a 64 bit buffer for holding 4 16 bit transfers to and from any unibus device. The result is that only one memory interconnect transfer (64 bits) is required for every four unibus transfers; making it much more efficient than the standard PDP-11 unibus. The maximum transfer rate through this bus is 1.35 million bytes per second (the internal bus of the 11/780 can handle 13+million bytes per second, the 11/750 about 5 million).

For higher speed transfers and for (again!) compatibility, the 11/780 can have up to four massbus adapters. Each massbus adapter includes a 32 byte silo data buffer. Data are assembled into 64 bit quadwords to make efficient use of the SBI bandwidth.

For faster floating point calculations there is an optional floating point accelerator. Note that this is an enhancement, not an addition; no software needs to be changed when adding or subtracting this option. This device actually increases the speed of several floating operations and the 32 bit integer multiply.

The new VAX 11/750 is architectually different. The console inlcudes a TU58 tape cartridge instead of a floppy disk, it supports only only one memory controller limiting memory to 2MB, one UNIBUS is standard (no more can be added), and up to three MASSBUS adapters can be fitted (all are optional . . . a UNIBUS ONLY VAX?). The actual bandwidth of the SBI-Memory is about 5 million bytes per second, less than half the 11/780 and about equal to the 11/70.

The real striking difference between the two machines is in the way they are built and housed. The 11/750 uses the new LSI gate array chips which pack the logic closer than DEC has before. The 11/750 exists on fewer boards than the (older) 11/70, and shows that the reduction in size, and power consumption is continuing. When we pack more into less we can expect manufacturing economies (the machine will cost less) and more reliability (it will cost less to maintain). All of this makes us continue to wonder why the 5 year old 11/70 isn't restated in newer and cheaper logic (maybe this new gate array LSI stuff will be the. . .).

All of what we have described here is a winner. The machine is fast, neat, expandable, and built right. The Hardware is state of the art. Deliveries are standard DEC (long), but the kind of manufacturing that the new technology allows should help bring that to reasonable levels. The key to this system (VAX) is now in the software. What does VMS (the only Vax operating system) do? How does it do it? Is it a sub(Super)set of RSTS or RSX or (as Anton thinks) RT-11? Where is it going? Who is doing what, with which, and to whom? In the next issue of the VAX-SCENE we will begin to report on what is happening with VMS.

Editors note: This is the first of our VAX-SCENE sections. We expect it to grow — how much and into what, we don't yet know. Most of that will depend on the support of the user community; how many of you want to know more, and how many of you can tell us more. We are looking forward to this new and continuing feature.

A DEFAULT COMMAND FOR TTYSET

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

This article describes the implementation of a new command for the RSTS/E (version 7.0) utility TTYSET. The command, DEFAULT, will allow a system manager to define a "default" configuration for any desired keyboard line, and permit a user to establish that default by typing SET DEFAULT. The command behaves as any other TTYSET command or macro; a privileged user can type set KBn:;DEFAULT to establish default conditions for KBn:.

Overview

The distinction between the DEFAULT command and any of the TTYSET macros lies primarily in usage. For example, a user sitting at any terminal may issue the VT100 macro to assign his keyboard the attributes of a VT-100 type terminal. Such an assignment would be inappropriate if his terminal did not support all of the VT-100's features. (Consider what happens when < RUBOUT > is typed at a hard-copy terminal with the SCOPE command in effect). That same user, issuing a DEFAULT command will establish only those attributes defined as "normal" by the system manager. The user need never know (or care) what those attributes are, or anything else about his terminal's characteristics. TTYSET macros define attrributes based on terminal types; DEFAULT defines attributes based on keyboard lines.

Default information for each keyboard line is maintained in the ASCII file 'SY:[1,2]TTYSET.DFL', and can be created or updated using any standard text editor (TECO, EDT, etc.). The DEFAULT file consists of one or more lines, each defining the defaults of a specific keyboard. The format of a line in this file is

```
kb#, command [;command] . . . < cr >
```

For example, the line

```
8,LA36;WIDTH 80
```

would identify KB8: as an LA-36 type terminal, restricting it to a width of 80 columns. A user issuing the SET DEFAULT command for KB8: would perform exactly the same function as typing SET LA36;WIDTH 80.

Any number of TTYSET macros or commands may appear as part of a given keybord's default line. (Note that the DEFAULT command itself should never appear in the list of default commands; doing so will put TTYSET in an infinite loop). The system manager need not include a default line for every keyboard in his system; issuing a DEFAULT command for a keyboard with no entry in the TTYSET.DFL file will cause the message '?No DEFAULTs for KBn:' to be displayed.

A side benefit of the DEFAULT command is a simplification of the start-up terminal control file (normally TTY.CMD). Once keyboard defaults have been defined in the TTYSET.DFL file, the TTY.CMD file can be re-written as

```
:
:
RUN $TTYSET
KB0:;DEFAULT
KB1:;DEFAULT
KB2:;DEFAULT
:
:
KBn:;DEFAULT
EXIT
```

Keyboard configuration, both at start-up time as well as during timesharing is now table-driven from a single file. As new terminals are added, or existing terminals relocated, the manager need only update the TTYSET.DFL file to implement those changes.

Installation

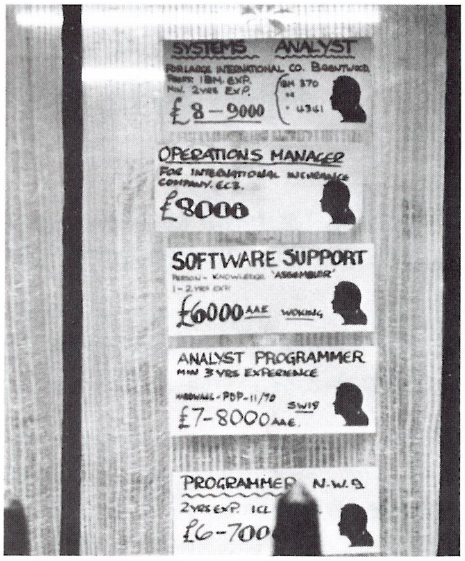
Installation of the DEFAULT option is straightforward. Use the listing provided to patch the TTYSET.BAS source code via the CPATCH patch utility. Be sure that any DIGITAL-supplied patches to TTYSET have been included before compiling the new version of TTYSET. Next, create the file 'SY:[1,2]TTYSET.DFL < 63 > ' as described earlier. You might consider using the /MODE:1536 switch (place at beginning of directory) when creating this file to improve file access time. Finally, make any desired changes to the start-up terminal control file TTY.CMD as described earlier. The DEFAULT command is now available as a standard part of the TTYSET utility.

Patch to Install DEFAULT Command

```
*H/ON -V% GOTO/V<cr>
<tab><tab>ON -V% GOTO 1170,1180,1230,1240,1260,1310 &<cr>
*G/1310/I/,1600/V<cr>
<tab><tab>ON -V% GOTO 1170,1180,1230,1240,1260,1310,1600 &<cr>
*H/! -6/V<cr>
<tab><tab>! -6<tab>PRINT<tab><tab>1310 &<cr>
*AI<cr>
<tab><tab>! -7<tab>DEFAULT<tab><tab>1600 &<cr>
<esc>*V<cr>
<cr>
*H/2000<tab>RESTORE/V<cr>
2000<tab>RESTORE &<cr>
*OAI<cr>
1600<tab>! &<cr>
<tab>!<tab>D E F A U L T   C O M M A N D &<cr>
<tab>! &<cr>
<tab>! &<cr>
<tab>GOSUB 13020 &<cr>
<tab> ON ERROR GOTO 1620 &<cr>
<tab> K% = T% AND 255% &<cr>
<tab> K% = ASCII(MID(SYS(CHR$(6%)+CHR$(9%)),2%,1%))/2% &<cr>
<tab><tab>IF T% AND 128% &<cr>
<tab> OPEN "$TTYSET.DFL" FOR INPUT AS FILE 1% &<cr>
<tab><tab>!GOSUB TO EXTRACT DEFAULT COMMAND &<cr>
<tab><tab>!INIT LOCAL ERR TRAP &<cr>
<tab><tab>!GET KB # FOR DEFAULT &<cr>
<tab><tab>!USE CURRENT KB # IF NO SPECIFIED KB: &<cr>
<tab><tab>!OPEN TTYSET DEFAULT FILE &<cr>
<cr>
1610<tab>INPUT #1%, I%, C1$ &<cr>
<tab> GOTO 1610 UNLESS I% = K% &<cr>
<tab> GOSUB 13000 &<cr>
<tab> CLOSE 1% &<cr>
<tab> C1$ = CVT$$(C1$,1%+4%+8%+16%+32%+128%+256%) &<cr>
<tab> GOTO 1360 &<cr>
<tab><tab>!READ KB# & COMMAND(S) FROM DEFAULT FILE &<cr>
<tab><tab>!DO AGAIN IF KB #'S DON'T MATCH &<cr>
<tab><tab>!IF KB #'S MATCH: &<cr>
<tab><tab>!  GOSUB TO RESET ERROR TRAP &<cr>
<tab><tab>!  CLOSE DEFAULT FILE &<cr>.
<tab><tab>!  CLEAN UP DEFAULT COMMAND STRING &<cr>
<tab><tab>!  PROCEED AS IF IT WERE A MACRO &<cr>
<cr>
1620<tab>PRINT FNQ$;"No DEFAULTs for KB";NUM1$(K%);":" &<cr>
<tab> GOSUB 13000 &<cr>
<tab> CLOSE 1% &<cr>
<tab> GOTO 1495 &<cr>
<tab><tab>!DEFAULT ERROR TRAP: &<cr>
<tab><tab>!  PRINT 'NO DEFAULT' MESSAGE &<cr>
<tab><tab>!  GOSUB TO RESET ERROR TRAP &<cr>
<tab><tab>!  CLOSE DEFAULT FILE &<cr>
<tab><tab>!  GO FINISH UP COMMAND &<cr>
<ff>
<esc>*V<cr>
2000<tab>RESTORE &<cr>
*H/ON -V% GOTO /V<cr>
<tab><tab>ON -V% GOTO 2100,2010,2010,2200,2300,2010 &<cr>
*3G/2010/I/,2010/V<cr>
<tab><tab>ON -V% GOTO 2100,2010,2010,2200,2300,2010,2010 &<cr>
*H/8180/AV<cr>
8190<tab>DATA<tab>"List",31,255<cr>
*I<cr>
8185<tab>DATA<tab>"Default",0,-7<cr>
<esc>*V<cr>
8190<tab>DATA<tab>"List",31,255<cr>
*EX<cr>

Checksum = 46774
```

United Kingdom RSTS Special Interest Group Meeting

By Pauline Noakes and Carl Marbach



£ = $2.40

On October 19, 1980 the RSTS SIG of Great Britain held a special one day seminar in London, England. The Seminar was put on by Al Cini, Dave Mallery, and Carl Marbach all from the U.S. The meeting was attended by about 175 persons from all over the U.K.

The Seminar opened at 10:00 with Al Cini presenting a tutorial on shared libraries. This new feature of Version 7.0 has many different applications and Al spent time developing some of the ones that he has implemented and exploring some new ideas that are yet to be done. For those of you who have not been fortunate enough to listen to Al or attend one of his courses, his style is relaxed but quickly paced; you have to be on your toes all the time, we noticed no one nodding off.

After a question and answer period we broke for lunch. The Royal Festival Hall is on the Thames River and is a new and interesting complex. The Lunch was hurried in order to make more time available for the afternoon sessions.

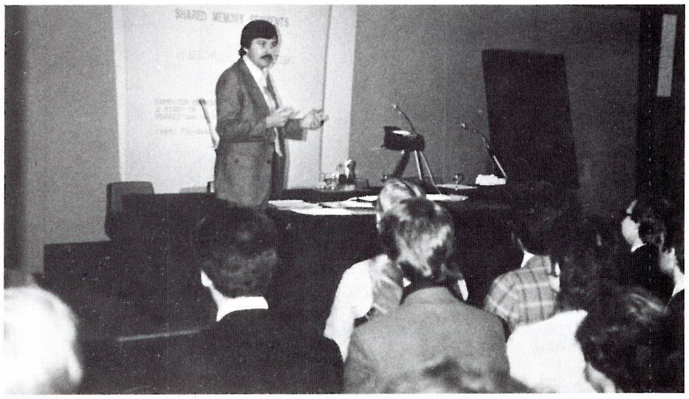
The early part of the afternoon was taken up by Carl and Dave explaining the anatomy of the RSTS monitor and how we could best deal with it in order to build the best system for our applications. We spent some time discussing the new and old features of RSTS and of course the small buffer problem and what to do about it. We also spent part of the afternoon discussing where RSTS was and where it was going; including some ideas from both the floor and podium about where it should go. Rumors flew. Finally we broke for Tea and some cloakroom discussions.

The final part of the day was spent discussing disk structure and how it can be managed. The presentation included the now famous "well structured disk" exposition. Some time was also spent in discussing how we could program effectively for performance.

We ended after dark, a bit tired but better for the day. The U.K. Decus and RSTS SIG are alive and well and planning the next such seminar for 1981. You are all invited!

Ed note: We enjoyed visiting with our U.K. friends and sharing some of our expertise with them. We were excited to find so many new and great ideas within the RSTS community in England. We managed to visit one large computer site and are certain that the level of RSTS expertise in the U.K. is very high. We ourselves felt broadend by the experience and wish to thank the U.K. DECUS and the RSTS SIG for their hospitality, their vitality, and last, for being as wonderful as they are.

Dave Mallery & Pauline Noakes


Al Cini discusses shared libraries.

Carl Marbach at day's end.

DILOG INTERFACES DEC 11*

10 intelligent hard disc and magnetic tape controllers offer LSI-11,* 11/2, 11/23, and PDP-11* single quad slot compatibility with up to 60% power saving.

Only DILOG (Distributed Logic Corporation) exclusive automated design, common proprietary architecture and sophisticated bipolar μPs give you • all single board quad size products requiring no external power or chassis . . . just a cable to connect the drive . . . *you don't need anything else* • high reliability • automated self-test including data base protect feature and indicator. And at cost savings of 50% or more.

LSI-11 MAGNETIC TAPE CONTROLLER, Model DQ 120, interfaces 4 industry standard reel-to-reel drives • emulates TM11* • handles 7 and/or 9 track NRZI drives to 112.5 ips • selectable DEC or IBM byte order formatting • data error checking • RT-11/RSX-11* compatible • extended addressing to 128K words.

LSI-11 MAGNETIC TAPE COUPLER, Model DQ 130, interfaces dual density (NRZI/PE) formatted drives • emulates TM11 • handles up to eight 9 track 800/1600 bpi industry standard drives at speeds from 12.5 to 125 ips • "streamer" mode capability • software or switch selectable density • RT-11/RSX-11 software compatibility.

LSI-11 MASS STORAGE DISC CONTROLLER, Model DQ 200, interfaces any two SMD flat cable interface compatible hard disc drives for up to 500 MB on-line storage • mix or match compatible Winchester, SMD or CMD • variable sector size • automatic media flaw compensation with bad sector flagging • optimized logical to physical unit mapping • implements Winchester fixed head option.

NEW LSI-11 SHUGART SA4000 WINCHESTER DISC CONTROLLER, Model DQ 201, emulates DEC RK* • runs drivers under RT-11 and RSX-11M* systems • compatible with 14.5 MB SA4004 or 29 MB SA4008 drives • automatic media flaw compensation.

LSI-11 DISC CONTROLLER, Model DQ 100, interfaces 2.5, 5, 10 or 20 MB cartridge and fixed platter drives in combinations to 80 MB • RKV-11/RKO5* emulator • handles front load (2315) and/or top load (5440) drives • automatic power fail/power down media protection • RT-11/RSX-11 compatible.

NEW LSI-11 EMULATING MASS STORAGE CONTROLLER, Model DQ 202. Cost effective interface of two 8 and/or 14-inch Winchesters, SMD or CMD hard disc drives without changing controller . . . 8 to 300 MB capacity • RP emulator • automatic media flaw compensation.

PDP-11 MAGNETIC TAPE CONTROLLER, Model DU 120, emulates TM-11 and has same features as Model DQ 120 (LSI unit) • software compatible with RT-11, RSX-11, RSTS, IAS and MUMPS.

NEW PDP-11 MAGNETIC TAPE COUPLER, Model DU 130, offers features of Model DQ 130 (LSI unit) • RT-11, RSX-11, RSTS, IAS and MUMPS software compatible.

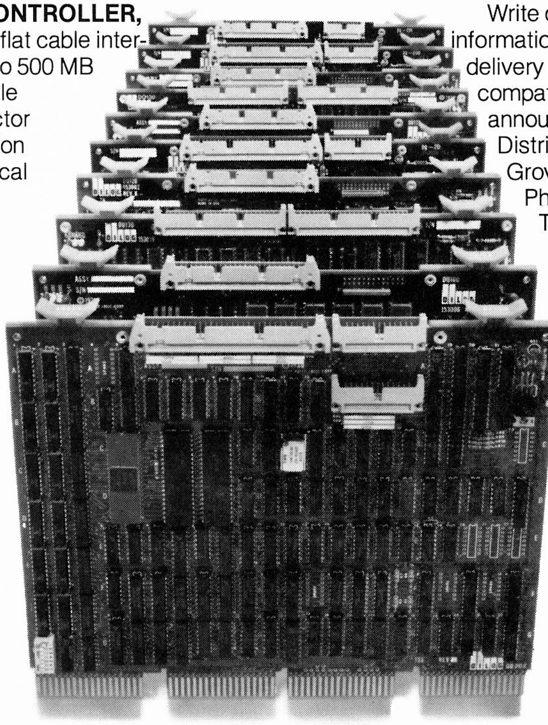
PDP-11 DISC CONTROLLER, Model DU 100 includes features of Model DQ 100 (LSI unit) • RT-11, RSX-11, RSTS, IAS and MUMPS compatible • emulates RK-11.

NEW PDP-11 EMULATING MASS STORAGE CONTROLLER, Model DU 202, offers same features as Model DQ 202 (LSI unit).

Write or call for detailed product performance information, OEM quantity pricing, stock to 30 day delivery or warranty data on these DEC 11 compatible products . . . or several soon to be announced new DILOG products.

Distributed Logic Corp., 12800-G Garden Grove Blvd., Garden Grove, CA 92643
Phone (714) 534-8950
Telex: 681399 DILOG GGVE

DISTRIBUTED LOGIC CORP.
DILOG



All DILOG μP Products are Low Power, Quad Size

*Trademark Digital Equipment Corp.

RSTS/E Disk Structure and Recovery

By Steven L. Edwards & Steven P. Davis, Software Techniques, Inc, Los Alamitos, CA 90720

1.0 Introduction

The purpose of this presentation is to give the reader a sufficient understanding of the RSTS/E on-disk directory structure to recover from most any corruption of this directory structure.

We will start by defining a few common terms, listing a brief overview of the directory structure, examine the RSTS/E minimal directory structure, then dive into the practical application of this knowledge.

2.0 Common definitions

First we must define a few terms so that we can understand each other:

1. Link. A link is 16 bits long. A link is a pointer to an entry. A link is a zero link if (LINK% and Not(15%)) = 0%
2. Entry. An entry is 8 words long. An entry is the building block of the directory structure.
3. Block. A block is 32 entries or 256 words long.
4. LB - logical block. A logical block is the smallest disk subdivision that can be addressed by software. A logical block is 256 words long.
5. LBN - logical block number. Logical block numbers start at 0 and continue with an increment of 1.
6. DC - device cluster. A device cluster is a contiguous group of N blocks where N = DCS. A device cluster is the smallest disk subdivision that can be addressed by hardware.
7. DCN - device cluster number. Every DC is assigned a unique 16-bit number called the DCN. DCN's start at 0 and continue with an increment of 1 until all of the logical blocks of the medium are contained in a DC.
8. DCS - device cluser size. Every disk type has a permanently assigned DCS. The DCS is always a power of 2 between 1 and 16. The DCS assigned to any given disk type is chosen such that the maximum DCN can be described by a 16 bit number.
9. PC - pack cluster. A pack cluster is a contiguous group of N blocks where N = PCS. A PC is the smallest disk subdivision which can be allocated. Each PC is represented by one bit in the storage allocation table.
10. PCN - pack cluster number. Every PC is assigned a unique 16-bit number called the PCN. PCN's start at 0 and continue with an increment of 1 up to the maximum pack cluster number.
11. PCS - pack cluster size. The PCS is always a power of 2 between 1 and 16, and must be greater than or equal to the DCS. The PCS is assigned when the pack is initialized.
12. FC - file cluster. A file cluster is a contiguous group of N blocks where N = FCS.
13. FCS - file cluster size. The FCS is always a power of 2 between 1 and 256, and must be greater than or equal to the PCS. The FCS is assigned when the file is created.

3.0 Overview

The RSTS/E on-disk directory structure is a two level linked list data structure. The first level is called the Master File Directory or MFD. The MFD catalogues accounts or UFD's. The second level is called a User File Directory or UFD. The UFD catalogues files. These two structures are almost identical, differing more in terminology than in substance.

The root of this structure is the pack label which is the first entry in DCN 1. The pack label is also the MFD label which is also the [1,1] label. This is the only entry that is at a fixed location on every pack. This entry links to the first name entry in the MFD.

The name entry can be for either an account or a file. The name entry links to the next name entry. The name entry links to the corresponding accounting entry. If the name entry is for an account, it will contain the 'DCN of 1st UFD cluster. If the name entry is for a file, it links to the corresponding retrieval entry, if any.

The accounting entry links to the attributes entry, if any. The retrieval entry links to the next retrieval entry for this file. A zero link serves as a terminator to the above links.

4.0 RSTS/E minimal directory structure

The RSTS/E minimal directory structure is created by initializing the disk pack. This initialization can be done by INIT.SYS, DSKINT.BAS, or a user written program. The RSTS/E minimal directory structure consists of two accounts ([1,1] and [0,1]), and two files (BADB.SYS, and SATT.SYS).

[1,1]	Label entry (pack label)
[1,1]	Name entry
[1,1]	Accounting entry
[1,1]	Cluster map
[0,1]	Label entry
[0,1]	Name entry
[0,1]	Accounting entry
[0,1]	Cluster map
[0,1]SATT.SYS	Name entry
[0,1]SATT.SYS	Accounting entry
[0,1]SATT.SYS	Retrieval entry (1 - 3 entries)
[0,1]BADB.SYS	Name entry
[0,1]BADB.SYS	Accounting entry
[0,1]BADB.SYS	Retrieval entry (0 - 23 entries)

Thus we can see that the RSTS/E minimal directory structure is composed of 2 label entries, 4 name entries, 4 accounting entries, 1 - 26 retrieval entries, and 2 cluster maps.

RSTS users:

Announcing, DISKIT, the first software tool kit for complete disk management.

