INSIDE:

- Segmenting Basic-Plus-2 Applications
- Points of Interest
- CTRL/F Monitor Support
- Disc Structure Notes
- Beginner's Guide to MACRO 11 Programming in RSTS/E
- QUE.11 System Manager's Guide
- ON2OFF.B2S
- FICHE.BAS
- RSTS/E Monitor Internals — Part 1
- The VAX-SCENE The Other Story About VAX and VMS
- All Things BRIGHT and Beautiful
- BLDCTL: Control File Processor
- DECUS Canada
- Access Control and Utilization Monitor
- PDP/11 Systems — UTILITIES
- New Feature — Technical Notes
- More ...
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.
When low cost and reliability are your print-out needs, Southern Systems has the answer.

Call SSI for versatile, simple-to-operate 200, 300 and 600 lpm printer systems.

Economical purchase price is just the start of your savings with SSI's 200 lpm matrix or 300 and 600 lpm band printer systems.

The 300 and 600 lpm bands, the lowest cost-of-ownership printers in their performance range, give you easy-to-change character font bands, clean cassette ribbons, quiet operation and up to five clear copies.

The 200 lpm matrix, at half the price of the 300 lpm band, gives you two-thirds the performance, with 128 characters in condensed, expanded or standard print and six clear copies.

These versatile, economical printer systems are unbelievably maintenance-free. Compatible with DEC, Data General, HP, Perkin-Elmer and most other computers. Parallel or serial interfacing, 30-day delivery, Nationwide full service. Unmatched printer and interface expertise.

Call SSI, the leader in add-on printer system technology, for low cost, high performance printer systems...at 200, 300 or 600 lines per minute.

Please send information about:
- The 200 lpm matrix; and/or the 300 or 600 lpm band.
- I'd also like to know about your high speed printers:
  - 900 lpm band ... Our newest high-performance product.
  - 300/600/900 lpm drum.
  - 1500 lpm charaband.
  - 600, 1000 or 1200 lpm ChainTrain.
  - Parallel or Serial (Synchronous or asynchronous).

My needs are:
- Immediate
- 3-6 months
- For Information Only.

My computer type is ____________________________

Name ____________________________
Title ____________________________
Company ____________________________
Address ____________________________
City ____________________________ State ____________________________ Zip ____________________________
Telephone ____________________________
Everyday we find the pressure to produce is being pushed on the programming staffs around the world. Programmer productivity is the name of the game.

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

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

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

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

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

Software for DEC RSTS/E Systems
Contents

SEGMENTING BASIC-PLUS-2 APPLICATIONS .................................................. 8
Al Cini
Professor Al continues what we are sure will someday be one of the first RSTS textbooks.

POINTS OF INTEREST ................................................................. 15
Wilf Forrow
My father used to refer to his "storehouse of useless information", that somehow wasn't so useless. Some day these ideas may help you.

CTRL/F MONITOR SUPPORT ......................................................... 18
If you run to another terminal to see what the file status is of your job — stop running. The boys from Software Techniques have come to the rescue again.

DISC STRUCTURE NOTES ............................................................ 25
Dave Mallery
By popular demand we are reprinting excerpts from the wisdom of Dave Mallery on DISK anatomy.

BEGINNER'S GUIDE TO MACRO 11 PROGRAMMING IN RSTS/E ............... 26
Thomas Courtney
Written for SENERUG, the first independent RSTS User's Group. This beginning tutorial will help you get off the ground in MACRO.

QUE.11 SYSTEM MANAGER'S GUIDE ............................................. 34
Carl Marbach
If you would like to replace all of your batch and line printer spoolers, read this.

ON20FF.B2S ................................................................. 40
Dave Schott
Did you ever want to change a line to "MODEM" without shutting the system down? Here's how.

FICHE.BAS ............................................................... 41
Scott Banks and Dave Mallery
If you're considering microfiche directly from your computer, here's a class "A" program. Written by two famous RSTS persons.

RSTS/E MONITOR SUPPORT ..................................................... 43
Mike Mayfield
The RSTS community is fortunate that Mike Mayfield has re-emerged and is producing the kind of articles we need.

THE VAX-SCENE ............................................................... 60
Installation pictorial.

ALL THINGS BRIGHT AND BEAUTIFUL ............................................. 62
Michael J.D. Mowat, B.Sc., M.B.C.S.
More ideas on Disk Structuring from Scotland.

BLDCTL: CONTROL FILE PROCESSOR ......................................... 68
David Spencer
For pure text manipulation, TECO just might be the best way to do it. Here's an example TECO program.

DECUS CANADA .............................................................. 73
Carl Marbach
Montreal, P.Q., was the site of this year's spring "7" DECUS.

ACCESS CONTROL AND UTILIZATION MONITOR ......................... 74
C.M. Battistel and Ed Giovanella
The "Northern" RSTS users should be heard from more often. They have lots to tell us.

PDP/11 SYSTEMS — UTILITIES .................................................. 79
M.A. Jackson
Some helpful BASIC-PLUS-2 utilities from across the sea.

Coming . . .

- Scott Banks on Disk Directories
- Bob "Macro Man" Meyer on How to Macro
- Technical PDP-11 Internals
- Continued RSTS Internals
- The RSTS Pro Installs Their Computer
- Spring Miami Decus Coverage
- RSTS Software Hall of Fame
- VAX X.25 Communications
- Modeling the VAX
- The Doctor Returns
- More . . .

From the Editors .......................................................... 4
Dear RSTS MAN .......................................................... 14
Letters to the RSTS Pro .................................................. 6
Technical Notes .......................................................... 66
Classified .......................................................... 92
News Releases .......................................................... 94
List of Advertisers ...................................................... 95


From the editors...

In case you haven't noticed, the RSTS PRO has created a thriving new marketplace for software. This was one of our original goals, six issues ago. I would like to single out two of the many products that I have personally bought and paid for. The first is OMSI Sort Version 1.6. I have used OMSI sort since its original release as a macro replacement for SQWIK and MQWIK, the old DEC type-1 header sort. It was the best software buy I ever made, yielding staggering reductions in run time. Version 1.6 contains the final additions to a fine product — the macro rendition of SQWIK and OQWIK, the extract and reordering portions of the sort. Amazingly fast and truly flexible. Bravo!

The second entrant to my 'Hall of Fame' is a package from Software Techniques called DSKIT. I ordered it as soon as I saw the proofs of their ad in the last issue. To make a long story short: last week, using DSKIT, I created 130 accounts and fully extended their centered UFD's in three minutes and forty seconds (a job that used to take four to eight hours). I then copied the full contents of a 300 MB RMOS equivalent to this new 'well structured' disk in 45 minutes, optimizing cluster size and contiguity on the process. I (for once) was speechless. Software Techniques delivered excellent response to the few minor problems I unearthed in the very short learning process. The documentation is excellent. In short, this package is the 'final solution' to structured disks, eliminating all of the time and complexity and reducing the job to one of the simplicity of a SAVRES.

Dave Mallory

Love at First Sight

I'm in LOVE. Head-over-heels in love. I am overcome when I see the twinkle in the darkness, but even in the light I am overjoyed when I come into the room. Our relationship has been short, but I am looking forward to many years of happiness. I look forward to that time of day when I can be alone, just the two of us talking to each other.

Now, before I really get into trouble with Helen, my spouse of some years, let me explain that my New Love is a rather inanimate (? sometimes I wonder ?) Lovely, old, well cared for, rugged, PDP 11/40 that we plan to make over into a New, Well Cared for, New peripheral, Speeded up, more functional, Great RSTS Machine. That PDP 11/40 was built in 1973 and even has the old style BA-11 Box and real (!) lights and switches — looks like an 11/70 if you don't look too closely. No light show is evident... no null job?... does any one know why? We don't.

This 11-40 is a real story; how it was bought (and others rejected), how it was delivered, the new memory (256KB on a single board from Data Systems Services), the new disk (160MB Winchester that looks like two RM03's from System Industries), the new Tape (800/1600 of course) also from DSS, the new Multiplexer (A Dmax from Able), the new line printer (Southern Systems), the new cabinet (Surplus RCA), and others good and bad things. We'll tell the story in the the next few issues, don't miss it.

The RDC has invited us to visit and see how they do their remote diagnosis. That story should be ready for the next issue.

Our VAX-SCENE has been well received and it is significantly expanded in this issue and will continue to get the attention it deserves. Lets hear it from all of the VAX people out there — your doing some neat things, tell us about them.

If you haven't seen the NEW RSTS brochure, ask your salesperson for one. This make RSTS seem like the up and coming product it is! Be sure to look closely at the SYSTAT...What are all those foreign(!) Run-Time systems? I had to go to Canada (see the Canadian DECUS article) to get one, I haven't seen a salesperson in over a year. While you have the DEC rep on the phone ask to see the RSTS, VAX and RSTS/VAX performance data that is available from the performance group in Merrimack; let us know if they can't or won't get it for you — it's got some surprises.

We're well on our way into our second year and it is more: more work, more fun, more people, more magazine. RSTS is a growing product and were going to grow along with it — keep giving us the help we need: more articles, more letters, more of you — tell a friend about RSTS and about the RSTS PROFESSIONAL.

Carl Marbach
Software Packages Include:

- Word Processing
- Inventory Control
- Accounting Packages
- Information Storage and Retrieval
- Financial Reports
- Finite Stress Analysis
- Micro-Processor Development
- Commercial Graphics

RENTAL • SALE • SERVICE • SUPPORT

IMMEDIATE DELIVERY FROM STOCK
ALL RENTALS INCLUDE MAINTENANCE, PURCHASE OPTION LEASE PLAN AVAILABLE

HAMilton 212-695-1936
Hamilton Rentals
One Penn Plaza, New York 10119

LONDON PARIS DÜSSELDORF TORONTO/CALGARY
LETTERS to the RSTS Pro...

Dear Dave and Carl,

In your recent report on the U.K. RSTS SIG Meeting you said that it was attended by people "from all over the U.K.". There were also several participants from Ireland — RSTS is alive and well over here too. Keep up the good work. Kind regards.

Yours sincerely,
David A. Reynolds
Peat Marwick Mitchell - Dublin

---

Dear Sir:

In your editorial “Go VAX Young Man!” [RSTS Professional, Vol. 2, #3], you mention that a product which allows the PDP 11/70 clock rate to be raised has appeared. I am interested in such a product, but have been unable to find any information on it.

Would you please send me the names and addresses of those who are marketing this product.

Yours very truly,

N. McRae
Computer Facilities Analyst
Hardy Associates (1978) Ltd., Canada

---

Dear Mr. Mallery:

Your article indicated that there is a "fast clock" available for the DEC PDP 11/70. I would appreciate it if you could tell me the manufacturer so that we can evaluate it as a means to increase our capacity.

H.J. Mainwaring
Staff Engineer, Computer Systems
Cadillac Motor Car Division

You can obtain a modified TIG board for the 70 from:TIG Board, Nassau Systems, P.O. Box 19329, Cincinnati, OH 45219.

This is currently being tested by a major 70 user and the results should be in soon.