Disk Optimization Made Easy

The DISKIT package provides you with all of the tools and utilities you need to create and manage a well-structured disk. Just look at all you get:

DSU Disk Structuring Utility

DSU is a disk-to-disk transfer utility which creates a well-structured disk. Fast. (We transferred 289,000 blocks in 4,000 files in 75 accounts. It took 32 minutes. That's 150 blocks/second.) And the features go on and on:

- Transfers between unlike disks
- Saves all accounting data
- Optimizes file clustersizes
- Places and pre-extends UFD's
- Allows manual file placement
- Full logging and statistics
- Sophisticated error handling and recovery

DIR MACRO-11 Fast Directory Program

Imagine. Disk directories 12 times faster than ever before. Look-up files by name, extention, and date (with wildcards) at the incredible rate of 250 files/second. And DIR isn't just fast, it's smart. DIR supports all of the standard DIRECT switches (including backwards, up to 1,000 files) along with features you can't find anywhere, like password lookup, UFD placement, and UFD size. It even works on dismounted disks, detecting bad directory structures and identifying them with comprehensive error messages. (It's the perfect disk diagnostic!)

DUS MACRO-11 Disk Utility Subroutines

The very same routines used in these disk management tools are available to you, with documentation, so that you can write your own disk handling programs. Included are routines (callable from Basic-Plus 2 or CSPCOM programs) which allow you to create, place, and fully extend UFD's under normal timesharing. In seconds.

OPEN MACRO-11 Open Files Display Program

OPEN displays open files on your system by job, with complete job and file statistics. It even has a sleep switch, allowing you to dynamically update the information at any desired interval.

Proven Technology

You don't have to be a field-test site to keep ahead of the game. All of these products have been thoroughly tested for six months.

Special Offer

All four products (DSU, DIR, DUS, and OPEN) are available separately. But now the entire package, with complete documentation, is only $850. If you need further information, just call us at (213) 594-9405 or write. But, don't wait too long. This offer expires February 28, 1981.

Once again, we've got the answer.

DISKIT from Software Techniques, Inc.

SOFTWARE SPECIALISTS ENGINEERING CONSULTANTS

5242 KATELLA AVE., SUITE #101, LOS ALAMITOS, CA 90720

For practical experience we suggest that the user initialize a pack using ODT.BAS.

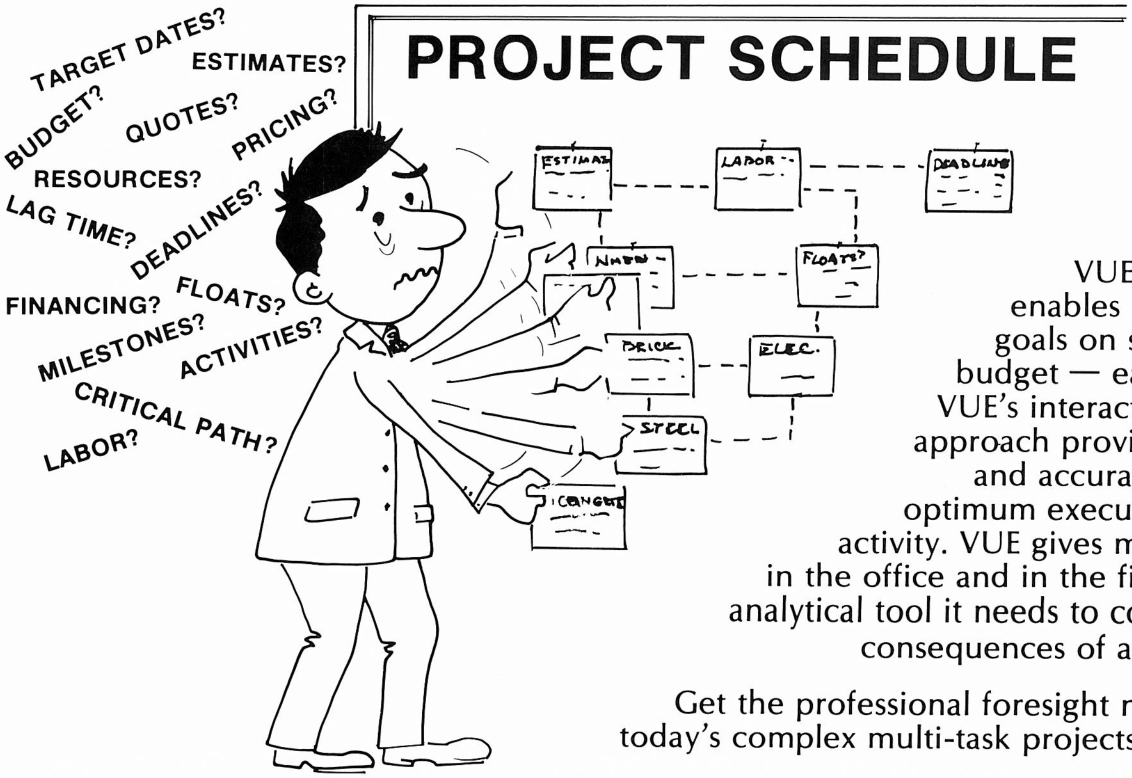
5.0 RSTS/E directory structure recovery

Now we will address the procedures to follow when you get an error during a clean or mount request:

1. Make an image copy of the disk using a stupid copy program ($COPY) or a simple user written program.
2. Put the corrupted disk in a safe place and use the copy to work on.
3. Examine the MFD to determine if it is intact or can be reconstructed. If the MFD has been clobbered, goto step 9.
4. Attempt to mount the disk. Note any errors returned by the system.
5. Attempt to clean the disk using the UU.CLN ($UTILTY) call. Note any errors returned by the system.
6. Find the corresponding error message in the appendix for a possible cause of the error.
7. Using ODT, see if you can correct the problem area on the disk.
8. Goto step 4
9. Re-initialize the disk pack using $DSKINT, placing SATT.SYS at the far end of the disk. DO NOT FORMAT OR PATTERN CHECK THE PACK.
10. Use the program in the appendix (FNDUFD) to locate UFD's.
11. Create the accounts using $REACT.
12. Re-write the'DCN of 1st UFD cluster' (MFD name entry, offset 16 octal) word using the octal value of the DCN displayed by FNDUFD.

5.1 Random notes

INICLN and ONLCLN fix things by zeroing entries, files, accounts, or the MFD. If the cluster maps for a UFD are screwed up, $REORDR will fix them, unless the UFD is [0,1] or [1,1] or the first clustermap is screwed up.



APPENDIX A

System Error Messages and Possible Causes

```
A.1   Mounting a pack
         ?Pack IDs don't match
                  Pack ID specified <> pack ID on disk.
         ?Fatal disk pack mount error
                  PCS > 16
                  PCS < DCS
                  PCS <> power of 2
                  Can't find SATT.SYS
                  SATT.SYS size > 16
                  SATT.SYS too small to map MPCN
                  SATT.SYS start DCN = 0
                  No BADB.SYS
                  BADB.SYS clustersize <> PCS
         ?Disk pack needs 'CLEANing'
                  Dirty bit set
A.2   Cleaning a disk (UU.CLN)
         ?Illegal SYS() usage
                  Device is not a disk
                  Device number not specified
                  Unlocked, read-only, or inuse
         ?Corrupted file structure
                  PCN of file > SATT.SYS's greatest PCN
                  Doubly allocated PCN
A.3   Cleaning a disk. (INICLN, ONLCLN)
         Warning - DCN in BADB.SYS not on pack cluster boundary
         Warning - Bad block doubly allocated in BADB.SYS
         Warning - DCN in BADB.SYS too big
         Warning - Link in BADB.SYS is bad, bad blocks may be lost
         CLEAN will delete account
         CLEAN will zero account
         MFD name entry contains a bad link, CLEAN will delete all  [1,1]
                  files and all accounts beyond [XXX,XXX]
         [XXX,XXX] is not a valid account number
         [XXX,XXX] has invalid account entry link
         [XXX,XXX] has invalid clustersize
         [XXX,XXX] has first DCN out of range
         [XXX,XXX] cluster map in UFD disagrees with MFD
         [XXX,XXX] has holes in cluster map
         [XXX,XXX] has UFD cluster number which is too big
         [XXX,XXX] has inconsistent cluster maps
         CLEAN will delete all files in [XXX,XXX]
         Invalid clustersize for file ??????.???
         User file looks like UFD:
         Invalid retrieval entries for file
         Fixed by CLEAN
         UFD has size too large for file
         UFD has size too small for file - changed to X by CLEAN
         Entry is a hole
         Cluster allocated to ??????.???  is not on a pack cluster  boun-
                  dry
         Directory entry for ??????.???   contains  pack  cluster  number
                  which is too big
                  (A disk is irrevocably corrupt only if you  for-
                           matted  the pack, ran pattern checks, or
                           a user program wrote over virtually  all
                           of the disk)
```

APPENDIX B
Directory Entry Formats

MFD Label Entry

```
                 !---------------------------------------!
     ULNK   +1   !   Link to 1st Name Entry in MFD       !  +0
                 !---------------------------------------!
            +3   !     -1 (to mark entry in use)         !  +2
                 !---------------------------------------!
            +5   !                 0                     !  +4
                 !---------------------------------------!
            +7   !                 0                     !  +6
                 !---------------------------------------!
            +11  !       Pack Cluster Size (PCS)         !  +10
                 !---------------------------------------!
            +13  !            Pack Status                !  +12
                 !---------------------------------------!
            +15  !   Pack ID (Part 1 of RAD50 Name)      !  +14
                 !---------------------------------------!
            +17  !   Pack ID (Part 2 of RAD50 Name)      !  +16
                 !---------------------------------------!
```

MFD Name Entry

```
                 !---------------------------------------!
     ULNK   +1   !   Link to Next Name Entry in MFD      !  +0
                 !------------------!--------------------!
     UNAM   +3   ! PPN Project      ! PPN Programmer     !  +2
                 !------------------!--------------------!
            +5   !     Password (Part 1 in RAD50)        !  +4
                 !---------------------------------------!
            +7   !     Password (Part 2 in RAD50)        !  +6
                 !------------------!--------------------!
UPROT/USTAT +11  ! Protection Code  ! Status Byte        !  +10
                 !------------------!--------------------!
    UACNT   +13  !            Access Count               !  +12
                 !---------------------------------------!
      UAA   +15  !      Link to Accounting Entry         !  +14
                 !---------------------------------------!
      UAR   +17  !       DCN of 1st UFD Cluster          !  +16
                 !---------------------------------------!
```

MFD Accounting Entry

```
                 !---------------------------------------!
     ULNK   +1   !    Link to Attributes Entry  !0!0!B!1!  +0
                 !---------------------------------------!
     MCPU   +3   !       Accumulated CPU Time (LSB)      !  +2
                 !---------------------------------------!
     MCON   +5   !  Accumulated Connect Time (Minutes)   !  +4
                 !---------------------------------------!
     MKCT   +7   !  Accumulated Kilo-Core-Ticks (LSB)    !  +6
                 !---------------------------------------!
     MDEV   +11  !  Accumulated Device Time (Minutes)    !  +10
                 !----------------!----------------------!
     MMSB   +13  ! CPU Time (MSB) ! Kilo-Core-Ticks (MSB)!  +12
                 !----------------!----------------------!
    MDPER   +15  !     Logout Quota of Disk Blocks       !  +14
                 !---------------------------------------!
    UCLUS   +17  !     UFD Cluster Size (UCS)            !  +16
                 !---------------------------------------!
```

MFD Cluster Map Entry

```
                 !---------------------------------------!
            +1   !       MFD Cluster Size (MCS)          !  +0
                 !---------------------------------------!
            +3   !         DCN of MFD Cluster 0          !  +2
                 !---------------------------------------!
            +5   !         DCN of MFD Cluster 1          !  +4
                 !---------------------------------------!
            +7   !         DCN of MFD Cluster 2          !  +6
                 !---------------------------------------!
            +11  !         DCN of MFD Cluster 3          !  +10
                 !---------------------------------------!
            +13  !         DCN of MFD Cluster 4          !  +12
                 !---------------------------------------!
            +15  !         DCN of MFD Cluster 5          !  +14
                 !---------------------------------------!
            +17  !         DCN of MFD Cluster 6          !  +16
                 !---------------------------------------!
```

UFD Label Entry

```
                 !---------------------------------------!
     ULNK   +1   ! Link to 1st Name Entry in UFD (or 0)  !  +0
                 !---------------------------------------!
            +3   !                 -1                    !  +2
                 !---------------------------------------!
            +5   !                 0                     !  +4
                 !---------------------------------------!
            +7   !                 0                     !  +6
                 !---------------------------------------!
            +11  !                 0                     !  +10
                 !---------------------------------------!
            +13  !                 0                     !  +12
                 !---------------------------------------!
            +15  !   PPN Project      ! PPN Programmer   !  +14
                 !---------------------------------------!
            +17  !           "UFD" (in RAD50)            !  +16
                 !---------------------------------------!
```

UFD Name Entry

```
                 !---------------------------------------!
     ULNK   +1   !    Link to Next Name Entry in UFD     !  +0
                 !---------------------------------------!
     UNAM   +3   !     File Name (Part 1 in RAD50)       !  +2
                 !---------------------------------------!
            +5   !     File Name (Part 2 in RAD50)       !  +4
                 !---------------------------------------!
            +7   !     File Name Extension (RAD50)       !  +6
                 !------------------!--------------------!
UPROT/USTAT +11  ! Protection Code  ! Status Byte        !  +10
                 !------------------!--------------------!
    UACNT   +13  !         File Access Count             !  +12
                 !---------------------------------------!
      UAA   +15  !  Link to Accounting Entry for File    !  +14
                 !---------------------------------------!
      UAR   +17  ! Link to 1st Retrieval Entry for File  !  +16
                 !---------------------------------------!
```

UFD Accounting Entry

```
                 !---------------------------------------!
     ULNK   +1   !    Link to Attributes Entry  !0!0!B!1!  +0
                 !---------------------------------------!
     UDLA   +3   !          Last Access Date             !  +2
                 !---------------------------------------!
     USIZ   +5   !       Number of Blocks in File        !  +4
                 !---------------------------------------!
      UDC   +7   !            Creation Date              !  +6
                 !---------------------------------------!
      UTC   +11  !            Creation Time              !  +10
                 !---------------------------------------!
     URTS   +13  ! Run-Time System Name (Part 1 in RAD50)!  +12
                 !---------------------------------------!
            +15  ! Run-Time System Name (Part 2 in RAD50)!  +14
                 !---------------------------------------!
    UCLUS   +17  !       File Cluster Size (FCS)         !  +16
                 !---------------------------------------!
```

UFD Attributes Entry

```
                 !---------------------------------------!
     ULNK   +1   ! Link to Next Attributes Entry !0!0!0!1!  +0
                 !---------------------------------------!
            +3   !              Word 1                   !  +2
                 !---------------------------------------!
            +5   !              Word 2                   !  +4
                 !---------------------------------------!
            +7   !              Word 3                   !  +6
                 !---------------------------------------!
            +11  !              Word 4                   !  +10
                 !---------------------------------------!
            +13  !              Word 5                   !  +12
                 !---------------------------------------!
            +15  !              Word 6                   !  +14
                 !---------------------------------------!
            +17  !              Word 7                   !  +16
                 !---------------------------------------!
```

Unused Entries

```
                 !---------------------------------------!
            +1   !                 0                     !  +0
                 !---------------------------------------!
            +3   !                 0                     !  +2
                 !---------------------------------------!
            +5   !                                       !  +4
                 !---------------------------------------!
            +7   !                                       !  +6
                 !---------------------------------------!
            +11  !                                       !  +10
                 !---------------------------------------!
            +13  !                                       !  +12
                 !---------------------------------------!
            +15  !                                       !  +14
                 !---------------------------------------!
            +17  !                                       !  +16
                 !---------------------------------------!
```

```
UFD Retrieval Entry

                  !-----------------------------------------!
ULNK   +1         ! Link to Next Retrieval Entry !0!0!B!0   !  +0
                  !-----------------------------------------!
UENT   +3         !          DCN of File Cluster N+0        !  +2
                  !-----------------------------------------!
       +5         !          DCN of File Cluster N+1        !  +4
                  !-----------------------------------------!
       +7         !          DCN of File Cluster N+2        !  +6
                  !-----------------------------------------!
       +11        !          DCN of File Cluster N+3        !  +10
                  !-----------------------------------------!
       +13        !          DCN of File Cluster N+4        !  +12
                  !-----------------------------------------!
       +15        !          DCN of File Cluster N+5        !  +14
                  !-----------------------------------------!
       +17        !          DCN of File Cluster N+6        !  +16
                  !-----------------------------------------!

UFD Cluster Map Entry

                  !-----------------------------------------!
       +1         !          UFD Cluster Size (UCS)         !  +0
                  !-----------------------------------------!
       +3         !          DCN of UFD Cluster 0           !  +2
                  !-----------------------------------------!
       +5         !          DCN of UFD Cluster 1           !  +4
                  !-----------------------------------------!
       +7         !          DCN of UFD Cluster 2           !  +6
                  !-----------------------------------------!
       +11        !          DCN of UFD Cluster 3           !  +10
                  !-----------------------------------------!
       +13        !          DCN of UFD Cluster 4           !  +12
                  !-----------------------------------------!
       +15        !          DCN of UFD Cluster 5           !  +14
                  !-----------------------------------------!
       +17        !          DCN of UFD Cluster 6           !  +16
                  !-----------------------------------------!
```

APPENDIX C

Directory Links

C.0.1 Directory Links

Directory links are the one word pointers used to tie the whole directory structure together. Directory links specify which entry, within which block, within which cluster the desired information can be found.

```
!1!1!1!1!1!1!
!5!4!3!2!1!0!9!8!7!6!5!4!3!2!1!0!
!-------!-----!---------!-------!
! Block !Clstr!  Entry  ! Flags !
!-------!-----!---------!-------!
```

There are for elements of a directory link, BLOCK, CLUSTER, ENTRY AND FLAG.

1. Block - These 4 bits select the block within the directory cluster.

BLOCK% = SWAP%(LINK% AND -4096%) / 16%

2. Cluster - These 3 bits select the directory cluster by selecting one of the DCN's from the directory cluster map.

CLSTR% = (LINK% AND 3584%) / 512%

3. Entry - These 5 bits select the entry within the directory block.

ENTRY% = (LINK% AND 496%) / 16%

4. Flags - These 4 bits are used for informational purposes like:this entry is in use, this file or account contains a bad block, this file is to be cached (data caching), or while cleaning a disk to indicate that this link has been referenced already. These bits will be ignored for our purposes of disk recovery.

APPENDIX D

FNDUFD

This program will search through an entire disk looking for UFD label entries. When the entry is found, the project, programmer number and the DCN of the first UFD cluster is displayed.

This program can be compiled by either Basic-Plus 2 or CSPCOM. Basic-Plus die-hards will have to translate the program down to their level.

```
1!      Title:          F N D U F D
!
!       FiND User File Directories by locating UFD label entries.

901     MAP     (BUF)
                 FILLS = 2%                 ! LINK TO FIRST NAME ENTRY.
                ,FLAG%                      ! MUST BE -1.
                ,FILLS = 8%                 ! 4 WORDS OF 0.
                ,PPN%                       ! PROJECT-PROGRAMMER NUMBER.
                ,UFD%                       ! "UFD" IN RAD50.
                ,FILLS = 496%               ! FILL OUT TO 512 BYTES.
                ,FILLS = 1536%              ! (DCS - 1) * 512
                ! MAP THE INPUT BUFFER SO THAT THE TOTAL SIZE = DCS * 512.

1000    Onerror Goto 19000
                ! Set standard error trap.

1010    Is = "V7.0-01"
®       Print "FNDUFD    " + Is + "      Software Techniques" + CR + LF +
                "FIND UFD'S" + CR + LF
                ! Print standard header.

1030    UFD.RAD.50% = -31692%
®       REC.CNT = 0.0
                ! SET UFD.RAD.50% = RAD50 OF "UFD"
                ! INITIALIZE THE RECORD COUNTER.

2000!           Start of MAIN

2010    PRINT "ENTER DISK <EXIT> ";
®       LINPUT TEMP.0s
®       TEMP.0s = EDITs(TEMP.0s, -1%)
®       GOTO 32700
                UNLESS  LEN(TEMP.0s)
®       OPEN TEMP.0s FOR INPUT AS FILE  1%
                ,MODE 8192%
                ,MAP BUF
                ! OPEN THE SUSPECTED DISK.

2020    GET  1%
®       IF      UFD% = UFD.RAD.50%
        THEN    IF      FLAG% = -1%
                THEN    PRINT "[" + NUM1s(SWAP%(PPN%) AND 255%)
                                + "," + NUM1s(PPN% AND 255%) + "]"
                                + " AT DCN " + NUM1s(REC.CNT)
                ! GET A DCN.
                ! SEE IF IT IS A UFD LABEL ENTRY.
                ! PRINT OUT THE PPN AND THE DCN

2030    REC.CNT = REC.CNT + 1.0
®       GOTO 2020
                ! INCREMENT THE BLOCK COUNTER.
                ! LOOP BACK UNTIL EOF.

19000!          Error Handler

19011   If      Err = 11%
        Then    Resume 32700
                ! End of file on device.

19999   Onerror Goto
                ! Give up.

32700!          Completion Routines

32710   Close 1%
                ! Close all channels.

32767   End
```

The System Works For You, But . . .

By Jeffrey R. Harrow

This is the first installment of a column devoted to the RSTS/E System Manager. It will deal with issues that are, or should be relevant to the person charged with overseeing the operation of any RSTS/E system.

I'll be dealing with system-related software that works well, or, well, sort of works, both from DEC and from other vendors, hardware of special note, tips on ways to more easily maintain your system, and tips on things **not** to do so as to continue to maintain your system.

For this issue, let's consider a couple of things **not** to do.

The first item shows an easy way to instantly make your system pack unusable as a bootable pack. Let's say that you're experiencing some hardware problems and choose to disable one or more "devices" in order to determine if they are the culprits (this scenerio is just a plausable excuse to find yourself using the "DISABLE" suboption of the "HARDWARE" option of the INIT code).

If, while typing in the name of the controller you inadvertently type the name of the controller from which you're currently booted (for instance if you're booted from an RM03 and you type "DR"), then when you leave the HARDWARE option and the system automatically re-boots itself, **it will be unable to come up.**

Unless you have another pack from which you can boot, and you are **very** smart on the internal structure of the SIL and/or INIT code and effectively re-ENABLE the controller with an ODT-like program, your pack is unusable as the system pack (although, assuming that you boot from another pack, you could mount the now un-bootable pack as a PRIVATE pack and recover its files).

The moral? Be rather careful as you disable controllers!

The second item on which to give your attention is Datatrieve. While I am an avid user and proponent of DTR, the RSTS/E implementation has an as yet unresolved **serious** system problem.