---

Dear Sirs,

I have been managing a PDP11/55 time sharing system running under RSTS for less than a year, and when I started it looked as if the system would need to be enhanced in the very near future as the response times to the users was on occasions completely unacceptable. There are about 120 accounts on our system, containing over 9000 files and although some effort is being made to reduce these numbers, it has been difficult to do this since the number of people using the system is large, and the job mix is very varied.

It was with great interest, therefore, that I attended the seminar at the Festival Hall, and I returned full of ideas and enthusiasm. Right, I will rebuild the system disk with nice contiguous directories. Then the trouble started. Did I have to run REACT to generate 124 accounts? No need. System functions are available to write a simple program to establish these the same as the existing disk. Now it is a bit of a fiddle to get the right size of file to fill the lower disk. This is not too difficult, however, so it is done and I start opening null files to fill up all the directories. About three in the morning I give up. It's only half done and it will take all night. I realize that since our system requires new accounts to be installed fairly frequently that it is not a once only job.

Thinking again I realize that a UFD fully extended with one null file is a fairly simple structure so why not build all the UFDs into a large contiguous file and patch the DCN of each into the MFD. The file into which the directories are built is in [1,0] and it may be removed after the build by changing the link. This does not clear the SATT so that the directories are protected. Surprisingly, apart from a minor bug in the alignment of clustersize 8 directories (the DCNs must be odd numbers) this worked first time and a disk containing all the accounts with seven cluster directories was built from an initialized disk in about 12 minutes.

Placed files were then transferred using PIP followed by a wildcard PIP of everything else.

Unfortunately, since our system uses new files first all the directories were reversed, and a time consuming REORDR was required.

The next time the disk was copied the new disk was initialized old files first, and the program INVERT was run to change it after the copy was complete. This saves the REORDR time and builds a better directory. Since the control file started the system after the copy to allow the copy to run unattended at night, the program INVERT must tell the monitor that the change has been made. I use the status bit on the disk is only looked at during the mount operation.

The improvement in system performance was enormous. Not only the directories but all the other files on the new disk were physically contiguous (or nearly so) and the disk access was improved by about 50%. The time for the whole operation was less than the BACKUP that we had used weekly before, and the backup disk was fully readable in the event of a disaster. So subsequently found that the retrieval of a single file from the old disk (using PIP) became a trivial operation as opposed to getting a file from a BACKUP set.

I am enclosing a description of the process and listings of the programs involved for your information and possible publication as I think it may be useful to other users with similar problems.

Yours faithfully,

Michael J.D. Mowat, B.Sc., M.B.C.S.
Dept. of Agriculture and Fisheries for Scotland
P.S. Your articles on directories are very useful. What about something on system table contents. We also feel Michael's article may be helpful to other users. See "All Things BRIGHT and Beautiful", this issue.

---

Editors,

I have been a subscriber to your magazine for over a year and have found much enjoyment from reading the RSTS Professional. It has proven to be an invaluable source of information on the RSTS Operating System and TECO.

Sincerely,

John J. Walczy, Royal Oak, MI

---

Dear Mrs. Noakes,

Following your request for contributions to the RSTS Professional, please find enclosed listings and documentation for half a dozen user subroutines callable from Basic-Plus-2.

I hope someone may be able to make use of them — either they provide functions not available or are very much faster than those provided. The execution time given is for an 11/34A processor.

May I add my thanks and praise for last month's DECUS Commercial SIG meeting which I found interesting and useful (that was the "Dave & Carl Show" at Festival Hall).

Yours sincerely,

M. A. Jackson, Nielsen Business Services

Readers will find Mr. Jackson's subroutines in this issue, "PDP11 — UTILITIES."

---

Dear RSTS Professional:

I would like to take a moment to thank the people who have written the excellent articles for the "RSTS Pro". I'm sure their efforts have been of great use to many people. I would like to especially thank Scott Banks, Steve Davis, Steve Edwards and your own Dave Mallery and Carl Marbach.

I feel we in the RSTS community have a responsibility to provide our peers with this type of information and I applaud those who have already done so. For my part I am starting a year long series of articles on monitor internals. I haven't seen anybody do anything like this yet so I hope it will be helpful. A copy of the first article is enclosed.

Sincerely,

Mike Mayfield
Northwest Digital Software

---

DO YOU REMEMBER THIS?

(Do not remember this?) RSTS Professional Vol. 2, #3, p.75 - STILL!

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

Because no one has gotten this right yet, we'll save the answer 'til the next issue. Following are the latest silly attempts.

"Tampa Elec. Co. truck."

Jeffrey Neu, New York, NY

Wrong!

---

Boys and Girls,

(1) Your mag is getting better and more informative with each issue. Keep up the goodies.

(2) The unresolved TECO "what is it" is a utility-company-type "Cherry Picker", commonly found somewhere near the top of a pole (lower case, no creative ethnic slur intended).

(3) I've enclosed my mailing label. Counting issues, I think my subscription must be about to expire but I need further clues.

Bye/F, Douglas P. Herman
Herman Management Company, Inc.
El Cajon, CA 92021

(1) Thank you. (However, flattery won't get you a T-shirt - usually!).

(2) How TECO, Why TECO remains unresolved.

(3) A clue follows.

(4)

---

Dave, Two of them, even (whatever they are...)

--- Bill, Merrimack, N.H.

Good Grief! They're Even Contemptibly Omni-present.

...continued on page 95
VAXination for DEC* Users.

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

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

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

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

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

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

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

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

*DEC and VAX-11/780 are trademarks of Digital Equipment Corporation.
Segmenting BASIC-PLUS-2 Applications

By Al Cini, Computer Methods Corporation

This is first in a series of articles on analysis and design of BASIC-PLUS-2 Applications.