The DTR V2.0 installation kit leaves [1,2]DTR.TSK with a protection code of < 104 > (allows all users to execute the task with no Temporary Privilege), and leaves the central data dictionary (QUERY.DIC) in the LB: account (which should not be [1,2]) with a protection code of < 40 > (allows all users Read-Only access to the file).

The problem develops when a non-privileged user performs any operation, through DTR, which requires him to write to the central data dictionary (a common operation). The user is denied write access to the data dictionary because, as it's in an account different from his own (LB:), and it has a protection code of < 40 >, RSTS/E disallows write access. Therefore, normal DTR activities are impossible.

Well, there are two "work-arounds": Set the protection code of the central data dictionary to < 0 > so that users are not prevented from writing to the file, or give [1,2]DTR.TSK "Temporary Privilege" by setting its protection code to < 232 >. Let's examine the implications of each "work-around".

Setting the protection code of the central data dictionary to < 0 > will indeed allow all DTR users to update the central data dictionary as required. Unfortunately, it also allows **any** non-privileged user to directly, **without going through DTR,** examine the entire dictionary (including the DTR protection mechanisms contained therein), modify the dictionary using any editor, or intentionally or unintentionally delete the central data dictionary (as: UNSAVE LB:QUERY.DIC), thereby losing all work in the central data dictionary by all users since the last BACKUP of LB:QUERY.DIC!

Needless to say, this opens the system to the potential for **serious** loss of data and compromised security, and blatently violates the generally secure status of most RSTS/E systems.

So, what happens if we set LB:QUERY.DIC to a protection code of < 60 > (don't allow users to be able to **directly** get to the central data dictionary), and give [1,2]DTR.TSK "Temporary Privilege" with a protection code of < 232 > ?

All users can now read and write to the central data dictionary, but **only** through DTR (which imposes appropriate protections against modifying or reading other user's information). The non-privileged user can no longer **directly** read, modify, or delete the central data dictionary, since RSTS/E imposes system protection based upon its < 60 > protection code. In effect, this would seem the ideal solution.

However, there is one fly in the DTR ointment! DTR was not written to, in the appropriate places, "temporarily drop Temporary Privilege". This means that during a DTR session, **any non-privileged user can execute any privileged activity of which DTR is capable!**

Now this in general would not pose too much of a threat to system security, **except for the DEFINE FILE command.** This command allows a user to create a file for a domain. When DTR is **not** privileged, there is no problem because RSTS/E will keep a non-privileged user from creating a file in an account other than his own. **However,** with DTR having the "Temporary Privilege" attribute, **there is NO protection, of any kind, to keep any non-privileged user from creating a file in ANY account, and thereby implicitly deleting ANY file (system or user) on the entire system!**

This is obviously, a completely untenable condition. Therefore, right now, check your DTR file (DIR $DTR.TSK/S) and verify that it's protection code is <104> and NOT < 232 > !

Based on this information, currently the safest (NOT safe!) method of using DTR (assuming that you require a central data dictionary) is that indicated in the first scenerio (LB:QUERY.DIC < 0>), even though this **seriously** compromises your DTR security and data dictionary safety.

I sent DEC an SPR outlining this set of conditions, and they indicated that, indeed the installation kit **should** have set the central data dictionary's protection code to <0> , and that the installation kit would be fixed in the future. Then, even though a central data dictionary is **a central part of the Datatrieve functionality** (to allow dictionary element sharing), they finished:

> "With a common dictionary, all users need write access. A possible solution to the vulnerability of the common dictionary is to use private dictionaries to which only the owner may write."

I have subsequently talked with some folks in SPR Administration, and they indicated that perhaps they will look more deeply into a method of actually resolving the problem and making Datatrieve, with its central data dictionary, as secure as the rest of the RSTS/E layered products. I'll keep you informed of their progress, but in the meantime, beware.

RESIDENT LIBRARIES II

. . . continued from page 11

UMPFQ.

This subroutine "unmaps" a mapped area.

CALL UMPFQ (ERROR%, WINDOW.ID%, WINDOW.STATUS%)

where:

ERROR% Returned RSTS error code (3-189)

WINDOW.ID% Identifier of window to be unmapped (3-188, FIRQB+6)

WINDOW.STATUS% Returned status flags (3-189, FIRQB+22).

Assembling the PLAS Subroutines.

Once keyed into PLAS.MAC (via EDT, EDIT, or whatever), the PLAS assembly language subroutines below can be assembled as follows:

```
RUN $MAC
MAC > PLAS = PLAS
MAC > ↑ Z
Ready
```

There should be no errors in this assembly.

Linking to the PLAS Subroutines.

Once assembled, the PLAS.OBJ moduled can be linked into your BP2 program by editing your ODL file to include a reference to it. Alternatively, you can use BUILD:

```
BUILD MAIN, PLAS,   < other subroutines >
```

A complete example.

The BP2 routine which follows simply loads the integers 1-32700 into a 32KW resident common area, and then prints them on the terminal (it uses APR 7 — 4K of user space — to accomplish this).

An undocumented BP2 subroutine — RAD — is used to convert the library name "VMTRX" to RAD50. You can use the filename string scan SYS call if you prefer.

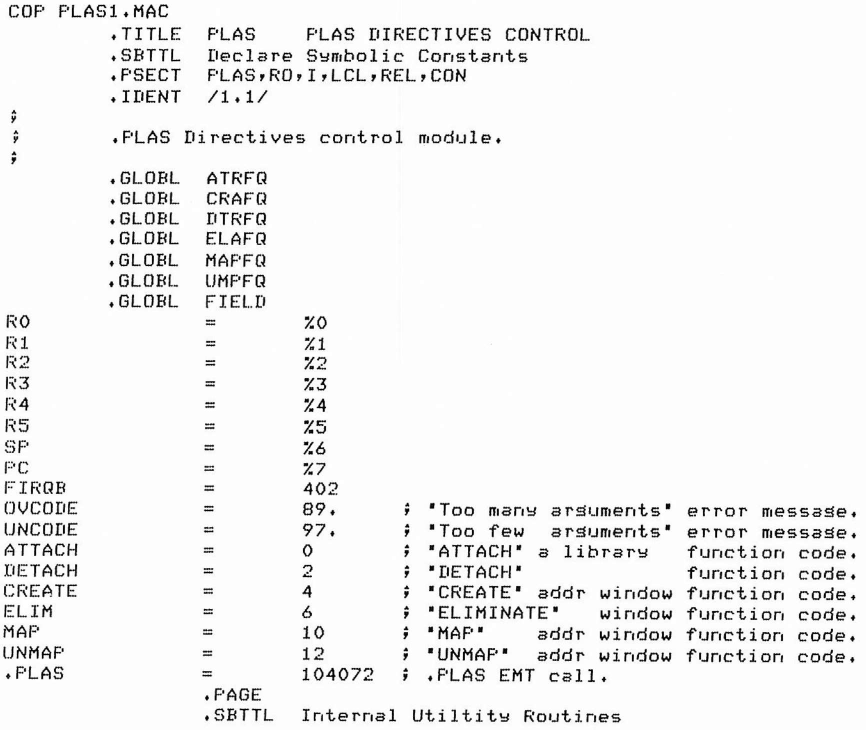
The VMTRX shared library was created as outlined in the "Programmer's Guide to Shared Libraries" article cited earlier, using the following assembly language program:

```
.TITLE     VMTRX
.PSECT     VMTRX,RW,D,GBL,REL,OVR
.BLKW      32700.
.END
```

Of course, since our BP2 program uses PLAS calls to "attach" the library at run-time, there is no need to link it to VMTRX via TKB "RESLIB" commands (in fact, this wouldn't work at all — TKB can't put 32KW of data into a 4KW bag).

In closing . . .

This exercise, as you can imagine, is not really for the faint of heart. As a practical matter, this technique will be of use in only a very narrow realm of systems programming applications and certainly not in routine DP. If you think you can use it, give it a try. You can't hurt anything (unless you attach and clobber some production shared library — impossible, if they were ADDed with the right protection code), and you'll learn a whole lot about PDP-11 memory mapping.



```
COP PLAS1.MAC
        .TITLE  PLAS    PLAS DIRECTIVES CONTROL
        .SBTTL  Declare Symbolic Constants
        .PSECT  PLAS,RO,I,LCL,REL,CON
        .IDENT  /1.1/
;
;       .PLAS Directives control module.
;
        .GLOBL  ATRFQ
        .GLOBL  CRAFQ
        .GLOBL  DTRFQ
        .GLOBL  ELAFQ
        .GLOBL  MAPFQ
        .GLOBL  UMPFQ
        .GLOBL  FIELD
R0              =       %0
R1              =       %1
R2              =       %2
R3              =       %3
R4              =       %4
R5              =       %5
SP              =       %6
PC              =       %7
FIRQB           =       402
OVCODE          =       89.     ; 'Too many arguments' error message.
UNCODE          =       97.     ; 'Too few  arguments' error message.
ATTACH          =       0       ; 'ATTACH' a library   function code.
DETACH          =       2       ; 'DETACH'             function code.
CREATE          =       4       ; 'CREATE' addr window function code.
ELIM            =       6       ; 'ELIMINATE'   window function code.
MAP             =       10      ; 'MAP'    addr window function code.
UNMAP           =       12      ; 'UNMAP'  addr window function code.
.PLAS           =       104072  ; .PLAS EMT call.
               .PAGE
               .SBTTL   Internal Utiltity Routines
```

```
;
; "Utility" subroutines used by all .PLAS functions.
;
; This subroutine sets the "too many"/"too few" error code.
;
OVRARG:                 MOVB    #OVCODE,FIRQB
                        JMP     RETURN
UNDARG:                 MOVB    #UNCODE,FIRQB
;
; This subroutine moves the FIRQB error code byte into the caller's
; ERROR% argument, then returns to the calling program.
;
RETURN:                 CLR     @2(R5)
                        MOVB    FIRQB,@2(R5)
                        RTS     PC
;
; This subroutine zeroes the FIRQB control area.
;
CLRFQB:                 MOV     FIRQB,R0
CLROP:                  CLR     (R0)+
                        CMP     #FIRQB+38.,R0
                        BLT     CLROP
                        RTS     PC
;
                        .PAGE
                        .SBTTL  "ATTACH" Resident Library Function
;
; "Attach" resident library function BP2 calling sequence:
;
ATRFQ:                  CMPB    #6.,(R5)
                        BEQ     .+14
                        BGT     .+6
                        JMP     OVRARG
                        JMP     UNDARG
                        JSR     PC,CLRFQB
                        MOV     @4(R5),FIRQB+12         ; RESLIB.NAME1% (RAD50)
                        MOV     @6(R5),FIRQB+14         ; RESLIB.NAME2% (RAD50)
                        MOV     @10(R5),FIRQB+22        ; ACCESS.MODE%
                        MOV     #ATTACH,FIRQB+4
                        .PLAS
                        MOV     FIRQB+6,@12(R5)         ; RESLIB.ID%
                        MOV     FIRQB+10,@14(R5)        ; RESLIB.SIZE%
                        JMP     RETURN
;
                        .PAGE
                        .SBTTL  "CREATE" a Memory Window
;
; "Create" memory window function BP2 calling sequence:
;
CRAFQ:                  CMPB    #11.,(R5)
                        BEQ     .+14
                        BGT     .+6
                        JMP     OVRARG
                        JMP     UNDARG
                        JSR     PC,CLRFQB
                        MOV     @4(R5),FIRQB+14         ; RESLIB.ID%
                        MOVB    @6(R5),FIRQB+7          ; BASE.APR%
                        MOV     @10(R5),FIRQB+12        ; WINDOW.SIZE%
                        MOV     @12(R5),FIRQB+16        ; MAP.AREA.OFFSET%
                        MOV     @14(R5),FIRQB+20        ; MAP.SIZE%
                        MOV     @16(R5),FIRQB+22        ; ACCESS.MODE%
                        MOV     #CREATE,FIRQB+4
                        .PLAS
                        MOV     FIRQB+6,@20(R5)         ; WINDOW.ID%
                        MOV     FIRQB+10,@22(R5)        ; WINDOW.START%
                        MOV     FIRQB+20,@24(R5)        ; ACTUAL.MAP.LENGTH%
                        MOV     FIRQB+22,@26(R5)        ; WINDOW.STATUS%
                        JMP     RETURN
;
                        .PAGE
                        .SBTTL  "DETACH" From a Memory Segment
;
; "Detach' from memeory segment function BP2 calling sequence:
;
DTRFQ:                  CMPB    #3.,(R5)
                        BEQ     .+14
                        BGT     .+6
                        JMP     OVRARG
                        JMP     UNDARG
                        JSR     PC,CLRFQB
                        MOV     @4(R5),FIRQB+6          ; RESLIB.ID%
                        MOV     DETACH,FIRQB+4
                        .PLAS
                        MOV     FIRQB+22,@6(R5)         ; DETACH.STATUS%
                        JMP     RETURN
;
                        .PAGE

                        .SBTTL  "ELIMINATE" a Memory Window
;
```

```
; "Eliminate" address window function BP2 calling sequence:
;
ELAFQ:          CMPB    #3.,(R5)
                BEQ     .+14
                BGT     .+6
                JMP     OVRARG
                JMP     UNDARG
                JSR     PC,CLRFQB
                MOV     @4(R5),FIRQB+6          ; WINDOW.ID%
                MOV     ELIM,FIRQB+4
                .PLAS
                MOV     FIRQB+22,@6(R5)         ; WINDOW.STATUS%
                JMP     RETURN
;
                .PAGE
                .SBTTL  "MAP" an Area within a Window
;
; "Map" part of an address window function BP2 calling sequence:
;
MAPFQ:          CMPB    #8.,(R5)
                BEQ     .+14
                BGT     .+6
                JMP     OVRARG
                JMP     UNDARG
                JSR     PC,CLRFQB
                MOV     @4(R5),FIRQB+6          ; WINDOW.ID%
                MOV     @6(R5),FIRQB+14         ; RESLIB.ID%
                MOV     @10(R5),FIRQB+16        ; MAP.AREA.OFFSET%
                MOV     @12(R5),FIRQB+20        ; MAP.AREA.SIZE%
                MOV     @14(R5),FIRQB+22        ; MAP.ACCESS.MODE%
                MOV     #MAP,FIRQB+4
                .PLAS
                MOV     FIRQB+20,@16(R5)        ; MAP.AREA.LENGTH%
                MOV     FIRQB+22,@20(R5)        ; MAP.AREA.STATUS%
                JMP     RETURN
;
                .PAGE
                .SBTTL  "UNMAP" a Memory Window
;
; "Unmap" an area function BP2 calling sequence:
;
UMPFQ:          CMPB    #3.,(R5)
                BEQ     .+14
                BGT     .+6
                JMP     OVRARG
                JMP     UNDARG
                JSR     PC,CLRFQB
                MOV     @4(R5),FIRQB+6          ; WINDOW.ID%
                MOV     #UNMAP,FIRQB+4
                .PLAS
                MOV     FIRQB+22,@6(R5)         ; WINDOW.STATUS%
                JMP     RETURN
;
                .PAGE
                .SBTTL  "Field" a mapped area using a string variable
;
; "Field" a mapped area using a string variable BP2 calling sequence:
;
FIELD:          CMPB    #4.,(R5)
                BEQ     .+14
                BGT     .+6
                JMP     OVRARG
                JMP     UNDARG
                MOV     10(R5),R0       ; Address of string header in R0
                MOV     @4(R5),(R0)+    ; Poke string address field in header
                MOV     @6(R5),(R0)     ; Poke string length  field in header
                CLR     @2(R5)          ; Clear error indicator
                RTS     PC
;
                .END


COP VIRTUE.B2S
10      MAP.BUFFER$ = "" &
!
1000    X% = FNINITIALIZE.RESIDENT.ARRAY% &
\       IF ERROR% THEN &
                PRINT "Failure to initialize resident array" &
\               PRINT "--Error code:"; ERROR% &
\               GO TO 32767 &
!       (ELSE) &
!       ENDIF. &
!
1010    FOR I%=0% TO 32700% &
\               X% = FNSTORE.ELEMENT% (I%, I%) &
\               IF ERROR% THEN &
                        PRINT "Failure to store element in location"; I% &
\                       PRINT "--Error code:"; ERROR% &
\                       GO TO 32767 &
```

```
!               (ELSE) &
!               ENDIF. &
!
1020    NEXT I% &
!
1030    PRINT "Virtual matrix initialized:" &
\       PRINT FNFETCH.ELEMENT% (I%); &
                FOR I%=0% TO 32700% &
!
10000 &
DEF FNINITIALIZE.RESIDENT.ARRAY% &
!
10010   LIBRARY.NAME$ = SPACE$(6%) &
\       LSET LIBRARY.NAME$ = "VMTRX" &
\       CALL RAD (LIBNAME1%, LEFT(LIBRARY.NAME$,3%)) &
\       CALL RAD (LIBNAME2%, RIGHT(LIBRARY.NAME$,4%)) &
\       ACCESS.MODE% = 2%       ! R/W Access requested &
\       CALL ATRFQ (ERROR%, LIBNAME1%, LIBNAME2%, ACCESS.MODE%, &
                        RESLIB.ID%, RESLIB.SIZE%) &
\       FNEXIT &
                IF ERROR% &
\       BASE.APR% = 7% &
\       WINDOW.SIZE%, MAP.AREA.SIZE% = 128% &
\       MAP.AREA.OFFSET% = 0% &
\       ACCESS.MODE% = 130% &
\       CALL CRAFQ (ERROR%, RESLIB.ID%, BASE.APR%, WINDOW.SIZE%, &
                MAP.APEA.OFFSET%, MAP.AREA.SIZE%, ACCESS.MODE%, WINDOW.ID%, &
                        WINDOW.START%, ACTUAL.MAP.LENGTH%, WINDOW.STATUS%) &
\       CURRENT.PAGE% = 0%      ! Currently, we are mapping the bottom 4KW &
                                ! of the resident array. &
\FNEND &
 &
 &

11000 &
DEF FNSTORE.ELEMENT% (ARRAY.LOCN%, VALUE%) &
!
11010   REQD.PAGE% = ARRAY.LOCN%/4096% &
\       CURRENT.PAGE% = FNPAGE.MAP%(REQD.PAGE%) &
                UNLESS REQD.PAGE% = CURRENT.PAGE% &
\       PAGE.OFFSET% = (ARRAY.LOCN% AND 4095%) * 2%&
\       PAGE.ADDRESS% = BASE.APR%*8192%+PAGE.OFFSET% &
\       CALL FIELD (ERROR%, PAGE.ADDRESS%, 2%, MAP.BUFFER$) &
\       LSET MAP.BUFFER$ = CVT%$(VALUE%) &
\FNEND &
 &
 &

12000 &
DEF FNFETCH.ELEMENT% (ARRAY.LOCN%) &
!
12010   REQD.PAGE% = ARRAY.LOCN%/4096% &
\       CURRENT.PAGE% = FNPAGE.MAP%(REQD.PAGE%) &
                UNLESS CURRENT.PAGE% = REQD.PAGE% &
\       PAGE.OFFSET% = (ARRAY.LOCN% AND 4095%) * 2%&
\       PAGE.ADDRESS% = BASE.APR%*8192%+PAGE.OFFSET% &
\       CALL FIELD (ERROR%, PAGE.ADDRESS%, 2%, MAP.BUFFER$) &
\       FNFETCH.ELEMENT% = CVT$%(MAP.BUFFER$) &
\FNEND &
 &
 &

12500 &
DEF FNPAGE.MAP% (PAGE.TO.MAP%) &
!
12510   MAP.ACCESS.MODE% = 2%   ! R/W access to map area desired &
\       MAP.AREA.OFFSET% = PAGE.TO.MAP%*128% &
\       CALL MAPFQ (ERROR%, WINDOW.ID%, RESLIB.ID%, MAP.AREA.OFFSET%, &
                        MAP.AREA.SIZE%, MAP.ACCESS.MODE%, MAP.AREA.LENGTH%, &
                                MAP.AREA.STATUS%) &
\       FNPAGE.MAP% = PAGE.TO.MAP% &
\FNEND &
 &
 &

13000 &
DEF FNCLOSE.RESIDENT.ARRAY% &
!
13010   CALL UMPFQ (ERROR%, WINDOW.ID%, WINDOW.STATUS%) &
\       CALL DTRFQ (ERROR%, RESLIB.ID%, DETACH.STATUS%) &
\FNEND &
 &
 &

32767   END &
 &
 &
```

AUTHORS!!

SEND YOUR ARTICLES TO THE RSTS PROFESSIONAL ON MAG TAPE, IN EITHER RNO, PIP OR WORD-11 FORMAT.

Eighty percent of this issue of the RSTS Professional was transmitted via telecommunications from author's mag tapes to phototypesetting equipment and was not retyped.

Scenes from the UK RSTS Meeting

CBM/RDM

San Diego DECUS-1980

The Hotel Circle complex in San Diego anchored by the 1000 room Town and Country Hotel make a fine setting for any large meeting. And we were large: estimated at 4000 DECUS attendees representing most (all) of DEC's customer groups. The Meetings began with a seminar day on Monday and continued with traditional meeting style until Friday.

Travel was unusually exciting this year as PSA, the main California Airline, was on strike, making the L.A. to San Diego trip sometimes difficult and always more interesting. The Town and Country lived up to its reputation of being an expert convention center. You can tell pro's when they can serve 4000 lunches outside surrounding the swimming pools, in under two hours without any cross words.

Alas, gone are the days when a RSTS person could stake out a chair in one of the large rooms and spend four days hearing about RSTS. For some reason the meetings were all over. Maybe a consequence of all the "layered products" now on RSTS. There also appeared to be holes in the program for some of us, yet many concurrent meetings where we wished we could be in two places at once. Question: Why can't we get tapes of all speakers? Answer: Legal. But: NCC does it all the time. Answer: silence. Seminars are hard to schedule, particularly if you have 20 different groups you are trying to please; in retrospect maybe it wasn't so bad after all, then again???

This year the sessions went far into the night every night except for the reception held on Wednesday. One well known paper on "How to get more out of RSTS" went until past midnite. Several sessions were cancelled outright when speakers either didn't show up or were unprepared; inexcusable! All DEC prepared sessions were held during the day and the overflow from the user community was held at night. If you go to future meetings (and we suggest you do), plan to spend long hours at the meetings themselves no matter where they are held. This is not a vacation. It is work and well worth it. Speaking of work, my sleeper of the year was Nancy Roth from Al Saloky and his performance measuring group. She gave a dynamite presentation on performance measurements on an 11/34, 11/44 and 11/70. In a nutshell she said 16 jobs for 11/34, 32 jobs for 11/44, and 48 jobs for an 11/70 each reasonably configured. After that many jobs performance degrades rapidly. "How", I asked, "can the user community get more information like this, and how does the 11/780 and 11/750 stack up in this test environment".

Let's hope we'll hear more in Miami. See you there!

RSTS DISK DIRECTORIES, Part 4

By Scott Banks, Systems Design

3.1 Take your shoes off . . .

The response to this series of articles has been very pleasing to me. I'm delighted that so many RSTS professionals are following my efforts. Yes, efforts. The most difficult phase of each production, believe it or not, is the creation of these introductory paragraphs. I am so afraid of repeating myself. And there is always the danger of getting too formal. At least, let's be comfortable. (It's interesting to note that no one has ever asked me to take my shoes off - more than once). Be all this as it may, the opening message is the same:

Welcome to the fourth article of the series. By now, it has become impractical to reproduce all the fine figures that have appeared so far. Please consult the RSTS Professional, Vol 1, No. 1, p. 30; Vol 2, No. 1, p. 45; and Vol 2, No. 3, p. 38 for background information. As soon as I catch my breath we'll dive into the MFD structure.

3.2 The Beginning of the Rainbow

To RSTS, there are only three areas on a disk that are of physical importance. Aside from this trio, any one device cluster is just as good as any other. Because each of these is a starting point (for somewhat independent processes), each has a specific physical DCN.

The boot block is always the first block of any RSTS disk. For those devices that have device clustersizes greater than one, e.g. RP06, DCN #0 is reserved for the boot block, with the remaining blocks of the cluster zeroed and unused. As long as we're on the subject, the boot block is the first step in initiating RSTS. It contains a list of some clusters belonging to [0,1]INIT.SYS and just enough disk driver code to bring them into memory. Then INIT.SYS gets himself together and eventually RSTS is up. The boot itself arrives in memory by only a handful of instructions, permanantly wired into the PDP-11 and only smart enough to read in one block from a disk or tape device. This multilevel initialization procedure is universally known as a 'bootstrap' load, and is employed by other disk-based systems. Can you pull yourself up by your own bootstraps?

In good conscience, I cannot completely ignore the pack serial number. It is stored way out near the end of the disk, in a far away place written to (usually) only by the factory. Every disk of the same type has it in the same offbeat place. Great. Now forget it.

Finally, we arrive at the Master File Directory. The first cluster of the MFD is synonymous with device cluster #1. Should the MFD have more than one cluster, the others may be located anywhere on the disk. The MFD has a File Directory Cluster Map, disclosing its clustersize and the location of all its clusters. Once the first block of the MFD is read, it is pure routine that finds the remainder of the MFD, the UFDs, and all the data files.

3.3 The MFD Label Entry

As I have stated to excess, the MFD is actually a UFD with some enhancements. The first difference we encounter is that the MFD Label blockette contains information regarding the entire disk pack. Figure 1, the MFD LB, has words 0 through 3 defined exactly as a UFD LB. Word 0, just as before, points to the first Name blockette in the MFD/UFD (or is null for an empty UFD). For the MFD, word 4 contains the pack clustersize, which is greater than or equal to the device clustersize, but never greater than 16. Do not confuse the pack clustersize with the MFD clustersize. The MFD clustersize is subject to all the rules of UFD clustersizes and must lie between the pack clustersize and 16, inclusive.

The pack status is stored in word 5. Four bits of the pack status word have meaning, as figure 2 suggests. Bit 15 is the'mounted' bit. When a pack is logically mounted, it is set. When the pack is logically dismounted, it is cleared. Everything is fine unless there is some sort of irregular dismount, such as a system crash. In this case, when the pack is mounted, the bit is found to be already set. Since RSTS will not allow such a pack to be properly mounted, a CLEAN is forced. INIT's CLEAN, Online CLEAN, and the UTILTY CLEAN command (via a system call) are the only automated means provided to clear a bit left set by catastrophe. I could easily devote one article (minimum) to CLEANing. The various CLEANs check, rebuild, and carefully adjust the internal structure of a RSTS disk. Bit 15 of the pack status is cleared by a proper dismount (which means the disk is OK) or by a CLEAN (which tries to make it OK). This scheme provides a realistic degree of crash protection. The public/private nature of a pack is determined by status bit 14. For both system and private packs, this bit is one. Only for public, non-system, disks is it zero.

Bit 12 flags the New Files First option. When this bit is one, it means that NFF was selected during DSKINT. There are certain environments that can benefit from the placement of new files at the top, rather than the bottom, of the directory. Note that this top versus bottom jazz is just a matter of how the file entry is linked. NFF immediately generates more disorderly link paths, in return for quicker access to recently created files.

When bit 11 is set, the Date of Last Write option has been selected at DSKINT time. For data files, a Date of Last Access is maintained. When the file is opened, the current date is written there. But with the DLW option selected, the current date is written in only when the file is opened with write access available.

The pack ID words, 6 and 7, contain six RAD50 characters. You specify what you want here when you initialize the disk, and surrender the name each time you mount the pack.

3.4 The MFD Name Blockette

The MFD contains a list of all the UFDs. Each UFD entry, like a data file within a UFD, is marked by a Name blockette. Of necessity, the NBs used in the MFD to describe UFDs are different than data file NBs. Figure 3 shows the layout of an MFD NB. These NBs are linked together in the same fashion as we discovered in the UFD. The change is merely the interpretation of their contents.

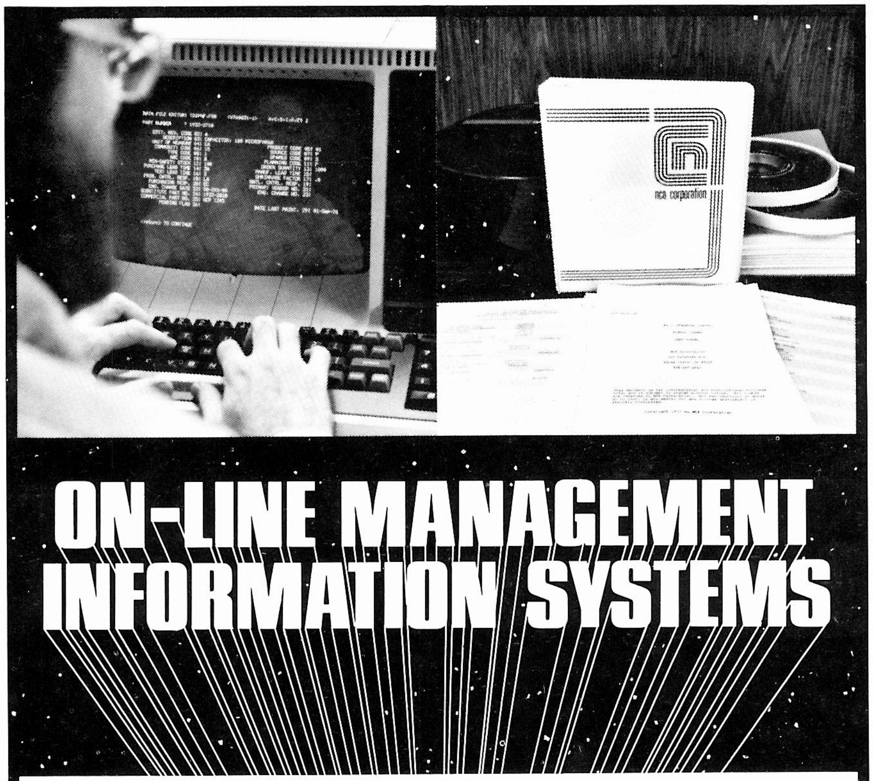
There is a great opportunity here for a confusion of terminology. When I refer to a UFD NB, I mean the Name blockette of a data file. We will see later that UFD NBs can appear in both UFDs and the MFD. But an MFD NB can only appear in an MFD, and describes a UFD entry. Hang in there.

The link to the next NB is again word 0. Word 1 contains the project number and programmer number (PPN) of the UFD it represents. The password is stored in words 2 and 3, via RAD50. Word 4, as in the UFD NB, contains the status and protection code bytes for the entry. The access count, word 5, is also maintained as two bytes. The high byte is a running count of the number of users currently logged-in the account of this PPN. The low byte contains a similar count, but based upon the number of I/O channels that have the UFD open as a file.

Word 6 is a link to the Accounting blockette for this entry. We will visit the MFD AB shortly. The first cluster of the UFD is revealed by word 7 of the NB. There are no Retrieval blockettes for UFDs. The UFD itself, as you recall, has a File Directory Cluster Map to keep track of its own clusters. (The MFD, being a good little UFD, has an FDCM too). If the UFD was just created with REACT, and no files have yet been saved there, it has no length and word 7 will be zero.

Now the plot thickens. The MFD happens to have UFD NBs as well as MFD style NBs. Remember that it is the UFD for account [1,1]. This takes us back to word 4, the status bits. If bit 6 is zero, it is a UFD NB and represents some data file. The AB and RBs have the exact meaning that they would have in any UFD. But, if bit 6 is one, FIP knows that this NB is to be taken as a UFD entry, along with its AB. The DIRECT program lists only entries with bit 6 clear, allowing it to work perfectly on [1,1].

One last note about MFD NBs. Except for bit 6, there is only one bit in word 4 that conveys true information. If bit 2 is set, write access has been given out to a channel that has this UFD open as a file. The other bits are set to a sensible pattern, the same for all MFD type NBs.



3.5 The MFD Accounting Blockette

For every MFD NB there must exist an MFD Accounting blockette (figure 4). Word 0, to be compatible with the UFD structure, contains a link to an Attribute blockette. For MFD Accounting blockettes, this link will be null (currently, only data files may have attributes).

Let's skip down to word 7, the UFD clustersize. When REACT is used to create a new user account, there is no physical UFD at first. REACT uses a SYS() call to add an MFD entry. The users specifies the PPN, password, disk quota, and desired UFD clustersize. The former two items reside in the MFD NB, while the latter reside in the AB. When the first user data file (for the new account) needs a place to go, FIP locates a free device cluster and hooks it all together. At this point, the DCN is written to the NB, and the first cluster of the UFD is formatted.

The disk quota, word 6 of the MFD NB, was derived in the manner described above. It comes into play when LOGOUT runs. If the quota is non-zero, LOGOUT demands that the user not exceed his quota, lest he not be allowed to leave the system (logically, not physically). A zero quota instructs LOGOUT to bypass this check, implying an infinite quota.

Also in the MFD AB we find information that is useful for the management of a timesharing system. The accumulated CPU time, in tenths of a second, is stored in words 1 and 5. The entire 16 bits of word 1 and the upper 6 bits of word 5 form a 22 bit counter. The console connect time, in minutes, is in word 2. RSTS keeps track of kilo-core-ticks using a 26 bit counter composed of word 3 and the low order 10 bits of word 5. Simply, for every tenth of a second of run time, a KCT is added for each 1K of memory used by the job.

Word 4 keeps track of device time, in minutes. Aside from keyboard and normal disk usage, device time can be charged on the basis of this accumulator. Unfortunately, the times for all such chargable devices are shmushed together here.

The MONEY program, using a SYS() function, displays most of the information contained in the MFD NB and AB. The statisical data is maintained in memory, separately for each job. At appropriate points, such as LOGOUT or DETACH, some MFD Accounting blockette information is updated.

3.6 Sample MFD Access Program

I have taken last issue's program, the simplified directory lister, and reworked it. This version exposes the MFD. The major difference is the early detection of NB type. For UFD entries (those with MFD style NBs) I print the PPN and the first cluster. For data files, the filename and first cluster are displayed. In those cases where there is no first cluster allocated, I detect this fact and leave a blank. If you run this program, you will notice that the first thing it prints is account [1,1] with a DCN of 1. That, of course, is the MFD. It appears in itself as a UFD . . . because it is.

Like the previous example, this program cannot damage your system. Never open a UFD with mode 16384, as that might allow writing to occur. Feel free to experiment with your own variations.

0	LINK TO FIRST NAME BLOCKETTE
1	-1
2	0
3	0
4	PACK CLUSTERSIZE
5	PACK STATUS
6	PACK ID (RAD50)
7	

FIGURE 1. MFD Label Blockette

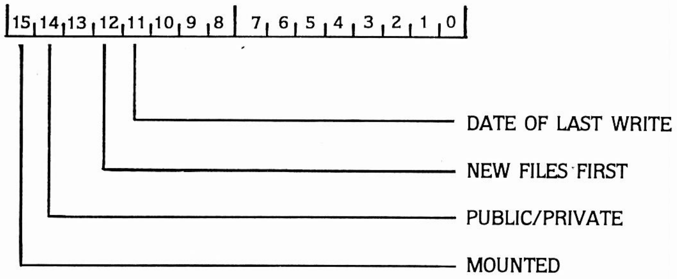


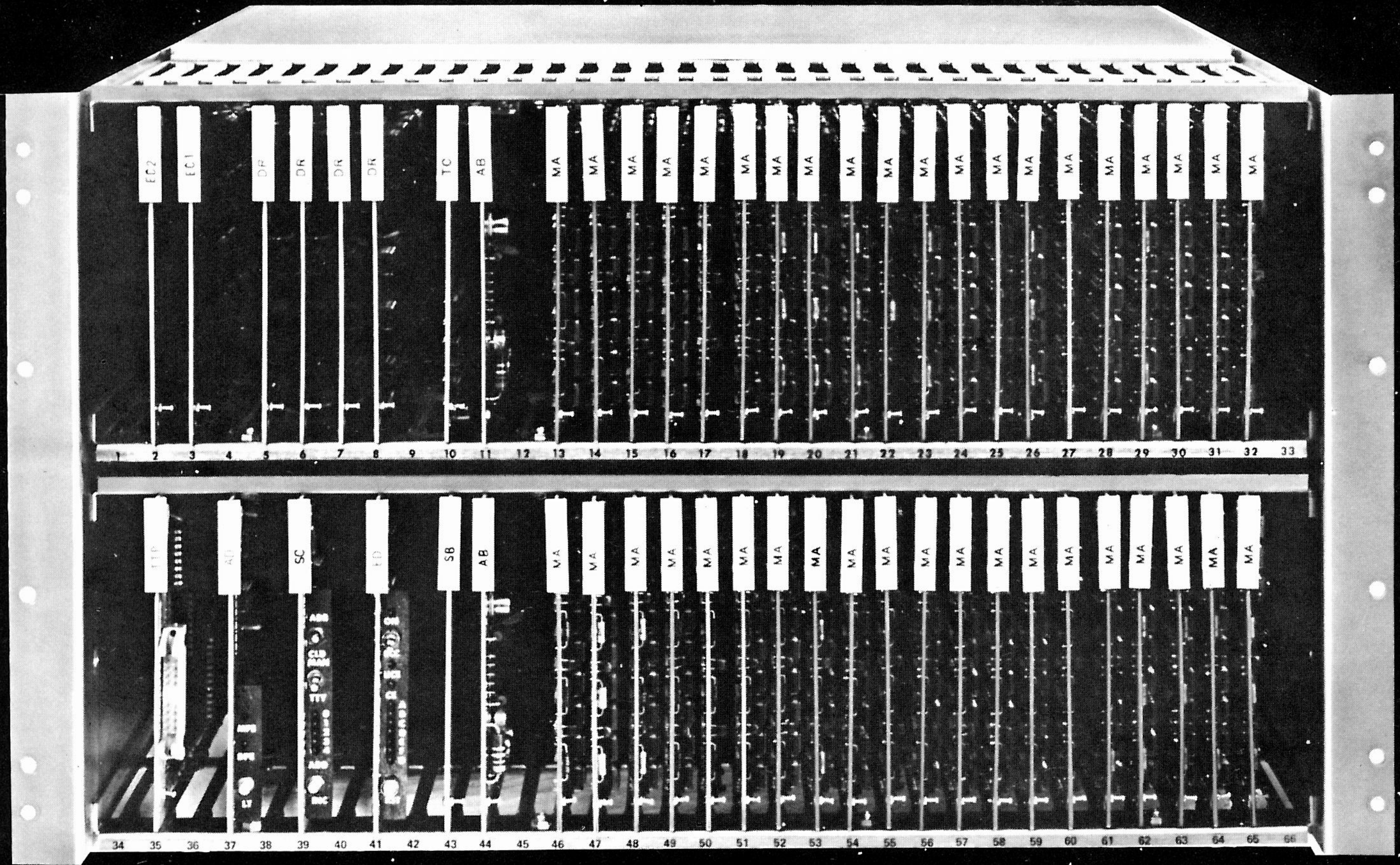
FIGURE 2. Pack Status Word

0	LINK TO NEXT NAME BLOCKETTE	
1	PROJECT #	PROGRAMMER #
2	PASSWORD (RAD50)	
3		
4	PROTECTION CODE	STATUS
5	LOGIN COUNT	UFD OPEN COUNT
6	LINK TO ACCOUNTING BLOCKETE	
7	DCN OF FIRST CLUSTER OF UFD	

FIGURE 3. MFD Name Blockette

0	LINK TO ATTRIBUTE BLOCKETTE	
1	LSB CPU TIME	
2	CONNECT TIME	
3	LSB KCT	
4	DEVICE TIME	
5	MSB CPU TIME	MSB KCT
6	DISK QUOTA FOR LOGOUT	
7	UFD CLUSTERSIZE	

FIGURE 4. MFD Accounting Blockette



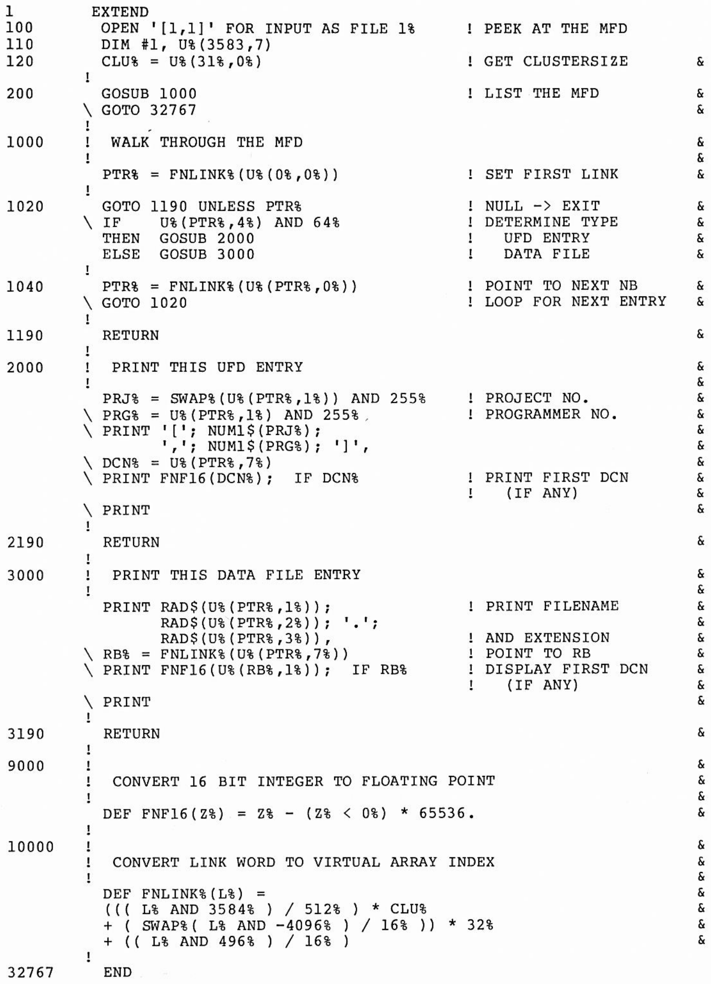
The MSC 3602 Intelligent Add-on Memory for the PDP-11/70

Capacity:	*First 10½″ chassis 2048KB in 64KB increments Second 10½″ chassis 2048KB in 64KB increments.*
Cycle time:	*690 ns*
Access time:	*500 ns*
Error Correction Circuitry (ECC):	*Single bit error correction Double bit error detection*
Power requirements:	*3.5A maximum at 115 VAC ±10V, 60Hz.*
Battery requirements:	*+5 and +12V, capacity determined by system size and required backup time.*
Operating environment:	*Operation—0°C to 50°C Storage— -40°C to 100°C Humidity—90% maximum without condensation.*
Physical dimensions:	*10½″ H x 20″ D x 19″ W standard RETMA rackmount or freestanding.*

Memory elements:	*Socketed 16K MOS RAM.*
TTP Micro Computer:	*6802 microprocessor based interface card with 26-pin flat cable connector. Applicable to terminal devices utilizing current loop, EIA or RS-232 standards.*

Monolithic Systems corp

14 Inverness Drive East
Englewood, CO 80112
303/770-7400
Telex: 45-4498



```
1        EXTEND
100       OPEN '[1,1]' FOR INPUT AS FILE 1%     ! PEEK AT THE MFD
110       DIM #1, U%(3583,7)
120       CLU% = U%(31%,0%)                     ! GET CLUSTERSIZE       &
        !
200       GOSUB 1000                            ! LIST THE MFD          &
        \ GOTO 32767                                                    &
        !
1000    !  WALK THROUGH THE MFD                                         &
        !                                                               &
          PTR% = FNLINK%(U%(0%,0%))             ! SET FIRST LINK        &
        !
1020      GOTO 1190 UNLESS PTR%                 ! NULL -> EXIT          &
        \ IF    U%(PTR%,4%) AND 64%             ! DETERMINE TYPE        &
          THEN  GOSUB 2000                      !   UFD ENTRY           &
          ELSE  GOSUB 3000                      !   DATA FILE           &
        !
1040      PTR% = FNLINK%(U%(PTR%,0%))           ! POINT TO NEXT NB      &
        \ GOTO 1020                             ! LOOP FOR NEXT ENTRY   &
        !
1190      RETURN                                                        &
        !
2000    !  PRINT THIS UFD ENTRY                                         &
        !                                                               &
          PRJ% = SWAP%(U%(PTR%,1%)) AND 255%    ! PROJECT NO.           &
        \ PRG% = U%(PTR%,1%) AND 255%           ! PROGRAMMER NO.        &
        \ PRINT '['; NUM1$(PRJ%);                                       &
                ','; NUM1$(PRG%); ']',                                  &
        \ DCN% = U%(PTR%,7%)                                            &
        \ PRINT FNF16(DCN%);  IF DCN%           ! PRINT FIRST DCN       &
                                                !    (IF ANY)           &
        \ PRINT                                                         &
        !
2190      RETURN                                                        &
        !
3000    !  PRINT THIS DATA FILE ENTRY                                   &
        !                                                               &
          PRINT RAD$(U%(PTR%,1%));              ! PRINT FILENAME        &
                RAD$(U%(PTR%,2%)); '.';                                 &
                RAD$(U%(PTR%,3%)),              ! AND EXTENSION         &
        \ RB% = FNLINK%(U%(PTR%,7%))            ! POINT TO RB           &
        \ PRINT FNF16(U%(RB%,1%));  IF RB%      ! DISPLAY FIRST DCN     &
                                                !    (IF ANY)           &
        \ PRINT                                                         &
        !
3190      RETURN                                                        &
        !
9000    !                                                               &
        !  CONVERT 16 BIT INTEGER TO FLOATING POINT                     &
        !                                                               &
          DEF FNF16(Z%) = Z% - (Z% < 0%) * 65536.                       &
        !
10000   !                                                               &
        !  CONVERT LINK WORD TO VIRTUAL ARRAY INDEX                     &
        !                                                               &
          DEF FNLINK%(L%) =                                             &
          ((( L% AND 3584% ) / 512% ) * CLU%                            &
          + ( SWAP%( L% AND -4096% ) / 16% )) * 32%                     &
          + (( L% AND 496% ) / 16% )                                    &
        !
32767     END
```

FIGURE 5.