INTRODUCTION

The advantages of a "modular" (many parts) over a "monolithic" (all-one-piece) software system are, these days, a matter of conventional wisdom. Most of us somehow know we write modular programs, but few of us stop to reflect on how or why we do.

Modular software system concepts are adapted from mechanical and electronic engineering notions, whose practitioners learned a long time ago how expensive "one big thing" can be to build, repair, and modify. In fact, while large-scale integration gets the credit for the third generation of computers, its the shrewd packaging of these chips into functionally independent components, and the creative assembly of these components into ever-expanding variety of processors and peripherals, that have made computing the boom industry it is.

It's largely the same in software. Functionally independent modules allow us to work separately on different elements of a project, with confidence that these conceptual "parts" will work properly together when ultimately assembled. Throughout all of engineering, a complete and cohesive design and faithful adherence to its rules during implementation go a long way toward taking the worry out of laying that last brick, soldering that last chip, and taskbuilding that last program. Having to tear up a routine because it doesn't "fit" into a system isn't any less exasperating than smoking a printed circuit board in a poorly designed backplane, or cracking the block of an expensive test engine—anticipating the connections, strains, and pressures among the parts assures the correct integration of the whole, and avoids costly trips back to the drawing board.

UNDERSTANDING MODULES AND MODULARITY

In programming, a module is a group of program statements with the following characteristics (Myers, p. 11):

1. The statements are contiguous (i.e., appear together in the program listing).
2. The statements are bounded by identifiable delimiters (such as SUB/SUBEND, DEF/FNEND statements; note that in the case of GOSUB, these boundaries can be very vague).
3. The statements are collectively referenced by a name (or statement number).
4. The statements can be referenced by the module name, from any other part of the program (a recursive module can even call itself).

The term "module" usually evokes thoughts of external subprograms (PL/1 procedures, FORTRAN subroutines), but its broad definition includes internally contained code sections (COBOL paragraphs, BASIC-PLUS functions and GOSUB-type subroutines) as well as complete concurrently or serially executed programs. Logically, software modules are the realization of our ability to analyze a big problem into manageable parts. In practical terms, software modules determine the feasibility of our work: we arrange them to promote system reliability, simplify maintenance delegate coding responsibility, and optimize memory utilization.

DETERMINANTS OF "GOOD" SOFTWARE MODULES

Before we go any further, I'd like to apologize for the inevitable use of the word "structure" in the text which follows. To say that "structure" in our business is an overworked word is to say that France is a place where a lot of French people live. Introduced formally during the late sixties, "structure" as it modified "programming" identified a coding discipline in which procedural programming languages (RPG isn't one of these: COBOL, BASIC, and FORTRAN are) could be organized into programs using combinations of three fundamental control mechanisms: sequence (one statement after another), selection (IF-THEN-ELSE), and iteration (DO-UNTIL, WHILE-NEXT). Structured programming stirred up some pseudo-controversy (GO TO statements, as it happens, are not included in the basic control devices), got lots of trade press attention, and spawned a host of "structured" disciplines as well as whole libraries of pamphlets and books on each. (A discussion of structured BASIC-PLUS and BASIC-PLUS-2 programs can be found in RP vol. 1 no. 1). "Structure" has become an industry buzzword which raises hackles, provokes debate, and unfortunately has come to have little useful meaning.
This message is for Managers, Project Leaders, and Programming Supervisors who just know there has to be a better way!

CAN WE INTEREST YOU IN A NEW PROGRAM DEVELOPMENT CYCLE?

BASIC-PLUS(-2) SOFTWARE ENGINEERING STRATEGIES
A 5-day training program conducted at your site — detailing analysis, design and programming methods aimed at productivity-minded software engineering professionals. Costs less for a group of as few as five students than comparable off-site education. Study these highlights from the course syllabus:

- Understanding reliability as a design objective.
- Structured programming.
- BASIC-PLUS(-2) language specifics.
- Designing "from the top."
- Introduction to data flow analysis.
- Hierarchy analysis.
- Transform analysis.
- Transaction analysis.
- Data base design techniques for RMS files.
- "Relational" RMS implementation.
- Elements of a RSTS software project.
- BASIC-PLUS-2 program segmentation.

COMPUTER METHODS CORPORATION
SKILLS DEVELOPMENT PROGRAM
P.O. BOX 592, MOORESTOWN, N.J. 08057
609/778-8440

I want to learn more.

Name __________________________
Title/Company __________________
Phone _________________________
Address ________________________

I'm particularly interested in these training areas:

□ Send information on your off-site training calendar.
□ Have someone call ____________________ area code

IN A HURRY? Give us a call at (609) 778-8400
For our purposes, we can define "structure" along the lines of its non-DP usage: a structure is a complex entity consisting of discrete but interrelated parts. The parts can be isolated and examined independently and, frequently, are themselves structures of smaller parts. The smallest possible parts in programming are sequences, which may appear within selection or iteration control mechanisms. Modules are composed of such elements, and combine with other modules to make programs and systems.

The "goodness" of a module can be evaluated along two dimensions (after Stevens et. al.): cohesion and coupling (figure 1).

**FIGURE 1. Dimensions of Module Strength**

Cohesion relates to the strength of the relationship among the elements which comprise a module with respect to the function a module performs. A highly cohesive module is one in which the elements work together to contribute to the execution of a specific function; an incohesive module is one in which the elements are assembled at random. As you might imagine, there are strata of modular cohesiveness along the continuum:

1. Coincidental. When there is no functional relationship among the elements in a module, we can say that the module is coincidentally cohesive. FORTRAN or BASIC-PLUS-2 programs which are too large to fit in memory can sometimes be "hacked" into overlays by randomly assigning program parts to subroutines; such subroutines are coincidentally cohesive modules.