Coming next issue . . .
A Trip with FIP

A RSTS PERFORMANCE CHECKLIST . . .

By Dave Mallery

HARDWARE:

For 34's I have maximum memory. For 70's and 44's, I have enough memory to reduce swapping to below 1%. I know that the addition of an FPP to a RSTS PDP-11 is a good idea. I know that RM's are faster than RP's in data transfer (on 11/70's) I know that DH's are preferrable to DZ's and DMAX's are preferable to DH's because of lower unibus and processor overhead on output and a mix of DH/DZ uses small buffer space. I know that parallel line printer interfaces are better than serial for local devices because they reduce small buffer loading. I am aware of the availability of 6250 bpi drives from foreign vendors.

During DSKINT:

I chose an efficient pack clustersize for my installation. I centered SATT.SYS I don't use NFF except in very special circumstances and REORDR daily. I have only one public volume.

During SYSGEN:

I generate just one or two jobs above my expected job max. In general, for 70's, the larger the monitor is, the better. I generate a minimum number of PK:'s because each PK: uses two DDB's. I also generate an optional monitor without statistics. I know how to disable and reset the statistics with the console switch register. I reduce the number of small buffers for both input and output. (3.3.41 September 79). I selected all resident monitor options. I selected overlapped seek for multiple drive installations. I selected the "large FIP" option of V7.0 even though I don't use large files.

After SYSGEN:

I centered the swap files during refresh. I create all my UFD's contiguous in the center of the disk and pre-extend them to needed size. I use a UFD clustersize of 16, but only enough clusters in each directory. I have enough XBUF. I change the cache replacement time to 10 seconds. (3.1.3F-Sep79) On large 11/70's, I try the first-fit memory allocation patch. (3.1.2F Sept 79)

I carefully select which run-time systems shall be resident. My major production files are contiguous and placed near the center of the disk. I make a conscious effort to place my files on separate drives to produce load balancing. When installing performance-related patches, I always do them one at a time and evaluate the net effect before proceeding. I tune my memory usage.

CUSPS, ETC.:

I keep LOGIN.BAC or TSK at the top of [1,2]. I have evaluated a dynamic priority scheduler for my installation. If I selected a DYNPRI, I keep the most important and busy files in my machine open in it. (Like LOGIN.BAC). I use CSPCOM or BASIC2 for the spooling package and major cusps, and use the separate library feature. I am aware of the availability of utilities (like SYSTAT and DIRECT) in Macro. I use SETUTL (RSTS PRO, September 1980). I consider Basic + 2 or BAC-MAC. I don't use $BACKUP. I use TECO but not VTEDIT. I use UTILTY.SAV. I use a fast sort suited to my environment. On 70's, I have my PIP CCL set for large size (8192 + 28).

During Operation:

I have $REORDR (in B+2 or CSPCOM). I understand it. (10.20.1N Sept 79). I use it at least weekly. I keep busy accounts' UFD's short - using other accounts and the # feature. I use the correct clustersize when opening files -I always use clustersize 256 for contiguous opens. I place the busy files at the top of the directory. (MODE 1552 = contiguous and top of directory.) I police my users: - I don't leave them logged-in doing nothing - I lower their priority when naughty. Since RSTS performance degrades rapidly as any of its limits are reached, I make sure there is always a surplus of disk space and a few empty job slots. I am aware of the new efficiencies in virtual array usage. (V7 Release Notes p. 2-12). I keep an eye out for'killer' 16K B+ programs that spend all their time doing garbage collection - I understand strings. I am aware of excessive string manipulation and re-code such programs for LSET & FIELD if possible. I don't try to run mixes of jobs that swamp the system. I schedule busy batch to the wee hours. I keep an eye on bad tapes and run-away modems.

In my Software:

I optimize directories by using optimum file clustersizes, pre-extending new files and making files contiguous. I don't delete and create scratch or temporary files - leave them there.

WELCOME AGAIN TO MACRO LAND

By Bob "MACRO MAN" Meyer

In the past few articles, we've covered basic Input & Output to the user KB: with 'in-line' assembly language code. Those of you who took the time to try out the demo programs surely realized on thing: thats a lota work to get something out to my CRT! And you're right; it wouldn't be very practical to write 10-15 lines of Macro code each time you wanted to print or input something to or from your KB. This time around, we're gonna see some easier ways to do the same terminal-type I/O we performed in the last issues. This may not be QUITE as exciting as issuing monitor calls or playing with your FIRQB, but what we're about to get into makes Macro programming a lot easier. If you'll turn your attention to the program 'DEMO', (figure one), you'll see the comment:

```
;DEFINE' PRINT' Macro
```

followed by 4 lines of assembly code (P.S. those who know how to create & use Macros may wish to skip some of this . . .).

This is called a Macro DEFINITION.

When the assembler sees this, it 'remembers' the keyword 'PRINT', and saves the code between the Macro definition and the '.ENDM PRINT' statement somewhere in its internal workings. From there it continues on with the remainder of the assembly. Now when it runs across a statement like the one at the label 'DEMO:', the assembler plops a copy of the code from the definition inline with the users code. Obviously, this saves you the effort of typing the blasted thing in over again, but it also places the ARGUMENT you specified ('#PROMPT' in this example), in place of the DUMMY ARGUMENT in the Macro's definition ('MSG'). This turns out to be very handy, as you'll see in the next few examples.

You may recognize the program segments in Figs. 2 & 3 as the print and input routines we put together the last few times. (For details on these, see the RSTS 'PRO', Vol. 2, No. 2, and Vol. 2, No. 3.) The fourth example is something I call RSXCOM.MAC. It's purpose in life is to globally define any symbols that I plan on using in other Macro programs. As you can see, it contains the XRB, FIRQB, .READ, .WRITE and a few others. This is somewhat similar to COMMON.MAC, except that it's a bit smaller for our purposes, and this one gets wired in at Task Build time, rather than assembly time. Because these symbols are 'global', we can make reference to them in other modules (like PRINT or INPUT), and TKB will put 2 + 2 together and make everything work.

These guys could be put together as follows:

```
MAC
MAC>DEMO=DEMO
MAC>PRINT=PRINT
MAC>INPUT=INPUT
MAC>RSXCOM+RSXCOM
MAC>↑Z
```

Then, assuming no assembly errors occurred, you would Task Build something like this:

```
TKB
TKB>DEMO,DEMO/-WI=DEMO
TKB>PRINT
TKB>INPUT
TKB>RSXCOM
TKB>//
```

Now you'll have DEMO.TSK, the executible image, and DEMO.MAP, a memory allocation map of all the goodies in your task. (Take a look at the map; it'll probably make sense to you after a few beers, and will become VERY important when we get into debugging in the next issue).

Let's follow the flow of this thing a bit.

Starting with DEMO:, the transfer address of the task, (because of .END DEMO at the bottom), we issue a Macro CALL, PRINT. This expands (with help from MAC), into the following:

```
MOV          #PROMPT,R0
JSR          PC,PRINT
```

Now the ADDRESS of the string PROMPT is loaded into R0, and a jump to subroutine (kinda lika a gosub . . .) is done to your PRINT module. The PRINT routine simply loads the XRB with the required parameters to print to the KB: and 'plugs in' the address of the string PROMPT. (Please note that for the sake of simplicity, NO error checking is being done here...we'll get to that in the future). Next, the INPUT Macro is called. Again, the Macro is expanded and looks like:

```
MOV          #RESP,R0
JSR          PC,INPUT
```

So, the ADDRESS of the KB buffer RESP is loaded into R0, and a JSR is done to the INPUT code. INPUT sets up the XRB for a .READ from the current KB, into the buffer area RESP. Right after the .READ we find the end of the string, and tack on a zero byte at the end. This makes the string easier to print out or work with later in the program, as the zero byte indicates the end of the string, so there's no need for any string header. (Not the greatest way of doing things, but by far the easiest.)

Next we do some LIMITED error checking; such that if ANY error occurs, we can let the calling program know that something's not right. What we're doing is using the CARRY bit of the PSW as an error indicator: if carry is clear, all's well; if it's set, some error (the exact error can be found in the FIRQB) has occurred.

Now we bounce back to the calling program, and check the CARRY bit for any problems. (This is a quick way to watch for ↑ Z, in our demo situation). In the event of any error, we exit the DEMO program right away; otherwise, we print back the user's original string, with some quotes and things before and after it, and do it all over again.

As you add more modules to your bag of goodies, you'll most likely want to put them all in an OBJECT LIBRARY. One reason for this is that you're gonna get tired of those long TKB command lines; but more important, Task Building from an object library improves TKB time quite a bit. The following commands to the LIBRARIAN will do nicely:

```
LBR
LBR>LIB/CR                          ;This creates the library on disk
LBR>LIB/IN=PRINT,INPUT,RSXCOM       ;This inserts modules in the library
LBR>↑ Z
```

To get a gander at what's floating around in your lib, type:

```
LBR LIB/LI
```

There are a handful of other options; they're all documented in V7.0 manuals.

Your Task Builder command line can now be changed to:

```
TKB
TKB>DEMO,DEMO/-WI=DEMO
TKB>LIB/LB                          ;This tells him to look for 'LIB.OLB'
TKB>//
```

And for those of you who REALLY like to have fun, here's how to put our MACRO modules (PRINT,INPUT,ETC . . .) into a RESIDENT LIBRARY:

```
TKB
TKB>LIB/-HD,LIB,LIB=PRINT
TKB>INPUT
TKB>RSXCOM
TKB>/
ENTER OPTIONS:
TKB>STACK=0
TKB>PAR=LIB:160000:20000
TKB>//
```

I suggest diving into the Task Builder Reference manual if you're interested in details on these TKB options. Briefly, the /-HD says that the task will have no HEADER, (and therefore is not runnable), which is fine for our purposes. The STACK parameter says that we want NO stack space in the library (the default is 256 words if we don't specify), and the PAR options tells TKB that we intend to create a partition called LIB, which will map our virtual workspace starting at address 160000 (that's APR 7, by the way, so you'll need the 'disappearing RSX' option in the monitor) and has a maximum length of 20000 bytes (octal, of course).

The last step is to run the task image & symbol table through MAKSIL, and we'll have a usable resident library.

```
RUN $MAKSIL
<header>
Resident Library name? LIB
Task-built Resident Library input file <LIB.TSK>? <CR>
Include symbol table (Yes/No) <Yes>? <CR>
Symbol table input file <LIB.STB>? <CR>
Resident Library output file <LIB.LIB>? <CR>
LIB built in 1 K-words, 9 symbols in the directory
LIB.TSK renamed to LIB.TSK<40>
```

(The MAKSIL.BAS program on your distribution kit is VERY well documented; you may want to refer to it for info.)

You'll now find LIB.LIB in your account. To put the library up for use type:

```
UT ADD LIBRARY [ppn]LIB/ADDR:<some memory address>
```

Specify the PPN that the library exists in, and find some free memory (1K will do nicely), and specify that address for the library to load.

Finally, re-build the DEMO task to link up with the new resident library like so:

```
TKB
TKB>DEMO=DEMO
TKB>/
ENTER OPTIONS:
TKB>RESLIB=LIB/RO
TKB>//
```

The RESLIB option will cause good ol' TKB to place .PLAS directives into your task image, which will attempt to map the resident library LIB at Run-Time. The /RO means you want read-only access. incidently, if you haven't added LIB with UTILTY, RSTS will give you hell with the message:

```
?Can't find file or account
```

and your task will not run.

Well, that oughta keep ya off the streets for a few hours . . . (Don't let your boss catch ya doin' this stuff until you can think of a practical use for it! (There are many!!))

Since our demo programs are getting a bit elaborate, we're undoubtedly going to run into some Buggies. So next issue we'll cover a dynamite piece of code called ODT (Octal Debugging Tool).

See ya then!

Figures 1, 2, 3 and 4 are on page 93.

SOME EASY TECO MACROS

By Dave Mallery

The listing of TECO.TEC presented here illustrates a group of simple macros that have helped us upgrade BASIC + syntax here at N.D.D. They provide an excellent tutorial for the intermediate Teco user.

When one issues the TECO CCL, what is actually invoked is a TECO.TEC file, containing commands to be interpreted by the TECO run-time system, much in the same manner as a .BAC file is interpreted by the BASIC RTS. This version of TECO.TEC starts by printing the message:

"For help, type MH$$"

You can see this message, enclosed by slash delimiters as the second executible line of TECO.TEC.

The operator that causes the type-out is @ ↑ A

The ↑ A is a normal type-out command, the @ states that the text will be contained within delimiters - in this case, slashes. Whatever character follows the A will be the delimiter.

The next line of TECO code sets up a command string into Q register H. Q registers are available in TECO to hold strings and numeric values.

Looking at the line:

@ ↑ UH% - put the stuff within the delimiters (%)
into register H.

What goes into register H is:

@ ↑ A/ML

and the next 10 lines thru

/% but not the final %.

Notice that the content of Q register H is now a command string of the same format we covered earlier. The difference is that this string will be executed any time the user types MH$$ — that is, when the user executes the"MACRO' in Q register H.

We will now skip down the page (over a group of screen filling macros). Notice that the macro ML includes in its string an invocation of the macro MT.

Now we will decipher a few of the macros for upgrading BASIC + code.

MC will take a line like:

```
100 X = Y:LSET R$ = X$
```

and convert it to:

```
100 X = Y
\    LSET R$ = X$
```

Notice that I don't bother with ampersands. They are handled by invoking:

TEC XYZ.BAS/72

When so prompted, teco strips all line-terminating ampersands coming in, and replaces them correctly in col. 72 (or higher) on output.

Now, lets examine the MACRO:

@ ↑ UC% says put the string within %'s into reg C.

The command string itself is

@ FS!:!C/R \ tab!V

change a colon into C/R tab and display it. Note that ! is the delimiter for the @FS command.

Macro MM addresses the problem

```
100  X% = 100%:
     Y% = 200%
```

If we simply do an MC, we are left with an extra CR/LF (the one after the colon). To be rid of it, the command string saved in register M is:

MC$KT

Where KT blows away the dangling CR/LF.

The other macros are simple variations on these themes. You can try your own flavors yourself by making a differently-named copy of TECO.TEC and adding your own CCL for testing. Any one macro can be loaded into a Q register by simply typing it in. It's just like immediate-mode BASIC!

The commands that wrap up TECO.TEC are housekeeping for shutting down the command file (@EI) and getting further input from the user KB. After handling the possibility of "can't find file or account" the macro finally exits and executes an MT which displays 20 lines on the screen and exits.

In summary, what this TECO program TECO.TEC has done is:

a) Prompt for a HELP message.

b) Set up command strings (macros) in a number of Q registers for possible use by the user

c) Handle openings and reading in the file specified by the user in the CCL invocation.

d) Display 20 lines of the file.

e) Exit to the user. ie: let the user enter his or her own teco commands or invoke the command strings now in the Q register.

```
!----------------------------------------------!
!        Responder to TECO CCL commands        !
!----------------------------------------------!

[0

@^A/FOR HELP, TYPE MH$$/
! Set up System-wide Macros !

! Help message - MH !
@^UH%@^A/ML- advance and display page
M2- Verify lines above & below the Dot
MT- display page
MC- change embedded colon to cr-\tab
MM- change colon at eol
MN- change \ at eol
MB- change embedded \
MP- change & to PRINT
MU- insert , after #´
MA- insert , after percent
/%
!MT- Display 1 page (20 lines), clear only if scope !
@^UT% 2&ET"N 155^T 72^T 155^T 74^T 140^T´ 0L 20T %

!m1 -- clear screen & splat some lines!
@^U1/ 155^t ^AH^A 155^t ^AJ^A 140^T 01 20t/

!mq -- skip some lines & do an ´m1´!
@^uq/ 151 m1/

!M2- DISPLAY some LINES ABOVE & BELOW THE DOT!
@^U2´ 155^T ^AH^A 155^t ^AJ^A 140^T 0L @I/->/ 0L 11v 2D´

!ML- Advance a page, display next page !
@^UL% 20L .-Z"N MT´%
!MC- fix an embedded colon to space <cr> \ <tab> !
@^UC%@FS!:!
\       !V%
! MB fixes embedded \ to cr \ tab !
@^ub%@fs#\#
\       #v%
! MM does MC$kt$$       for lines ending in : !
@^UM%MC$KT%
! MP does & to print conversion !
@^UP%@FS/&/PRINT/V%
! MU does print using comma insertion after #´ !
@^UU%@S!#´!@I!,!V%
! MN CHANGES TRAILING \ TO CR \ !
@^UN%@FS!\!
\!KT%
!MA- insert comma after percent !
@^UA#@S!%!@I!,!V#
! Shut off this indirect command file           !
@EI%%
! Is there any command there at all?            !
Z"E @O!DONE! ´
! Is the CCL command ´TECO´ ?                   !
J ::@S%TECO%"S @FR%% <@FS/^ES//;> Z"E @O/DONE/´ HXO HK :@EB/^EQO/"S Y @O/DONE/´
@^A/%Can´t find file or account "/ :GO @^A/"
%Creating new file
/ @EW/^EQO/ @O/DONE/ ´

! We could not recognize this CCL command       !
@^A/?Unrecognized CCL command "/ HT HK @^A/"
/ ^C
!DONE!
]O Z"N MT´
$$
```

HARDWARE INDEPENDENCE USING RESIDENT LIBRARIES

By Dan Esbensen & Dave Kachelmyer,
North County Computer Services, Escondido, CA

ABSTRACT

With the advent of RSTS/E V7.0, resident libraries provide a simple solution to the problem of sharing software between machines that have floating point hardware and those that don't.

INTRODUCTION

At our site there are several computers of varying configurations, some with floating point hardware (FPU) and some without. Moving software between computers requires re-building the programs to add or remove FPU support, unless the software is run without FPU support on all machines. Rebuilding the software is quite time-consuming and requires several sets of software to be kept on the development machine.

The resident library feature of RSTS/E was used to solve this problem. Two resident libraries were developed, one with and one without FPU hardware support. These libraries have identical names and entry points so that tasks built against one may run with either resident library without modification. Once the proper library has been placed on each machine, the same copy of the software may be run on either type of machine without re-building the task.

TASK BUILDING AND RESIDENT LIBRARIES

The following is a brief description of task building and resident libraries. It is intended as an aid to understanding the sections that follow, and is not a thorough description of either task building or resident libraries. Those already familiar with these topics may skip to the next section.

In general, before a program can be run, it must be converted into machine instructions for the computer to use. This is usually a two-step process. The first step converts the source program to intermediate code. The second step converts the intermediate code into a form executable by the computer.

The first step is performed by a language compiler (such as BASIC-PLUS-2, COBOL, or MACRO). The compiler builds an object module which contains data and information for assembling the program.

The second step is performed by the Task Builder. The Task Builder converts the object module made by the compiler into a task image file. The task image file can then be loaded into memory and run by the computer.

During the task building process, the Task Builder merges modules referenced by the program into the task image. These modules may be subprograms written by the programmer, or modules from some common library of routines.

For high level languages such as BASIC-PLUS 2 or COBOL, a common library (called the "Object Time System" library) contains the routines which perform the functions supported by the language. Such functions might be adding numbers, calculating cosines, or printing to a terminal. In some cases, such as floating point math, the routines are quite large (several thousand bytes). The finished task image may be much larger than the original module built by the compiler.

Since many of the OTS library routines are common to all programs, there are copies of these routines in memory for each program that is running. Much memory can be tied up with all the copies, keeping this memory from being used for additional programs. Resident libraries are used to avoid this problem.

A resident library allows a copy of a group of routines to be shared between programs. This is done by placing one copy of the routines in a block of memory, and making this block a part of a program's memory image.

The resident library build process produces task, symbol table, and library image files. The task and symbol table files are used in building programs against the resident library. The library image file contains the resident library itself. Once the resident library is added (similar to adding run-time systems), the resident library may be used.

A program must be built against a resident library in order to use it. This is done by including a RESLIB directive in the program's task build command file. The RESLIB directive causes the Task Builder to build references to the routines in the resident library, rather than copy routines from an object module library. Since the routines from the resident library are not included in the task, the task image is smaller than it would be otherwise.

When the program is run, it automatically attaches to the resident library. This makes the resident library part of the program's memory image. The resident library is then used as if it were part of the program.

MAKING TASKS HARDWARE INDEPENDENT

In the following discussions, code written for floating point hardware will be referred to as FPU code. Code written for machines with only the extended instruction set will be referred to as EIS code.

It is possible to make a program that senses the presence of floating point hardware (RSTS does this in the INIT code) and automatically adjusts to use FPU or EIS routines. However, such programs would necessarily be large and complex, containing routines for both types of configurations. A more practical solution would be to isolate the EIS/FPU code from the program and use either EIS or FPU routines as required by the hardware configuration. The EIS/FPU code can be isolated by placing it in a resident library. With the proper resident library in place, the same task file will run on both hardware configurations without modification.

TECHNICAL CONSIDERATIONS

Resident libraries can be changed for a given task if the RESLIB name and entry points are identical. This normally requires that the modules in both libraries be the same size so that the entry points match. However, EIS math routines are larger than the FPU counterparts, so entry points would normally be different. The entry points can be standardized, however, by accessing the routines through a dispatch vector. This method uses a table of jump instructions with the same entry point names as the routines being accessed.

The dispatch vector approach requires making separate assemblies for the resident library and dispatch vector structure (to avoid global symbol name conflicts). The symbol table from one assembly is used to build the resident library file. The other symbol table is used when task building programs against the resident library.

BUILDING THE RESIDENT LIBRARIES

The following discussion describes the procedure for building the EIS and FPU resident libraries for use with

Six Stages of Program Development

1. Wild Enthusiasm
2. Disillusionment
3. Total Confusion
4. Search for the Guilty
5. Punishment of the Innocent
6. Promotion of Non-Participants

. . . thanks to Charlie Brown of AMCOR

A prize for the best caption for this photo. Send entries to: RSTS PROFESSIONAL, Caption Contest, P.O. Box 361, Ft. Washington, PA 19034.

BASIC-PLUS 2 programs.

The steps for building the interchangeable resident libraries are:

Select the modules to be included in the library

Build the .MAC, .ODL, and .CMD files

Assemble, task build, and run MAKSIL

Test resident libraries

Module selection involves choosing the double or single precision math routines. This selection should match that of the compiler. These modules may be found in the libraries BP2OLB.OLB (EIS) and BP22LB.OLB (FPU) in account [1,10] on the BASIC-PLUS-2 distribution medium. The module names are listed in appendix A.

Once the math routines are selected, the two modules needed for the dispatch vector should be built. This requires making two MACRO source files, one for the jump vector, another for the symbol table assembly.

The jump vector source file consists of a series of JMP instructions to the math module entry points. An abbreviated version is shown below.

```
.TITLE   BP2RLB BP2 RESIDENT LIBRARY
.IDENT   /07.001/
.DSABL     GBL
.PSECT   BP2OTS,RW,I,LCL,REL,CON
.GLOBL   ADD$MS,ADD$PS,ADD$SS,SUD$MS,SUD$PS,SUD$SS

JMP      ADD$MS
JMP      ADD$PS
JMP      ADD$SS
JMP      SUD$MS
JMP      SUD$PS
JMP      SUD$SS
.END
```

The symbol table source file consists of a series of .WORD directives with symbol names duplicating those of the module entry points. The order of these symbols must match that of the jump vector. An abbreviated version is shown below.

```
         .TITLE   BP2RLB BP2 RESIDENT LIBRARY
         .IDENT   /07.001/
         .DSABL   GBL
         .PSECT   BP2OTS,RW,I,LCL,REL,CON
         .GLOBL   ADD$MS,ADD$PS,ADD$SS,SUD$MS,SUD$PS,
                  SUD$SS
ADD$MS:  .WORD    0,0
ADD$PS:  .WORD    0,0
ADD$SS:  .WORD    0,0
SUD$MS:  .WORD    0,0
SUD$PS:  .WORD    0,0
SUD$SS:  .WORD    0,0
         .END
```

Once the MACRO source files have been built and tested, the task builder Overlay Descriptor Files should be built. Four ODL files are needed, two for the EIS and FPU jump vector assemblies, and two for the EIS and FPU symbol table assemblies. The jump vector ODL file lists the modules to be included in the resident library. The symbol table ODL file contains only the reference to the symbol table module from the previous step. A sample set of ODL files is shown below.

```
;          EIS1.ODL
;           EIS          RESLIB          DESCRIPTION
;
           .ROOT    EIS1-GRP1
GRP1:      .FCTR    GRP1A-GRP1B-GRP1C
GRP1A:     .FCTR    BP2OLB/LB:$BFPER:$DADD:$DATAN:
                    $DCON1:$DDIV:$DEXP:$DFIX
GRP1B:     .FCTR    BP2OLB/LB:$DING:$DINT:$DLOG:
                    $DMUL:$DNEXT:$DSCAL:$DSIN
GRP1C:     .FCTR    BP2OLB/LB:$DSQRT:$DXDI:$FCON1:
                    $FMUL:$JCON1:$MATII
           .END
;          EIS2.ODL
; EIS SYMBOL TABLE DESCRIPTION
;
           .ROOT   EIS2
           .END
```

Once the ODL files have been built, the task builder command files should be made. Four CMD files are needed, one for each ODL file. These files specify the input and output files, and parameters for the task build process. A sample set of files is listed below.

```
;          EIS1.CMD
; TASK BUILD RESLIB FILE
;
EIS/-HD,EIS1,EIS1 = EIS1/MP
UNITS = 0
STACK = 0
EXTSCT = 0
PAR = EIS1:140000:000000
//
;          EIS2.CMD
; TASK BUILD SYMBOL TABLE
;
,EIS2,EIS/-HD = EIS2/MP
UNITS = 0
STACK = 0
EXTSCT = BP2OTS:12330
PAR = EIS2:140000:000000
//
```

The last files to be made are the ATPK indirect command files. Two files are needed for the EIS and FPU build procedures. A sample build file is shown below.

```
!          EISBLD.CMD
! BUILD EIS RESIDENT LIBRARY
!
$ALLOW NO ERRORS
RUN $MAC.TSK
EIS1 = EIS1
EIS2 = EIS2
↑C
RUN $TKB.TSK
@EIS1
↑C
RUN $TKB.TSK
@EIS2
↑C
RUN $MAKSIL
EIS
EIS .TSK
YES
EIS1.STB
EIS .LIB
↑C
$ALLOW FATAL ERRORS
```

Once the files are done, the ATPK program is run to build the EIS and FPU resident libraries and symbol tables. Both resident libraries should then be tested. This test requires that FPU hardware be present on the test machine. The test is run as follows.

Write a test program and build against the EIS reslib

Install EIS reslib and run test program

Install FPU reslib and run test program

Build the test program against the FPU reslib

Install FPU reslib and run test program

Install EIS reslib and run test program

Once the test procedure is complete, the resident libraries may be installed on their respective machines.

Appendix A

EIS/FPU Math Modules

$BFPER	Floating point error and setup
$DADD	Double floating add and subtract (EIS only)
$DATAN	Double precision arctangent
$DCON1	Convert double to integer
$DDIV	Double floating divide
$DEXP	Double precision exponent
$DFIX	Double to double truncation
$DING	Special integerize code (EIS only)
$DINT	Integer part of double
$DLOG	Double precision log
$DMUL	Double floating multiplies
$DNEXT	Double NEXT threads (EIS only)
$DSCAL	Scale factor preprocessor
$DSIN	Double precision SIN and COS
$DSQRT	Double precision square root
$DXDI	Double ** integer
$FADD	Floating adds
$FADDA	Real add via address
$FADDM	Real add to memory
$FADDP	Real add to pointer
$FATAN	Arctangent
$FCON1	Convert float to integer
$FDIV	Floating divides
$FEXP	Real exponent routine
$FFIX	Real to real truncation
FINT	Integer part of real
$FLOG	Single precision log
$FMUL	Floating multiplies
$FNEXT	Floating NEXT threads
$FSIN	Floating SIN
$FSQRT	Floating square-root
$FXFI	Floating ** integer
$JCON1	Convert integer to float
$MATII	Matrix conversion — integer

ACKNOWLEDGEMENTS

This algorithm was implemented with the aid of information supplied by the Basic-Plus-2 Development Group of Digital Equipment Corporation on the BP2 OTS library.

NEW PRODUCT. See our ad on page 45.

QUE.11 User's Guide

Version 1.0
February 1980

INTRODUCTION

QUE.11 is a package of programs which allow jobs to be run under the RSTS/E operating system on PDP-11 computers in batch mode; that is without direct control from the user.

A request to start a job is sent to QUE.11 which then takes over the control of the job until it finishes. This method of operation is most useful when all the responses to the controlled job are known in advance and can be written into a control file. This file is then used by QUE.11 to provide responses whenever the job requests inputs.

QUE.11 provides three types of service, all based on the principle just outlined. These services are:

1. Batch Mode

Used to run jobs which are not interactive and so do not require the use of a terminal. Another reason for batch mode is that the job requires some facility which is not immediately available; in this case the job is entered into the queue with a request for the required resource. The job will run as soon as the resource becomes free. Yet another reason for queuing the job is that it needs attention from the operator; perhaps it requires a special type of paper in the printer or perhaps it slows down the computer and so must only be run at night. In all these cases the requirements can be indicated to QUE.11 leaving the originator of the job free of the task of scheduling.

2. Print Spooling

A special case of the batch mode described above is printing files on the printer. The printer is one of the devices that is most frequently required by several people at once and it is often a cause for delay. QUE.11 provides a spooling service that allows requests for printing to be queued. The file(s) are printed as soon as the desired printer is free and the operator has loaded the required paper. The programs which transfer the files to the printer run under QUE.11 control like a standard batch job.

3. Immediate Mode

A user working at a terminal often finds that the same sequence of commands must be typed many times on the terminal. This is especially true during program development. Immediate mode allows the commands to be written once in a control file. QUE.11 will type each line from this file on your terminal as if you were typing them yourself.

Parts of the control file may be changed easily by means of parameters which alter some of the words in the file.

EXAMPLES

Several examples of how QUE.11 is used are shown here to illustrate some of the points raised above. You should try these on your system before reading the detailed descriptions which follow the examples.

Examples of QUE.11 in operation

1) Printing a file

```
Ready
Print XYZ.LST/delete ON normal
Ready
Job 31 sent to QUE.11
```

[This command prints the file XYZ.LST on normal paper and then deletes the file. The job will run as soon as the printer is free and the correct paper is loaded]

2) Printing on a non-standard printer

```
Ready
Print XYZ.LST ON Kb7:Form9
Ready
Job 32 sent to QUE.11
```

[The standard spool controller would print the file on Kb7: if this was allowed by the System Manager]

3) Standard Batch

```
Ready
Submit TABRUN
Ready
Job 34 sent to QUE.11
```

[TABRUN.CTL is a batch control file]

4) Batch with text substitution

```
Ready

Submit MTSAVE/file=list.dat for MT:T1249
Ready
Job 36 sent to QUE.11
```

[MTSAVE.CTL is a skeleton control file where the word FILE is replaced by LIST.DAT. The job will run when the operator loads a tape drive (MT:) with tape no. T1249]

5) Immediate Mode Execution

```
Ready
Do Compile/src:Prog 1
Keyboard controlled by QUE.11
old prog1
Ready
Compile
Ready
TKB prog1=prog1,LB:bp2com/lb
Ready
End of control by QUE.11
```

[In this example COMPILE.CTL is a control file with these lines:

```
old SRC
compile
TKB SRC=SRC,LB:bp2com/lb
```

The word SRC is replaced by PROG1 and then each line is typed on the user's terminal.]

BATCH MODE

Jobs run in batch mode are controlled by means of a 'pseudo-keyboard' which is a 'hidden' terminal that looks exactly like a real one to the controlled job. However, it is also connected to QUE.11 which can feed text to, and receive output from it. These pseudo-keyboards have terminal numbers which are lower than the numbers assigned to the real devices.

QUE.11 operates by reading lines from a control file which it 'types' on the pseudo-keyboard whenever the job requires input. The control file does not need HELLO or LOG commands at the begining or a BYE command at the end; these are automatically generated by QUE.11.

As an example, a control file might contain the following lines:

```
RUN MULTPY
3,5
```

The program MULTPY takes two numbers and prints their product. Suppose the control file is called TEST.CTL. You would request QUE.11 to run this file by typing:

```
SUBMIT TEST
```

The index of the job in the queue will be printed the terminal is now free for other work or you could log-off.

When the job is finished, which can be discovered by means of the SHOW command you would look at the Log File to see what has happened. This file has the same name as your control file but the extension name is '.LOG'. It might look like this:

```
LOG 5/6
Password:
RUN MULTPY
Input the first number? 3
          and the second? 5
The result of 3 x 5 is 15
Ready
End of control by QUE.11
```

(The program MULTPY might look like this, in Basic:

```
10    Print "Input the first number";
20    Input a
30    Print "          and the second";
40    Input b
50    Print
60    Print "The result of" ;a;"x";b;"is";a*b      )
```

The log file TEST.LOG shows everything that was typed on the hidden pseudo-keyboard and so it may be used to contain results in the same way that a real terminal might be used. This is often useful as a way to obtain a hardcopy of a run if you do not have a printing terminal.

The control file in the example is inflexible; it can only be used for the case of 3 and 5. However, there is a substitution facility in QUE.11 that allows general variables to be used in place of particular values. Suppose the control file is changed to look like this:

```
Run MULTPY
x,y
```

Now the values may be given when the batch request is entered:

```
SUBMIT test/x=3,y=5
Job 9 entered into the queue
Ready
```

RESERVED DEVICES

One of the reasons for using batch is to queue a job until one or more peripheral devices become free. This type of request is indicated by giving a 'FOR' or 'TO' clause in the SUBMIT command, like this:

```
SUBMIT test FOR mt:tape79
```

This command prevents the job starting until the operator loads a tape called TAPE79 on one of the magnetic tape drives.

This method of reserving devices may also be used to put a job into a special category in the queue. For example, your system might have a special 'dummy device' called CLASS: which would be used to schedule large jobs to run at night. You might enter a request like this:

```
SUBMIT test TO class:night
```

Ask the System manager which, if any, dummy devices are installed.

CONTROL FILES

Any commands that may be typed directly on a terminal may also be put into a control file. However there are some points to note:

1. Any line starting with $! will be treated as a comment line; it appears in the log file but has no other effect.

2. $-mark at the begining of any line is removed before the line is passed to the pseudo-keyboard.

3. Any line starting with $JOB or $EOJ is ignored completely; this allows some control files for other batch processors to be read.

4. ↑Z and ↑C (typed with up-arrow as shown) will be converted to Ctrl-Z and Ctrl-C at run time. (Note that $EOD is not supported.)

5. Any line starting with $ERR is ignored unless an error is detected. Errors are signaled if the controlled job prints a line starting with a ?-mark on the pesudo-keyboard. When this happens QUE.11 reads foward in the control file until the first line beyond $ERR or until the end-of-file.

This error control cannot be turned off and any RSTS/E error which occurs during the run will cause the job to end. No message is printed in the log file.

6. A line starting with $RECEIVE switches the status of the job into receive mode so that it may receive messages from OP commands.

7. Any occurrence of an item shown on the left of this table will be replaced by the text indicated at the right:

&PPN	[proj,prog] number of requester.
&TIME	current time
&DATE	current date
&DV	first reserved device
&VOL	volume on this device

BATCH COMMANDS

SUBMIT

```
SUBMIT | file    |[/parameters][ | TO   | dev:[volume]
       | ?       |               | FOR  |
       | /HElp   |               | WITH |
```

Notes:

1. ? or /HElp instead of the file name will produce a help message.

2. /parameters is a list of words in the control file which are to be replaced by the text also given. Each substitution is given like this:

word = replacement
or
word:replacement

Each pair is separated by '/' or ',' and the replacement text may be quoted if it contains either of these symbols or spaces. An example will illustrate how to build up a parameter list:

```
SUBMIT copy/input:"[1,4]test.dat"/output=MT:DATA80 for MT:DATA
```

3. TO, FOR, WITH have identical meanings in the Submit command.

4. Up to 5 devices and associated volumes may be specified in the FOR clause.

CANCEL

If you want to stop a queued Batch job use the CANCEL command with the index number of the job you wish to end (use SHOW to find the number).

Syntax:
CANCEL n

Note that this command only allows you to end jobs which are owned by your project, programmer number; if you try to end someone else's job an error message will appear on your terminal:

?Protection Violation

This restriction does not apply to privileged users.

SHOW

This command is used to get information about the jobs in the queue and about the devices which have been loaded by the operator with particular volumes. In general it returns one line of information about the group of jobs specified in the command but it can also return additional information about one selected job

Syntax:

```
SHOW | dev:[ppn][volume] |
     | ppn               |
     | n                 |
     | D[evices]         |
     | V[olumes]         |
     | O[wn]             |
     | S[elf]            |
```

These modes of use are explained below.

1. SHOW
   On its own SHOW gives a list of all the jobs in the queue.
2. SHOW dev:[ppn][volume]
   SHOW LP:[9,32]WIDE
   shows all jobs belonging to [9,32] which are queued for the printer with wide paper. Items in [] may be omitted.
3. SHOW ppn
   SHOW [9,32]
   shows al! jobs belonging to [9,32]
4. SHOW n
   SHOW 3
   shows additional information about job 3
5. SHOW D[evices]
   SHOW V[olumes]
   gives a list of all the devices which the operator has loaded.
6. SHOW O[wn]
   SHOW S[elf]
   show jobs belonging to this user.

PRINT SPOOLING

Because there are many users of a computer system it often happens that the printer is in use just when you want to list a file. It may also happen that you want to print on a special design of paper which must be loaded by the operator. QUE.11 provides a 'spool' service to make the use of the printer easier by puting all requests for printing into a queue which is cleared when the printer is free or when the operator loads the required forms.

The request for printing may also specify that several copies of a file are to be printed or that the file should be deleted after it has been printed.

Files produced by FORTRAN programs can be handled so that they appear correctly.

If required, the perforation between pages may be skipped and a header printed at the top of every page (this is very useful for program listings).

SPOOL COMMANDS

PRINT

This command places your request in the printer queue.

Syntax

```
PRINT | file list | [ ON [device:]paper[/SKIp] ]
      | /HElp     |
      | ?         |
```

file list expands to:

```
file[/DElete][/COpies:n][TYPe:FTN][,file. . . . .]
```

up to 50 characters may appear in the file list.

Notes:

1. The default extension for files is '.LST'

2. The ?-mark or /HE will cause a help message to appear

3. If the ON clause is omitted it is assumed to be:
   ON LP:NORMAL/SKIP

Thus:

```
Print XYZ/delete
```

is the same as

```
Print XYZ/delete on LP:normal/skip
```

4. The device and paper are always check to ensure that the System Manager allows their use. Otherwise the message

```
?Paper or Device not available
```

will appear.

5. /SKIP prevents printing accross the perforation at the end of each page. 6 lines are skipped and a one line page header is printed giving the file name, date, time and page number.

6. Wildcards may be used in the file name; the rules are those given in the description of the Directory Lookup SYS call.

CANCEL

This command will remove an item from the queue. If the job has actually started it will be stopped with this mesage on the last line:

```
*-*-*-* Job stopped without finishing
```

Syntax:

```
CANCEL n
```

where 'n' is the queue index number of the job. If you try to cancel a job that does not belong to you this message will be printed:

```
? Protection violation
```

SHOW

The SHOW command gives a list of the jobs in the queue. For full details see the description under Batch. Jobs waiting for a particular printer can be listed like this:

```
SHOW LP0:
```

assuming LP0: was the device in question. An account number or a type of paper may also be given.

If you use SHOW with the index number of a particular job you can obtain additional information for the job. The line giving the parameters has between one and three items in it. These are:

1. #FS:'list of files'
   This is the list of files to be printed, with switches, if any.

2. #SK:Y
   This appears if the /SKIP option is used.

3. #RQ:/name/page/block/char/copy/
   This appears if the job was REQUED by the operator

IMMEDIATE MODE

The DO command is a useful way to store sequences of commonly required commands so that they can be executed on a terminal by typing one line.

Syntax:

```
DO | FILE   |[|/|parameters]
   | /HElp  | |;|
   | ?      |
```

Examples:

```
DO compile/name=program
DO build/RTS:bp2com
DO list
```

Notes:

1. The default extension for the filename is '.CTL'

2. If parameters are given they cause some words in the control file to be replaced by the given text. The format of a parameter list is:

```
/old=new,old2=new2,old3=new3
```

The commas(,) may be replaced by obliques(/). The new part may be enclosed in quote marks (' ' or " ") if it contains spaces, commas or obliques (, or /).

3. Cntrl-C and Cntrl-Z may be inserted into the file by typing C or Z (uparrow C, uparrow Z).

4. Comments may be given by starting a line with $!

5. All the rules which apply to batch control files also apply here; in particular the reserved words (&PPN etc.) are also reserved for immediate mode files even when they are not relevant.

6. Batch files can often be tested in immediate mode so that step by step operation may be observed.

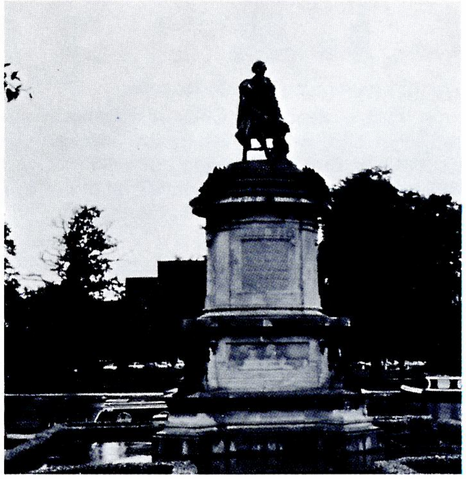
7. Do not use parameters which can be confused with RSTS/E file switches, e.g.

```
DO COMPILE/FILE=XYZ
```

This should be given in the alternative form:

```
DO COMPILE;FILE=XYZ
```

8. Control-C may be used at any time to cancel a DO command.



Who and where is this?
A prize if you tell us exactly!

CLASSIFIED

Send Classified Ads to: RSTS Classified, P.O. Box 361, Fort Washington, PA 19034. $1.00 per word, first 12 words free with one year's subscription.)

MACRO SYSTAT 7.0 LARGE FIP
Super Fast — almost no CPU, all standard features + core & open file by job. Works with RSX or BP2 RTS's. Nine track 800 or 1600 only. Send $\$40^{00}$ cash, check or m/o. Bob Meyer, 9 Lockwood Ave., Fieldsboro, NJ 08505.

HARDWARE & SOFTWARE DESIGN
Mini — Micro
RSTS, A Specialty
Scott Banks, Systems Design, 1108 Morefield Rd., Phila., PA 19115. 215—677-4836.

WANTED FOR CASH: 11/70 w or w/o FPP minimum DEC memory, no peripherals, high-boy cabinets. DEC Service Letter, Dave Mallery 215—364-2800.

RSTS INTERNALS, TUNING, RT11/RSX MACRO under RSTS consultation & programming — call MACRO MAN, Bob Meyer, 609—298-9127.

MACRO PROGRAMS; Subroutines for B+2 DIBOL, Mark Diaz, 28W 59th, Apt.E, Westmont, ILL 60559.

TESTED DATA SYSTEMS. Application support for Basic Plus Two Users. 1718 College Ave., Regina, Sask, 306—522-5280.

FOR RSTS/E SOFTWARE consulting in New England, Call (617) 895-5090.

SYZYGY ANNOUNCES RS232 Test Set $75, 252 San Lorenzo, Pomona, CA 91766.

MASS11 Call 312—843-DATA.

NEW MINI/MICRO SOFTWARE Sourcebook. I.S.I., 1807 Glenview, Glenview, IL 60025 (312)724-9280.

MAJOR RSTS/E SITE seeks communication with same for performance measurement (403) 266-9927.

FORMAT, PATTERN-CHECK RK05's during timesharing. Program $200 plus media. Structured Software Systems, PO Box 7893, Ann Arbor, MI 48107, (313)665-7377.

HIGH SCHOOL Students interested in RSTS computing, consider Proctor Academy, Andover, NJ 03216.

GEORGE W. HALLAHAN and Company Business Data Processing/Software Consultants, PO Box 43, Newton, MA 02158.

BASIC+ MEDICAL GROUP A/R package with four years online operation (309) 757-9207.

RSTS BASIC+ PROGRAMMER wants to freelance. Educational and manufacturing background. Call Dave (617) 327-9476.

AUTOMATE PACKAGING Specifications on RSTS. Call Mike Moore (214) 980-4766.

SYSTEM DESIGN ASSOC. KNOWS who John Galt is, and always has.

OPTIMA CABINETS with AC Distribution and Fans. Call Mr. Connell, 505/294-5051.

TESTED DATA SYSTEMS. Application support for Basic Plus Two Users. 1718 College Ave., Regina, Sask. 306-522-5280.

BOSTON AREA: Problems with RSTS? Call Considine Computing Services, 617-484-6862.