2. Logical. Modules whose components are similar with respect to the work they do, but which do not contribute to the execution of a specific function, are logically cohesive. For example, including all READ statements in a READ subroutine, or building a subroutine which edits all social security, phone, and zip code numbers, are instances of logical modularity. "ON ERROR GO TO 19000" essentially invokes a logical module whose job is to "handle all errors."

3. Temporal. Like logically bound modules, temporally bound modules consist of elements which perform similar services. In addition, however, such elements in temporally cohesive modules are related by the time when their services are called for within a program. Housekeeping procedures like "initialization", "termination", and "clean-up" routines are examples of temporally cohesive modules.

4. Functional. A functionally cohesive module is one in which all constituent elements contribute to the execution of a single function. The function, further, is keyed to an operation within the overall application; it is not a computer function (read, or write), but an application function (accept telephone number, compute overtime pay). Functional cohesion is the strongest form of intramodular relationship.

In discussing functional modules, analogies with mathematical functions are frequently drawn. In mathematics, "Y = F(X)" represents a mapping of a domain of parametric inputs (X) onto a range of outputs (Y) according to a set of transformation rules (F). Likewise, a computer system transforms a set of input data into a required set of output data via a precisely defined series of procedures. A functional module is one which performs one specific function against a set of input data, producing a precisely determined set of output data.

Coupling describes the devices by which a calling program communicates with a called module. In general, low coupling—in which the connection is minimal and isolated—is better than high coupling—in which the connection is complicated and diffuse.

The discrete argument list (such as that between a calling program and a separately compiled FORTRAN or BASIC-PLUS-2 subprogram) represents the lowest form of coupling mechanism, because it provides a simple interface between program components which is clearly identifiable in the listings. A functional module which communicates via an argument list can be removed from a system and tested independently without any special consideration.

Named COMMON areas (available in FORTRAN and BASIC-PLUS-2 environments) offer a more complicated interface (the relationship between a COM or MAP and its host program/called subprograms is not clear in the listings), but implementation problems like limits on the size of argument lists make them necessary.

Blank COMMON areas are treacherously easy ways of hooking two programs together in a hurry, but extremely complicated to sort out during maintenance. They should be avoided.

Data reference coupling—when a module directly references a data element within its calling program without going through the argument list—is the only way to connect a calling BP2 program with an internal function or GOSUB subroutine (in DEF/FNEND, the argument list, which is read-only within the function, is of limited value).
In this case, a "conceptual" list of data items passed between caller and called must be kept in the programmer's head and documented in comments, since it doesn't appear in the listing.

Code coupling—when a module executes statements in its caller or vice-versa except through the normal call/return devices—is the highest form of coupling and the source of the worst kinds of maintenance headaches. GOSUB subroutines and COBOL paragraphs should be clearly delimited within the program, and should NEVER branch outside their boundaries.

**STRUCTURED DESIGN.**

There are several approaches to "structured design" in the literature these days, all of which offer a different way of representing the relationship among calling and called modules in a system. We can represent a "hierarchy" of functions and sub-functions pictorially using a hierarchy chart (figure 2). In such a chart, calling programs are arranged in super-ordinate positions with respect to called sub-programs, and the functional "parts explosion" of an application becomes evident.

![Hierarchy Chart](image)

**FIGURE 2. Hierarchy Chart**

For example, let's build an interactive display program which accepts a telephone number from a CRT, then displays either the name/address of the telephone owner or a "no such phone number" message (figure 3).

The hierarchy chart for our hypothetical phone inquiry system shows that we've refined the overall phone inquiry job into four sub-functions: accept a phone number from the terminal; look for the phone number in a file; display phone information; and display error message. Each of these "modules" represents an independently programmable/testable function. The numbering scheme in the hierarchy chart is patterned after IBM "HIPO" documentation standards, and may be replaced by program names or any other unique identifier.

The Get Phone Number module (2.0) accepts no arguments from its caller (in fact, certain environment-specific arguments, such as logical units numbers, might need to be passed—we've omitted them from this functional diagram) and returns either a collected phone number or a flag which signals that the operator is finished. The "finish" flag can be set by typing escape or CTRL/Z or a null phone number; in any case, it's important to note that the decision about what to do when the flag is raised is the responsibility of the caller (1.0), which will execute the functions on level 1 iteratively until then (see the program listings in figure 4).

![Phone Inquiry System Hierarchy Chart](image)

**FIGURE 3. Phone Inquiry System Hierarchy Chart**

The Lookup Phone Number module (3.0) accepts a phone number from the caller, and looks for it in a phone number file (this is just a GET statement with some error handling using BP2 and RMS). If it finds it, it returns a record containing information about the phone number account. If not, it raises an error flag. Again, it is the responsibility of module 1.0 to interpret and deal with the error flag. In this case, the caller decides whether to call module 4.0 (Display the Record) or 5.0 (Display Error) based on information returned by 3.0.

Inspecting the listings, you'll find that 2.0, 4.0, and 5.0 call screen formatting functions which do not appear on the hierarchy chart. Screen formatting routines, and other utility service routines which are frequently called at the lower hierarchy levels are really separate "support hierarchies" which can be documented separately (RMS routines are a good example of a support hierarchy; such routines are excellent candidates for overlay co-trees,
DO YOU HAVE A HARD TIME PROVING YOUR BILLS TO YOUR PDP-11 USERS?