DATA PROCESSING
MINICOMPUTER
PROFESSIONALS

Leading software company is expanding R&D staff. Seeking qualified, self-motivated, ambitious CAREER professionals. 1 to 2 yrs. experience. Commercial systems application, RSTS/E experience a plus. Excellent opportunity with state-of-the-art, progressive and growth oriented company. Excellent benefits, salary commensurate with experience. If you are interested in a CAREER opportunity, please send resume, references and salary history to:

V. P. Technical Operations
AMCOR Computer Corporation
1900 Plantside Drive
Louisville, KY 40299
502/491-9820

NICHOLSON PARTNERS HOSPITAL Consultants, 328 Pacific Highway, LINDFIELD 2070, AUSTRALIA, Phone 02 46 1113.

$200 RSTS/E* WORD PROCESSOR
CBEDIT.BAS

Single Basic-Plus* program with CRT input, window edit and file save. Add, locate, change, replace, delete, block move & copy, standard paragraph append, etc. VT* series and Hazeltine terminal drivers. Others easy to add.

Fully formatted output (margins, justify, center, underscore, headers, page numbers etc.) to terminal, disk or line printer. Bi-directional driver for Diablo. We use it daily for all our secretarial work.

9-Track $200 RK05 $260 ppd
T. F. Hudgins & Assoc., Inc.
P.O. Box 10946, Houston TX 77018
Woods Martin — 713/682-3651
*TM Digital Equipment Corporation

SYSTEM DESIGN and Support Available-RSTS- Write Nemtine, Box 767, Mar. MA 01945.

BUY, SELL, SWAP DEC Equipment. Computel Systems Ltd. (212) 351-5395.

HUMANS, KLINGONS, ROMULANS: Announcing SPB! The Ultimate, Frank Farmer. (313) 286-8800.

STANDARD MEMORIES OFFERS add-on core and add-in MOS memories for 11/70 computers, ECOM 70/127, $15K per megabyte. Donna 800-854-3792.

RSTS-11 does not TOPS-10 at Eastern Michigan University!

DESIRE CONTACT USERS involved Linguistic Analysis. Smith Linguistics, MUN, St. Johns, Newfoundland, Canada. 709-737-8131.

RSTS DIBOL PROGRAMMER WANTED. Live and work on Cape Cod, Mass. 617-888-5200.

DATA COMMUNICATIONS Consulting remote/on-site support. RT-11 also. 803-798-8043.

CONSULTING DEC Mini-Micro specialists serving Western Colorado, OWL Associates. 303-245-2303.

PAYROLL RSTS/E VERY FLEXIBLE. Avoid year-end rush. Send for info now. C. Wilson, Box 12152, Lexington, KY 40581.

SAVE MONEY and TIME. Order your 11/70, 11/34 or peripherals from Nordata. Immediate Delivery, call (206) 282-1170.

RANDALL SYSTEMS USERS in New England call 617-444-8920 for reciprocity.

11/34 RSX 11-M. Used tape drive for Unibus 11, Jerry Boris 382-5587.

CR11BC FOR SALE plus controller. Never used, $7000, 216-929-5697.

HELP START Albany N.Y. RSTS LUG. Call Joyce Karl (518) 489-2538.

LOOKING FOR BUSINESS Application Software?? We have the Best! IAS COMPUTER Corp., Ltd. 902-453-4620.

FOR SALE: RWM03, RM03, TME-11, RK611, RK06. (204) 477-1870.

AMERICAN BUSINESS COMPUTERS, Inc. specializes in RSTS turnkey systems. Call: 713-265-2573.

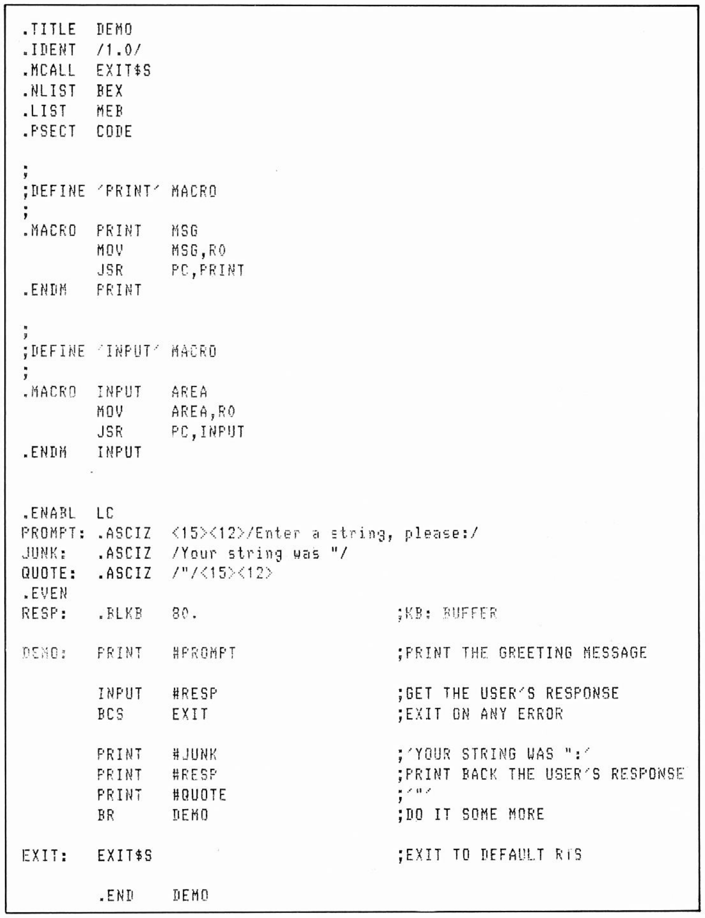
LOWRY and ASSOCIATES - Engineering Software - Time and Frequency Domain analysis, Mathematical modeling, Analog systems and circuits. Irvine - (714) 751-3820.

Billing Package for RSTS/E Session Accounting. Write: Mark/Ops, 636 Beacon, Boston, MA 02215.

RSTS/E and BASIC-PLUS Programming Training Courses. Conducted at your site. EDGE DATA CORPORATION, 100 Franklin St., Boston, MA 02110 (617) 451-1919.

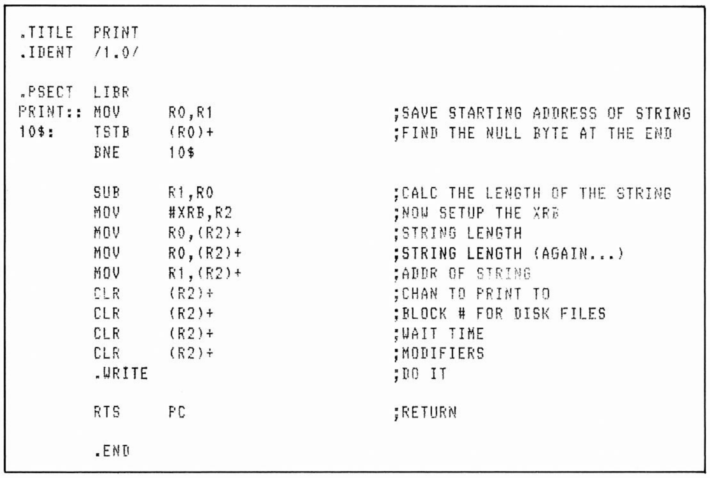
WELCOME AGAIN TO MACRO LAND . . . continued from page 79

FIGURE 1.



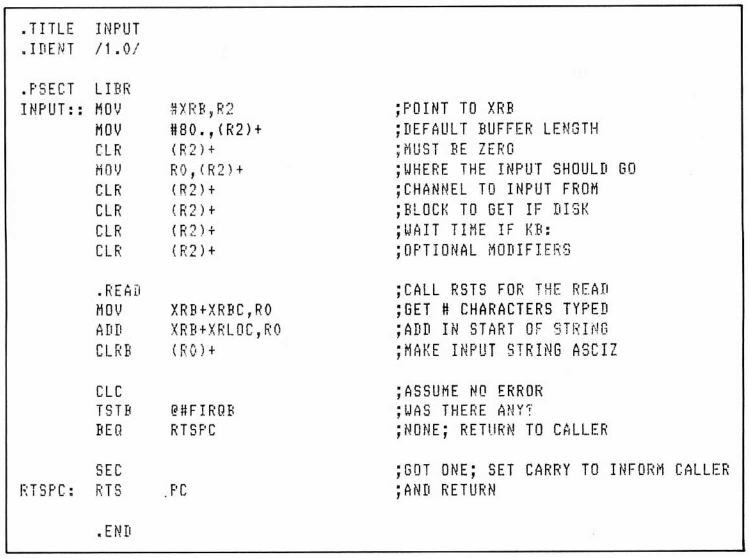
```
.TITLE  DEMO
.IDENT  /1.0/
.MCALL  EXIT$S
.NLIST  BEX
.LIST   MEB
.PSECT  CODE

;
;DEFINE 'PRINT' MACRO
;
.MACRO  PRINT   MSG
        MOV     MSG,R0
        JSR     PC,PRINT
.ENDM   PRINT

;
;DEFINE 'INPUT' MACRO
;
.MACRO  INPUT   AREA
        MOV     AREA,R0
        JSR     PC,INPUT
.ENDM   INPUT


.ENABL  LC
PROMPT: .ASCIZ  <15><12>/Enter a string, please:/
JUNK:   .ASCIZ  /Your string was "/
QUOTE:  .ASCIZ  /"/<15><12>
.EVEN
RESP:   .BLKB   80.                     ;KB: BUFFER

DEMO:   PRINT   #PROMPT                 ;PRINT THE GREETING MESSAGE

        INPUT   #RESP                   ;GET THE USER'S RESPONSE
        BCS     EXIT                    ;EXIT ON ANY ERROR

        PRINT   #JUNK                   ;'YOUR STRING WAS ":'
        PRINT   #RESP                   ;PRINT BACK THE USER'S RESPONSE
        PRINT   #QUOTE                  ;'"'
        BR      DEMO                    ;DO IT SOME MORE

EXIT:   EXIT$S                          ;EXIT TO DEFAULT RTS

        .END    DEMO
```

FIGURE 2.



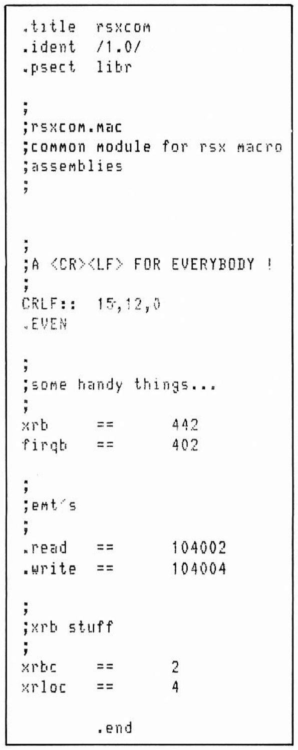
```
.TITLE  PRINT
.IDENT  /1.0/

.PSECT  LIBR
PRINT:: MOV     R0,R1                   ;SAVE STARTING ADDRESS OF STRING
10$:    TSTB    (R0)+                   ;FIND THE NULL BYTE AT THE END
        BNE     10$

        SUB     R1,R0                   ;CALC THE LENGTH OF THE STRING
        MOV     #XRB,R2                 ;NOW SETUP THE XRB
        MOV     R0,(R2)+                ;STRING LENGTH
        MOV     R0,(R2)+                ;STRING LENGTH (AGAIN...)
        MOV     R1,(R2)+                ;ADDR OF STRING
        CLR     (R2)+                   ;CHAN TO PRINT TO
        CLR     (R2)+                   ;BLOCK # FOR DISK FILES
        CLR     (R2)+                   ;WAIT TIME
        CLR     (R2)+                   ;MODIFIERS
        .WRITE                          ;DO IT

        RTS     PC                      ;RETURN

        .END
```

FIGURE 3.



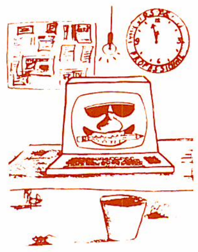
```
.TITLE  INPUT
.IDENT  /1.0/

.PSECT  LIBR
INPUT:: MOV     #XRB,R2                 ;POINT TO XRB
        MOV     #80.,(R2)+              ;DEFAULT BUFFER LENGTH
        CLR     (R2)+                   ;MUST BE ZERO
        MOV     R0,(R2)+                ;WHERE THE INPUT SHOULD GO
        CLR     (R2)+                   ;CHANNEL TO INPUT FROM
        CLR     (R2)+                   ;BLOCK TO GET IF DISK
        CLR     (R2)+                   ;WAIT TIME IF KB:
        CLR     (R2)+                   ;OPTIONAL MODIFIERS

        .READ                           ;CALL RSTS FOR THE READ
        MOV     XRB+XRBC,R0             ;GET # CHARACTERS TYPED
        ADD     XRB+XRLOC,R0            ;ADD IN START OF STRING
        CLRB    (R0)+                   ;MAKE INPUT STRING ASCIZ

        CLC                             ;ASSUME NO ERROR
        TSTB    @#FIRQB                 ;WAS THERE ANY?
        BEQ     RTSPC                   ;NONE; RETURN TO CALLER

        SEC                             ;GOT ONE; SET CARRY TO INFORM CALLER
RTSPC:  RTS     PC                      ;AND RETURN

        .END
```

FIGURE 4.



```
.title  rsxcom
.ident  /1.0/
.psect  libr

;
;rsxcom.mac
;common module for rsx macro
;assemblies
;


;
;A <CR><LF> FOR EVERYBODY !
;
CRLF::  15,12,0
.EVEN

;
;some handy things...
;
xrb     ==      442
firqb   ==      402

;
;emt's
;
.read   ==      104002
.write  ==      104004

;
;xrb stuff
;
xrbc    ==      2
xrloc   ==      4

        .end
```

NEWS RELEASES



Occasionally we are requested to print news that may be of interest to the RSTS community. We are happy to offer this feature to our readers. We reserve the right to print only as time and space permit. We cannot return photos or manuscripts. Send news releases to: *RSTS NEWS RELEASE, P.O. Box 361, Fort Washington, PA 19034.*

September 8, 1980
SSI NAMES DISTRIBUTORS . . .

Fort Lauderdale, Florida — Southern Systems, Inc. (SSI), printer system manufacturer, has named six distributors for a nationwide system handling sales of the SSI M-200, a 200 line per minute impact matrix printer system.

The M-200 line printer offers the computer industry the first 200 lpm printer selling in the price range of 180 cps* devices. SSI's M-200 offers standard features that include 128 characters, 132 columns, internal self-test, form length select switch, standard 10 cpi*, condensed 16.7 cpi*, and elongated character spacing, as well as 6/8 lpi*, diagnostic display and clean cassette ribbon. SSI provides this product with all necessary interfacing (parallel or serial) and has a nationwide service organization.

The six firms and their sales areas are Wild & Rutkowski, Inc. of Jericho, NY, handling New York state; Digital Computer Products of San Mateo, CA, handling northern California, Oregon, Washington and northern Nevada.

Also, Rohr Associates of Philadelphia, handling Delaware, southern New Jersey and eastern Pennsylvania; Barry Sales, Inc. of Richardson, TX, handling Louisiana, Texas, Oklahoma and Arkansas; J.J. Wild Inc. of Needham, MA, handling Maine, Vermont, Massachusetts, New Hampshire, Rhode Island and Connecticut; and Deerfield Systems Inc. of Deerfield, IL, handling Illinois, Wisconsin, Minnesota, Iowa, Nebraska, Kansas, Missouri, and northwestern Indiana.

Southern Systems creates a wide range of line printer systems, from the low end speed of the M-200 up to a 1500 line per minute system. The company specializes in interface technology, allowing both parallel and serial functioning of printers with the majority of computers, including mainframes.

SSI is located at 2841 Cypress Creek Road, Fort Lauderdale, FL 33309; (305) 979-1000; (800) 327-5602.

*For non-trade business editors' information:
*cps - Characters per second; *cpi - Characters per inch *lpi - Lines per inch.

— — — — — — —

September 15, 1980
SSI INTRODUCES 900 LPM BAND PRINTER . . .

Fort Lauderdale, Florida — A 900 line per minute band printer system, the B-900, is the newest addition to the Southern Systems Inc. band printer series.

The B-900 joins the SSI 300 and 600 lpm band printer systems, which have been field-proven over the past two years by SSI, one of the first to apply the highly efficient band technology for impact printing.

Designed for operator efficiency, easy maintenance and reliability, the B-900 features several bands — a 48, 64 and a 96 character set, as well as many specialized and foreign character sets. Bands are changed as easily as typewritter ribbons, permitting versatility and economy in meeting a variety of output needs.

The B-900 print speed is 1,100 lpm at 48 characters, 900 lpm at 64 characters, and 600 lpm at 96 characters.

Switch-selectable features include the vertical spacing — for 6 or 8 lines per inch — and for multiple form lengths. The B-900, like the 300 and 600, provides up to five clear copies.

Other features include floor standing quietized cabinet, paper puller, a built-in diagnostic display, large viewing window so that output is easily monitored, paper-out detect sensors and print-to-bottom-of-form capabilities. Both the paper and the clean cassette ribbons are easily loaded from the front.

In addition to the diagnostic display which monitors printing status at all times, the B-900 features a built-in self-test feature which allows the printer to be run independently of the computer for performance monitoring.

The B-900 is compatible with DEC, including the DECsystem 10 and 20; Hewlett Packard, Data General, SEL, TI, Burroughs and most other mini-computers and mainframes.

Parallel interfacing is standard, however, SSI also supplies serial interfacing, both synchronous and asynchronous, for operation of printers remote from computers.

Southern Systems is located at 2841 Cypress Creek Road, Fort Lauderdale, Florida 33309; telephone (305) 979-1000, (800) 327-5602.

— — — — — — —

October 3, 1980
ROBERT LESSER TO HEAD SSI SALES IN U.S. . .

Fort Lauderdale, Florida — Robert E. Lesser has been named national sales manager for Southern Systems, Inc., (SSI), Fort Lauderdale-based computer printer company.

Lesser was formerly New England district sales manager for Dataproducts Inc. During his five years with that firm, he was heavily involved in introducing new printers, including those based on band, matrix, daisy wheel and thermal technology. That involvement included coordination with technical staffs to meet OEM customer needs in printer capabilities.

SSI services end-users with a wide range of line printer systems, including low-end speed of the 200 lpm matrix, the medium speeds of the new band printer systems, plus drum and charaband printers up to 1,500 line per minute systems. The company specializes in interface technology, allowing both parallel and serial functioning of printers with most computers, including mainframes.

During five years with Centronics, Lesser worked with OEMs and end-users of printers as well as with the company's manufacturers reps. While there he also specialized in management of product design improvements and customization.

"Mr. Lesser's background and total experience with add-on technologies in computer printing and in sales is a tremendous addition to our staff and customers. Mr. Lesser will direct all sales for SSI within the U.S.," said James Rule, SSI vice president of marketing.

SSI is located at 2841 Cypress Creek Road, Fort Lauderdale, FL 33309.

— — — — — — —

October 6, 1980
NEW PROGRAM ENHANCEMENTS TO RABBIT-1 RESOURCE ACCOUNTING FOR VAX/VMS USERS. . .

West Palm Beach, Florida — RAXCO Inc. announces two new program modules for RABBIT-1, a Resource Accounting and Billing System for VAX/VMS and PDP 11/RSTS users.

The first new option identifies the communication port used by interactive users. This feature is important if different rates are to be charged for various communication facilities, such as Telenet or Datapac.

Line speed and port identification data is captured by RABBIT-1 on a port by port basis at login time. This information may be utilized for security control as well as accounting purposes. The current level of VMS (2.0) does not identify the port used nor line speed. RABBIT-1 port identification module provides capability to fill the void.

The second new RABBIT-1 option is a program usage accounting module designed to handle program utilization and surcharges for specific software products. It is also useful in determining the value and frequency of use of specified programs.

At the conclusion of program usage, a record is written to the accounting file. The record contains the program name, user name, and quantities of computer resources consummed. This data may be summarized for program usage totals.

Only specified programs will have usage data generated. The normal VAX/VMS accounting records will not be altered.

Both modules are available under RABBIT-1 for $250 each or may be rented for $12.50 monthly.

RAXCO develops and markets a full line of DEC software systems for system support, data management and financial planning. The entire RABBIT System is marketed and supported in the U.S.A., Canada, and the United Kingdom.

For more information, contact: RAXCO Inc., 3336 N. Flagler Drive, West Palm Beach, FL 33407. Telephone: (305) 842-2115.

— — — — — — —

October 7, 1980
SSI TO OPEN NEW PLANT
IN CLEARWATER, FL

Fort Lauderdale, Florida — Southern Systems Inc. (SSI), computer printer system company based in Fort Lauderdale, will open a new 10,000-square-foot manufacturing plant in Clearwater, FL on Oct. 15, according to Joseph Horn, SSI president.

The company's research/development and manufacturing will occupy the new structure at 2131 Sunnydale Blvd. in the Clearwater Industrial Park. Designed by Peter Marich Architect & Planners, the plant was built by Ferris Constructors, Inc. at a cost of $400,000.

SSI has doubled sales each year for the past four years and anticipates an increasingly larger share of the projected computer printer market, expected to reach $10 billion by 1988, said Horn. SSI's sales for 1981 are expected to reach $10 million plus. Employment in the new plant at opening will be 18, with a projected 25 employees by mid-1981.

The firm supplies end users and OEMS with impact printer systems ranging from 200 lpm to 1,500 lpm. These printer systems are configured and supplied with complete electronics and interfacing to allow operation with computers from DEC. Hewlett Packard, Data General, Honeywell, SEL and many others.

Corporate offices are located in Fort Lauderdale, FL at 2841 Cypress Creek Road.

— — — — — — —

November 4, 1980
RESOURCE-11 OFFERS NEW
DISK SAVE & COMPRESS PACKAGE . . .

Gaithersburg, MD — RESOURCE-11 is a systems integrator of Digital Equipment Corporation (DEC) computers. We provide a competent field service organization which can install and service DEC and DEC compatible components. RESOURCE-11 also provides numerous generalized business, word and data processing software products. The main interest of the company is to provide "total systems" for our customers. RESOURCE-11 provides all the necessary hardware and software support to enable a business to more successfully incorporate data processing into their business environment.

RESOURCE-11 can supply the computer needs of your growing business — for economy, quality, reliability, performance and dependable support services in a computer system. The "total system" includes market-tested hardware from DEC, reliable RESOURCE-11 software, training, and continued hardware and software support.

RESOURCE-11 only selects compatible, quality components, designed with advanced technology and engineered performance which are integrated with our software to provide an affordable computer system.

The major operating philosophy of

RESOURCE-11 is to utilize highly skilled technical personnel in the performance of all assignments. We are professional problem-solvers who bring a rare blend of hardware and software expertise to the design, development and implementation of computer related tasks.

RESOURCE-11 offers complete system support for OEMs and end users of Digital Equipment Corporation (DEC) systems. Professional evaluation of your needs before and after your investment. RESOURCE-11's technical expertise encompasses the entire PDP-11 and VAX-VMS product line.