The Billings 11 Program From Mark Ops Gives You:

- Session by Session Accounting
- Variable Time of Day Pricing Schemes
- Bulk Rate Storage and Block by Block Pricing
- Summaries by Session Identification (15 chars)
- Crash and Error Recovery
- Surcharges for Special Software

Fully Supported — Installation Time about 1 Hour
Call Tom Courtney (617) 266-1930 for a set of report samples or write to:

Mark Ops 636 Beacon Street
Boston, MA. 02215

which is a topic for a future article). The listings also show that the contents of any flag variables are never assumed; that is, all flags are explicitly set or cleared within the module whose function determines the value of the flag. A flag variable should never be assumed to be initially zero/non-zero, and it should never be necessary to initialize a flag you expect back from a subprogram before calling the subprogram.

THE EFFECTS OF FUNCTIONAL MODULARITY ON SYSTEM CHARACTERISTICS.

A system which is refined into functional modules is less likely to fail during integration testing (since the integration testing is built into the design), more readily adapted as user needs change (the functions of each module don't change, but new ones may be added and called from superordinate modules to handle new requests), and more readily repaired after a failure (fault isolation is keyed directly to the function which is performed incorrectly, and the failing software module can be removed and tested/debugged independently of the rest of the system). In addition, functionally modular systems tend to be more available (unrelated parts of the system can be kept in service while other parts are repaired) and efficient (the hierarchy chart is a different way of expressing the ODL file; overlays are a design product rather than an after-the-fact hatchet job).

REFERENCES


HIPO — A Design and Documentation Technique, IBM, GC20-1851-1, 1975.

FORTRAN PAYROLL PACKAGE

SOURCE, COMMAND FILES and DOCUMENTATION for a COMPLETE SYSTEM with FIVE SPECIAL DEDUCTION, CHECK WRITING, and REPORT GENERATION.

(Sample State Income Tax Module included.)

GOLDEN TRIANGLE COMPUTERS
Phone 713/883-4588
Box 1866, Orange, TX 77630
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.

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

Multipoint multiplexing lowers costs dramatically.

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. )

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

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

DEAR RSTS MAN: Perhaps you can explain the why's and wherefores of the drop/regain temporary privileges SYS call particularly as they are used by $QUE and $ATPK when both are protected <232>.

I have had 2 problems — one with $QUE, the other with $ATPK.

$QUE — Trying to queue a job from a program using the method given in the Systems User Guide (QUE.11 User's Guide, RSTS Professional, Vol. 2, 4), QUE "bombed" with a "Protection Violation at line 1040" error message. The particular line was the one opening $QUE.SYS for read access. That statement was preceded by the SYS call trying to "regain" its temporarily dropped privileges. $QUE.SYS had a protection code of < 60 > at the time but why does that make a difference to a program with privileges? — The error did not happen if my calling program was itself privileged.

$ATPK — Trying to perform the contents of an indirect command file using $ATPK. I also chained to it from a user program. Its code causes it to drop its privileges temporarily — read Monitor tables I and II — regain its privileges and perform several PEEK functions. Of course, it too died when executing the first PEEK even though it had allegedly regained its privileges. However, it works if called as a CCL and also works if the calling program is privileged.

I have performed some of these tests on two other machines — one running Release 7.0 and one running Release 6C — results were the same on all machines.

What's going on — HELP!

Mike Farrell

Dear Mike: Old RSTS adage: "When you chain from non-privileged to a privileged program at a line number, the privileged program permanently drops privileges." Be glad that it does! That's why CCLs have PRIV in their definitions.

DEAR RSTS MAN: Regarding your answer to 'Frozen Solid,' I've had the same problem of two programs in DIBOL UPDATE mode locking each other — and themselves — out of a record. I work on a PDP-11/70 using 6C; the friend who introduced me to RSTS PROFESSIONAL has also experienced the problem under 7.0.

The solution you gave may work in BASIC-PLUS, but it fails in DIBOL. In DIBOL, one may access records that cross block or cluster boundaries, which means that a lock on one record may in fact lock two blocks. The second program may access the first block properly but find the second locked; when it goes back to get the record again, it finds the first block locked — by its read the first time through.

The addition of one line of code, however, frees things up. The following is our standard solution to the problem (minus our own external subroutines, which add nothing to the understanding of the situation):

Dear Mike: Old RSTS adage: "When you chain from non-privileged to a privileged program at a line number, the privileged program permanently drops privileges." Be glad that it does! That's why CCLs have PRIV in their definitions.

DEAR RSTS MAN: Regarding your answer to 'Frozen Solid,' I've had the same problem of two programs in DIBOL UPDATE mode locking each other — and themselves — out of a record. I work on a PDP-11/70 using 6C; the friend who introduced me to RSTS PROFESSIONAL has also experienced the problem under 7.0.

The solution you gave may work in BASIC-PLUS, but it fails in DIBOL. In DIBOL, one may access records that cross block or cluster boundaries, which means that a lock on one record may in fact lock two blocks. The second program may access the first block properly but find the second locked; when it goes back to get the record again, it finds the first block locked — by its read the first time through.

The addition of one line of code, however, frees things up. The following is our standard solution to the problem (minus our own external subroutines, which add nothing to the understanding of the situation):

DEAR RSTS MAN: Perhaps you can explain the why's and wherefores of the drop/regain temporary privileges SYS call particularly as they are used by $QUE and $ATPK when both are protected <232>.

I have had 2 problems — one with $QUE, the other with $ATPK.

$QUE — Trying to queue a job from a program using the method given in the Systems User Guide (QUE.11 User's Guide, RSTS Professional, Vol. 2, 4), QUE "bombed" with a "Protection Violation at line 1040" error message. The particular line was the one opening $QUE.SYS for read access. That statement was preceded by the SYS call trying to "regain" its temporarily dropped privileges. $QUE.SYS had a protection code of < 60 > at the time but why does that make a difference to a program with privileges? — The error did not happen if my calling program was itself privileged.