RESOURCE-11 now offers a new package for disk optimization under RSTS/E called "DISK SAVE & COMPRESS (DSC). DSC permits pre-extension of RSTS/E UFDs, data compression, and manual placement of files for disk seek optimization on a total system scale. The above are just a few of the features currently available. New releases will incorporate RESOURCE-11 and user requested enhancements.

— — — — — — —

RABBIT-4 SECURITY SYSTEM
FOR DATA FILES
UNDER RSTS/E VERSION 7 . . .

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

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

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

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

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

RABBIT-4 is available through a permanent license or rental program from $2500 and $99/month respectively. It is the newest member of the RSTS RABBIT family which includes: RABBIT-1, Resource Billing; RABBIT-2, System Performance Analysis; RABBIT-3, Job Accounting and Monitoring.

RAXCO develops and markets a full line of DEC software systems for system support, data management and financial planning. The entire RABBIT System is marketed and supported in the U.S.A., Canada, and the United Kingdom.

For more information contact, RAXCO Inc., 3336 N. Flagler Drive, West Palm Beach, Fl. 33407, phone (305) 842-2115.

— — — — — — —

October 6, 1980
NEW PROGRAM ENHANCEMENTS TO
RABBIT-1 RESOURCE ACCOUNTING
FOR VAX/VMS USERS . . .

West Palm Beach, FL — RAXCO Inc. announces two new program modules for RABBIT-1, a Resource Accounting and Billing System for VAX/VMS and PDP 11/RSTS users.

The first new option identifies the communication port used by interactive users. This feature is important if different rates are to be charged for various communication facilities, such as Telenet or Datapac.

Line speed and port identification data is captured by RABBIT-1 on a port by port basis at login time. This information may be utilized for secuirty control as well as accounting purposes. The current level of VMS (2.0) does not identify the port used nor line speed. RABBIT-1 port identification module provides capability to fill the void.

The second new RABBIT-1 option is a program usage accounting module designed to handle program utilization and surcharges for specific software products. It is also useful in determining the value and frequency of use of specified programs.

At the conclusion of program usage, a record is written to the accounting file. The record contains the program name, user name, and quantities of computer resources consummed. This data may be summarized for program usage totals.

Only specified programs will have usage data generated. The normal VAX/VMS accounting records will not be altered.

Both modules are available under RABBIT-1 for $250 each or may be rented for $12.50 monthly.

RAXCO develops and markets a full line of DEC software systems for system support, data management and financial planning. The entire RABBIT System is marketed and supported in the U.S.A., Canada, and the United Kingdom.

For more information, contact: RAXCO Inc., 3336 N. Flagler Dr., West Palm Beach, FL 33407. Phone (305) 842-2115.

— — — — — — —

November 1980
VT100 CONVERSION KIT . . .

Sunnyvale, CA — Data Node, Inc. is pleased to announce the immediate availability of VT100 Conversion Kits. This kit allows any user of the very popular Digital Equipment Corporation VT100 video terminal to convert the terminal to a full microcomputer system with the installation of an INT/200 microcomputer board. Actual installation of the board can be accomplished by the user in an average of ten minutes using the step by step instructions supplied with the conversion kit.

The converted VT100 microcomputer is called a Micro Node and is the industry's first commercial grade microcomputer featuring built-in networking capability.

The VT100 Conversion Kit includes a 64 kilobyte Z80A microcomputer board with 32K bytes of ROM/PROM space, a detailed installation manual, a Node Basic license, and either CP/M or MP/M licences depending on the user's choice. An optional diskette/printer controller board allows the user to attach local single or dual-density 8 inch diskette drives, and a Centronics/Data Products/Serial Interfaced Printer.

Node Basic is a superset of Microsoft Basic with additional verbs to properly handle the screen, such as: blank, blink, reverse, position, accept input, prompt, make bold, display wide or tall characters, etc.

Node Basic also contains proper extensions for the network communications with central data management facilities on a host PDP-11 RSTS/E system. These extensions provide for the handling, in context, of Node Central data managed files on the PDP-11 in parallel with local files on an attached diskette drive.

Software for the conversion kit is furnished on diskette or magtape depending on the users choice for a stand-alone Micro Node with diskette drives, or a down-line loadable Micro Node connected to a PDP-11 RSTS/E System.

The Micro Node user should choose the local operating system best suited to the planned Micro Node usage. CP/M is the de facto standard microcomputer operating system and is used by most of the microcomputer industry. CP/M supports a single user environment. MP/M is the multi-tasking version of the CP/M operating system ad is required if the user wishes to take advantage of Data Node's unique background spooling capability for the Micro Node. This capability allows the Micro Node to continue data entry/inquiry with the Node Central while spooled output is simultaneously transmitted from the Node Central to the Micro Node's attached printer or diskette. MP/M is also required for those users wishing to utilize local Micro Node to Micro Node Networking, or desiring to support several dumb terminals off one Micro Node.

The VT100 Conversion Kit lists for $1850 includig Node Basic, while CP/M and MP/M are individually priced at $120 and $300 respectively. For Details write or call: Sales Manager, Data Node, Inc., 432 Toyama Drive, Sunnyvale, CA 94086, (408) 744-0561.

— — — — — — —

List of Advertisers

Able Computer Technology, Inc.	I.B. Cover
AMCOR	p. 2
American Used Computer	p. 41
Computer Methods Corp.	p. 9
Datafusion Corp.	p. 12
Datalease	pp. 49, 65, 71
Data Node, Inc.	p. 48
Data Processing Design, Inc.	B. Cover
Data Systems Services	p. 96
Digital Communications Associates, Inc.	p. 43
Distributed Logic Corp.	p. 57
Enterprise Technology Corp.	p. 77
Evans Griffiths & Hart, Inc.	pp. 55, 91
Finar	p. 33
Grossman Graphics	p. 91
Interactive Management Systems	p. 25
M Systems, Inc.	pp. 45, 76, 91
McHugh, Freeman & Associates, Inc.	p. 46
Monolithic Systems Corp.	p. 75
National Information Systems, Inc.	p. 60
Nationwide Data Dialog	p. 91
NCA Corporation	p. 73
Northwest Digital Sales	p. 21
Plycom Services, Inc.	p. 50
Raxco, Inc.	p. 39
Resource-11	p. 11
Ross Systems, Inc.	I.F. Cover
Software Results Corp.	pp. 13, 85
Software Techniques, Inc.	pp. 5, 59
Southern Systems, Inc.	p. 1
System Industries	p. 7
Telecom Computer Systems, Inc.	p. 87
Transnet Corp.	p. 30
URS	p. 50

The New Standard For PDP-11 Disk System Technology

Check with us before you buy, because we can beat all prices!

Know All the Hidden Costs!

Only System Designed Just for PDP-11 Family
Designed exclusively for DEC's UNIBUS or MASSBUS CPU's. On the UNIBUS, it's just one card that plugs into any spare SPC slot. On the MASSBUS, four cards plug into any spare existing RH70 standard back plane.

Same Disk Drive as DEC RM02-03, RP06 and RM04-05
We use the same disk manufacturers as does DEC. The RM02-03 is the 9762 CDC 80MB and the new RM04-05 300MB is the CDC 9766. Only the LOGO is different.

Transparent to All DEC Software, Diagnostics and Drivers
You bet! Use your existing Software . . . no change needed. Runs all DEC's Diagnostics plus has its own. Fully emulates DEC disk Drivers.

Worldwide Installation and Maintenance
Through Data Systems Services, maintenance and installation is provided via CDC for both Drive and Controller. We also offer full PDP-11 system support.

Full Media Compatible?
That's right! You can read or write on our drives. Put it on DEC's and it will play or vice versa. TRULY MEDIA COMPATIBLE.

OEM Discounts Available

DRM02-3 80MB Slave
DRJM02 80MB + Controller
DRWM03 80MB + Controller
DRJM04-5 300MB Slave
DRJM04 300MB + Controller
DRWM05 300MB + Controller
RJM07 600MB + Controller
RWM07 600MB + Controller
RM07 600MB Slave

DRP06 200MB Slave
DRWP06 200MB + Controller
DTU45
DTU77
DTU125
Line Printers
Memories
11/34 → 11/70, VAX
Systems

DATA SYSTEMS SERVICES

(714) 770-8024

23901 Remme Ridge
El Toro, CA 92630

PDP and DEC are registered trademarks of Digital Equipment Corporation.

An important tip for PDP-11 users

Play the blue-chip cards from Able.

COMMUNICATIONS PRODUCTS

DMAX/16™
(16-LINE DH11 REPLACEMENT)

INSTALLS IN: All PDP-11's; in less than one half the space of DH11. **DATA RATES**: All 14 standard baud rates plus 19.2K baud and one user programmable data rate (16 baud rates). **PROCESSING ADVANTAGES**: Word transfers (in lieu of byte DMA) cut processing overhead by half! **OPERATING MODES**: Full or half duplex with full modem control via DM/16 option. **CAPACITY**: Up to 256 lines on a single PDP-11.

QUADRASYNC/B™
(4-LINE DL-11 REPLACEMENT/EIA)

INSTALLS IN: All PDP-11's; 4-lines per SPC slot at one unit load to Unibus. **DATA RATES**: 7 independently selectable baud rates for each of 4 channels (150-9600). **ELECTRICAL**: EIA standard RS232C (Modem control not supported). **VECTOR/ADDRESS SELECTION**: Vector and address values to be set on boundaries of 00_8 or 40_8. 16 continuous word address for Vector or Address.

QUADRASYNC/C™
(4-LINE DL11 REPLACEMENT/CL)

INSTALLS IN: All PDP-11's; 4-lines per SPC slot at one unit load to Unibus. **DATA RATES**: 7 independently selectable baud rates for each of 4 channels (150-9600). **ELECTRICAL**: 20MA current loop (Send : Receive). **VECTOR/ADDRESS SELECTION**: Vector and address values to be set on boundaries of 00_8 or 40_8. 16 continuous word address for Vector or Address.

QUADRASYNC/E™
(4-LINE DL11-E REPLACEMENT)

INSTALLS IN: All PDP-11's; 4-lines per SPC slot at one unit load to Unibus. **DATA RATES:** 7 independently selectable baud rates for each of 4 channels (150-9600). **ELECTRICAL:** EIA standard RS232C – with modem control. **VECTOR/ADDRESS SELECTION:** 16 continuous word address for Vector or Address – starting values selected on any boundary.

QUADRACALL™
(4-LINE DN11 REPLACEMENT)

INSTALLS IN: All PDP-11's; 4-lines per SPC slot at one unit load to Unibus. **PERFORMANCE:** Interfaces up to 4 Bell 801 ACU's with Unibus enabling any PDP-11 to dial any DDD network number to establish data link. **INPUT/OUTPUT:** 5-input signals from ACU are handled by EIA RS232 receivers. 6-output signals are transmitted using EIA RS232 drivers. **VECTOR/ADDRESS SELECTION:** Allows selection of device address and vector by use of pencil switches.

DV/16
(16-LINE DV11 REPLACEMENT)

INSTALLS IN: All PDP-11's; in less than one half the space of DV11. **DATA RATES:** 16-line throughput of up to 30,000 char/sec (19.2K baud full duplex for each line) total. **PROCESSING ADVANTAGE:** Word transfers (in lieu of byte DMA) permit user to operate within one half the DV11 bandwidth for data transfers. **OPERATING ADVANTAGE:** User may mix sync and async lines in combinations of 4 or 8 lines with modem control and full system software compatibility with all DV11 performance features.

DZ/16
(16-LINE DZ11-E REPLACEMENT)

INSTALLS IN: All PDP-11's in any standard hex-width SPC slot; takes half the space at half the bus loading imposed by the DZ11-E. **DATA RATES:** All 15 standard DZ11 baud rates (50-9600). **IMPLEMENTATION ADVANTAGES:** On-board pencil switches allow address and vector selection flexibility without the need for jumpers. Data format is program-selectable for each channel.

MEMORY PRODUCTS

SCAT/45™
(ADD-IN FASTBUS MEMORY)

INSTALLS IN: PDP-11/45, -11/50 and -11/55. **EXPANDS IN:** 32K word increments/board. One-half of the available Fastbus space will accept full 124K word complement. **ADDRESSES ON:** Any 4096 word boundary across entire 124K word range. User has full memory complement at 330 nsec cycle-time memory instead of 32K word limitation imposed by the computer manufacturer.

CACHE/45™
(CACHE BUFFER MEMORY)

INSTALLS IN: PDP-11/45, -11/50 and -11/55. **CAPACITY:** 2048 byte (1K word). **ENHANCEMENT FACTOR:** Run time reductions to 50% (100% speed improvement) are achievable. **CACHE PARITY:** Automatically goes off-line in event of any data error. **RANGE SELECTION:** User may optimize hit ratio by upper/lower limit switch settings. **SPECIAL FEATURE:** Cache/45 can be enabled via software or console switches.

CACHE/434™
(4K WORD CACHE MEMORY)

INSTALLS IN: PDP-11/34 and -11/34A **without** using any additional backplane space! **CAPACITY:** 8192 byte (4K word). **ENHANCEMENT FACTOR:** Run time reductions to 40% (70% speed improvement) are achievable. **CACHE PARITY:** Automatically goes off-line in event of any data or address error. **RANGE SELECTION**: User may optimize hit ratio by upper/lower limit switch settings. Cache action monitor indicates hit rate.

CACHE/440™
(4K WORD CACHE MEMORY)

INSTALLS IN: PDP-11/35 and -11/40 **without** using any additional backplane space! **CAPACITY:** 8192 byte (4K word). **ENHANCEMENT FACTOR:** Run time reductions to 40% (70% speed improvement) are achievable. **CACHE PARITY:** Automatically goes off-line in event of any data or address error. **RANGE SELECTION:** User may optimize hit ratio by upper/lower limit switch settings. Cache action monitor indicates hit rate.

EMULOADER™
(ODT/BOOTSTRAP LOADER REPLACEMENT)

INSTALLS IN: PDP-11/05, -11/10, -11/35, -11/40, -11/45, -11/50 and -11/55. **MECHANICAL:** Dual width card replaces standard Unibus termination; requires no additional backplane space. **OPERATING ADVANTAGE:** Provides fixed console emulator (ODT) and bootstrap loaders for DL11, PC11, RF11, RK06, RK11, RP04/05/06, RP11, RS03/04, RX11, TC-11, TM11 and TU16. **SPECIAL FEATURE:** Performs memory diagnostic each time a boot operation is done from ODT.

GENERAL PURPOSE PRODUCTS

QNIVERTER™
(Q-BUS TO UNIBUS CONVERTER OR UNIBUS TO Q-BUS CONVERTER)

INSTALLS IN: LSI-11, LSI-11/23, PDP-11/03 and PDP-11/23 via quad-width card. **APPLICATIONS:** Allows Unibus-compatible controllers and memories to be used with LSI computer systems, or LSI-based peripherals to be used with PDP-11 computer systems. **FEATURES:** Supports features of LSI-11/23 including the full 128K address capability.

REBUS™
(BUS REPEATER – DB11 REPLACEMENT)

INSTALLS IN: All PDP-11's; **without** using any additional backplane space. **MECHANICAL**: One dual-width card plugs into the same pair of connectors as the Unibus extension cable which is then plugged into the REBUS connectors. **COMPATIBILITY**: Allows for 18 additional bus loads and 50 foot bus extension. Requires no software changes. Bus cycle time unaffected for devices on CPU side of REBUS – increased by 250 nsec max. for devices on outboard side.

DUAL I/O™
(GENERAL INTERFACE-DR11-C REPLACEMENT)

INSTALLS IN: All PDP-11's; in any SPC slot via quad-width card. **APPLICATION**: Dual I/O is equivalent to two (2) DR11-C's and provides the logic for program-controlled parallel transfer of 16-bit data between two (2) external user devices and a Unibus system. **OPERATING ADVANTAGE**: Provides user the hardware/software equal to a dual DR11-C in one-half the space and one-half the bus loading of DR11-C's.

INTERLINK/UNI
(DR11-B AND ½ DA11-B REPLACEMENT)

INSTALLS IN: All PDP-11's in any SPC slot via hex-width card. **APPLICATIONS:** Provides full DR11-B (DMA INTERFACE) and one side of DA11-B (UNIBUS LINK) capability on a single card. **OPERATING ADVANTAGES:** Requires only one hex-width card in each computer to effect link vs. full four-slot system unit per computer. Exhibits one bus load. Directly software transparent as a DR11-B replacement or when expanded to DA11-B equivalency.

BUSLINK/UNI, LSI OR U TO Q
(CPU TO CPU LINK: UNIBUS TO UNIBUS, UNIBUS TO Q-BUS OR Q-BUS TO Q-BUS)

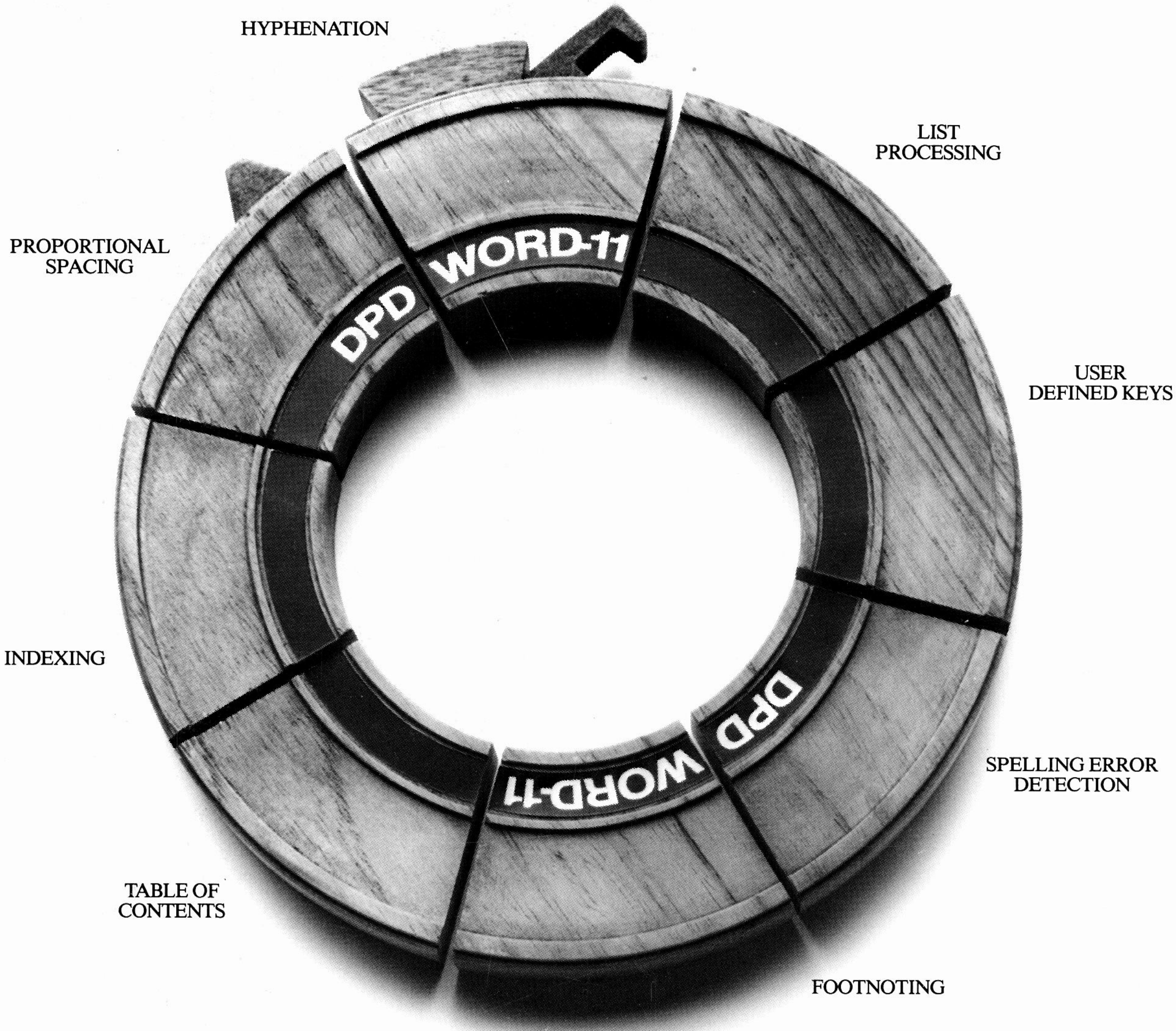
INSTALLS IN: All PDP-11's and/or LSI-11's via pairs of hex-width, hex/quad-width, or quad-width cards and supplied cables. **APPLICATION:** Provides full DA11-B (Unibus or Q-bus link) compatibility on single cards. BUSLINK operates at DA11-B transfer rates over distances of up to 50 feet. **OPERATING ADVANTAGE:** Requires only one card per CPU to effect link at minimal bus loading vs. full system unit per computer.

Able cards give you the best way to make your present PDP-11 run better than ever and avoid up-grading to a more expensive model. We give you time to plan ahead and to control your cash flow with the most sophisticated line on the market today. Our cards are priced competitively. They install in minutes. They provide immediate results. And they always out-perform the competition. They should. We are the only computer people in the computer-system-enhancement business. Our customers will tell you we are the best qualified to help you get the most out of your present PDP-11. Write for details. Able Computer, 1751 Langley Avenue, Irvine, California 92714. (714) 979-7030, TWX 910-595-1729.

Able, the computer experts

DEC, PDP, LSI, UNIBUS, and FASTBUS are registered trademarks of Digital Equipment Corporation.

The Whole Truth About PDP-11 Word Processing.



When you explore word processing systems, you'll find a number of systems that offer part of the package. And frankly, if you're only looking for text editing, almost any system will do.

But if you're looking for full word processing capabilities, you should insist on a system that can handle all of the tough tasks you'll encounter.

For starters, a good system will have list processing, the vital function that generates correspondence, reports, even statistical analyses. It should have user defined keys to simplify repetitive operations.

And a good system can handle the details. Like spelling error detection, automatic renumbering of footnotes, tables of contents and indexes. And proportional spacing and hyphenation for professional looking documents.

A good system can be shared. It should be able to support up to fifty terminals. And it must be backed up by successful installations and a strong service team.

After you've examined the options, we think you'll select WORD-11, the only system with all the sophisticated features you could want.

For the details, please call Data Processing Design Inc., Corporate Office; 181 W. Orangethorpe Ave., Suite F, Placentia, CA 92670. (714) 993-4160. Telex 182-278. New York Office: 420 Lexington, Suite 647, New York, NY 10170. (212) 687-0104.

dpd
Data Processing Design, Inc.
AUTHORIZED digital COMPUTER DISTRIBUTOR

Overseas Distributors: *Management Information Services PTY. LTD.* *Melbourne, Australia* *Jenson LTD.* *Bristol, England* *Systime LTD.* *Leeds, England* *Network Computer Services PTY. LTD.* *Sydney, Australia*