$ATPK — Trying to perform the contents of an indirect command file using $ATPK. I also chained to it from a user program. Its code causes it to drop its privileges temporarily — read Monitor tables I and II — regain its privileges and perform several PEEK functions. Of course, it too died when executing the first PEEK even though it had allegedly regained its privileges. However, it works if called as a CCL and also works if the calling program is privileged.

I have performed some of these tests on two other machines — one running Release 7.0 and one running Release 6C — results were the same on all machines.

What's going on — HELP!

Mike Farrell

Dear Mike: Old RSTS adage: "When you chain from non-privileged to a privileged program at a line number, the privileged program permanently drops privileges." Be glad that it does! That's why CCLs have PRIV in their definitions.

DEAR RSTS MAN: Regarding your answer to 'Frozen Solid,' I've had the same problem of two programs in DIBOL UPDATE mode locking each other — and themselves — out of a record. I work on a PDP-11/70 using 6C; the friend who introduced me to RSTS PROFESSIONAL has also experienced the problem under 7.0.

The solution you gave may work in BASIC-PLUS, but it fails in DIBOL. In DIBOL, one may access records that cross block or cluster boundaries, which means that a lock on one record may in fact lock two blocks. The second program may access the first block properly but find the second locked; when it goes back to get the record again, it finds the first block locked — by its read the first time through.

The addition of one line of code, however, frees things up. The following is our standard solution to the problem (minus our own external subroutines, which add nothing to the understanding of the situation):

How to count your chickens before they hatch.

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

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

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

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

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

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

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

New York • (212) 222-2784
Chicago • (312) 876-1081
Houston • (713) 966-0848
San Francisco • (415) 956-1178
Toronto • (416) 245-8473

FINAR
Points of Interest
By Wilf Forrow, Schroder Life Group, Portsmouth P01 2AW

TERMINAL PRINTERS

RSTS normally outputs data quite happily to a powered down terminal. This is a problem for a printer such as an LA120, especially if the spooler deletes the file thinking it was printed O.K. Does anyone know how to prevent this other than by remembering to type control/S before powering the printer down?

MAGNETIC TAPE DENSITIES

Who thinks 1600 bit per inch is risky for secure tapes? I used to, but I have changed my mind. The U.S. National Bureau of Standards recommend that 800 bit per inch be avoided as this is the least reliable of all current densities. (Presumably, 1600 bpi is safer because of the different recording method). I still get some mag tape errors in my error log, but at least a full RM03 back up fits on two tapes instead of three and runs considerably faster.

The bureau also recommends for long term storage without loss of data:
- Purchase good quality tapes.
- Avoid tapes with more than 4 or 5 write skips.
- Avoid tapes over 5 years old.
- Transport and store tapes vertically in cannisters.
- Keep tapes clean, at constant humidity and temperature.

DISABLING TERMINAL KEYS ON VT100's, LA120's ETC.

Some installations may want to disable selected keys, for example CONTROL, NO SCROLL, BACK SPACE, BREAK. Here is a simple method which is reliable, reversible, cheap and will not invalidate your warranty. It works by placing a spacer below the keytop so that the key is locked in the 'up' position (so you can immediately tell that the key is disabled). A section from a plastic book-binder is ideal — do not use metal in case it falls into the electronics.

1. Obtain a length of plastic bookbinder (5/16 of an inch internal measurement, approximately).
2. Cut and file smooth a section approximately 5/32 of an inch long. (cut several and keep the best).
3. Gently lever off the keytop.
4. If the plunger comes out with the keytop, remove it and put it back in the keyboard.
5. Insert the spacer below the shoulders. It should be a tight fit.
6. Replace keytop and check that the key is disabled.
DEC Terminals, Peripherals and Supplies?

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

<table>
<thead>
<tr>
<th>DEC Peripherals</th>
<th>DEC Terminals</th>
<th>DEC Supplies</th>
</tr>
</thead>
<tbody>
<tr>
<td>BA11KE</td>
<td>LA 34 DECwriter IV</td>
<td>Disk Cartridges and Packs</td>
</tr>
<tr>
<td>DD11CK/DK</td>
<td>LA 36 DECwriter II</td>
<td>Diskettes</td>
</tr>
<tr>
<td>DH11AD</td>
<td>LA 38 DECwriter IV</td>
<td>Diskette Storage Units</td>
</tr>
<tr>
<td>DL11E</td>
<td>LA 120 DECwriter III</td>
<td>Mag Tape</td>
</tr>
<tr>
<td>DZ11-A,B,C,D,E,F</td>
<td>LA 180 DECprinter I</td>
<td>Ribbons</td>
</tr>
<tr>
<td>FP11-A</td>
<td>VT 50 DECscope CRT</td>
<td>Paper</td>
</tr>
<tr>
<td>MS11LB</td>
<td>VT 52 DECscope CRT</td>
<td>Forms Caddy</td>
</tr>
<tr>
<td>MS11LD</td>
<td>VT 100 DECscope CRT</td>
<td>Anti-Static Rugs</td>
</tr>
<tr>
<td>MSV11DD</td>
<td>VT 103 DECscope CRT</td>
<td>Print wheels</td>
</tr>
<tr>
<td>DLV11J</td>
<td>VT 132 DECscope CRT</td>
<td>Terminal/Printer Stands</td>
</tr>
<tr>
<td>RM05-AA</td>
<td></td>
<td>Forms Bursters</td>
</tr>
<tr>
<td>RK07EA</td>
<td></td>
<td>and Decollators</td>
</tr>
<tr>
<td>RK711EA</td>
<td></td>
<td>Check Signers</td>
</tr>
<tr>
<td>RL02-AK</td>
<td></td>
<td>and a whole lot more</td>
</tr>
<tr>
<td>RL211-AK</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RL01-AK</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RL11-AK</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RLV21</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RLV11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RX211-BA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RXV21-BA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TJE16-EA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TWE16-EA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TWU45-AA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>KDF11-AD,HD,HF,HH,HK</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DR11C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MR11EA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VT103-AA,BA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Call us at (904) 434-1022</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Call us for our complete catalogue:
1-800-874-9748.
In Florida, call (904) 434-1022.
CTRL/F Monitor Support

Version: V7.0-07

Copyright (C) 1981 by Software Techniques, Inc.

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

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

Nothing is more inevitable than an idea whose time has come.
- Emerson

1.0 Introduction

When RSTS/E V7.0 was (finally) released in early 1980 and the Large File Processor was unveiled it became clear that information about open files was now contained in memory (as opposed to being pointed to on disk.) I set about at that point to develop a routine that would allow users of the system to strike a control key to display their open files, much the same way CTRL/T displays a "mini-SYSTAT" for the user.

With all the information readily available in memory it merely became a matter of finding out where the information was and how it was stored. In addition, the Terminal Driver would have to be modified to support this added functionality.

It turned out to be fairly easy to solve the above problems. However, writing the CTRL/F code itself turned out to be quite an effort. I had a large amount of information, and several different display formats. Not to mention the trouble of debugging. Many lonely hours of using MONODT and a listing of the TTOPNF code were spent before the final version was complete.

What follows is the installation procedure for CTRL/F Open Files support for your RSTS/E V7.0-07 Large File Monitor. The installation of this option allows the user to strike the <CTRL> and <F> keys simultaneously to get a listing of the files currently open by the job.

TTOPNF WILL ONLY WORK ON LARGE FILE SYSTEMS.

2.0 Installation

Rather than get into a very tedious technical discussion of the exact mechanics of TTOPNF, I will merely present how it may be included in your RSTS/E monitor. To install TTOPNF requires a SYSGEN. Because modifications to the standard files is required, the SYSGEN should be run off disk (as opposed to tape). To do this copy all the files off your SYSGEN tape into the same account as the TTOPNF files. Several things have to be done before you can run the standard SYSGEN procedure to include TTOPNF support. These are:

1. Assemble TTOPNF.MAC
2. Modify the terminal driver (TTDVR.MAC)
3. Insert TTOPNF.OBJ in the RSTS.OBJ library
4. After SYSGEN dialog, modify SYSGEN.CTL to include TTOPNF.

2.1 TTOPNF.MAC

It should be noted that the source as it stands now has been thoroughly tested at several sites for well over a year. Beware, modifications can lead you into trouble.

The assembly procedure for TTOPNF.MAC is as follows:

RUN $MACRO
*TTSYSF,TTSYSF/C=COMMON,KERNEL,TTSYSF

ERRORS DETECTED: 0
* ^Z

NOTE
The files KERNEL.MAC and COMMON.MAC can be located on your RSTS/E Sysgen Medium.

2.2 TTDVR.TEC

To make as simple as possible to modify the terminal driver (TTDVR.MAC on your Sysgen Medium), I developed a TECO macro that will modify TTDVR.MAC to include support for TTOPNF. To modify the standard TTDVR.MAC file found on your RSTS/E distribution, place it in the same account as TTDVR.TEC. Then run TTDVR.TEC. It should produce output similar to the following.

RUN TTDVR.TEC

Found 'TTDVR.MAC' working

When TTDVR.TEC is through the copy of TTDVR.MAC in that account will have been patched to support TTOPNF. If any errors occur they will be TECO errors. Analyze the error and take appropriate action to correct the problem.

2.3 Inserting TTOPNF.OBJ in RSTS.OBJ

To insert the TTOPNF module in RSTS.OBJ requires the use of the program LIBR.SAV. This program should currently be in your system library ([1,2]). If not there it can be found on your RSTS/E Sysgen Distribution. Follow this procedure to accomplish the insertion.

RUN $LIBR.SAV
*RSTS/X/N=RSTS,TTOPNF
* ^Z

TTOPNF has been sucessfully inserted if no errors occurred.
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 cluster sizes
- 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, extension, 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
2.4 Modifying SYSGEN.CTL

At this point you should run the SYSGEN.SAV program and configure your system. After this is completed, a small modification must be done to SYSGEN.CTL. During the LINKing of the TER phase of the monitor an additional library search must be performed for TTOPNF. If you have included CTRL/T mini-systat support then insert the text "TTOPNF" directly below that in the SYSGEN.CTL file as shown.

```bash
$R LINK.SAV
TER/Z,TER/A/W,TER=IN:TER,DK:RSTS.STB/X/B: 117000/U: 1000/I/C

TTDVR/C
TERPAT
TTSYS
TTOPNF

$R LINK.SAV
```

If TTOPNF has not been included insert the text following "TERPAT".

3.0 Appearance

After the new monitor SIL has been installed striking a CTRL/F sequence will produce a listing of your currently open files. It will look similar to this:

```bash
>TEKB
TEKB
1 KB4
8 DBO [1,71] TTOPNF.MAC < 60 > 0 0 16
15 DBO [1,2] TKB TKB < 104 > 142 169 32 CR

Most of the information is self explanatory. From left to right: The channel number, the device open on that channel, the project-programmer number, the file name (the '"' indicates the file has been marked for deletion or is tentative), the protection code, the current virtual block number of the file being accessed, the current virtual size of the file, the clustersize, and then a set of one-character flags. The flags:

C The file is contiguous.
R The file is open in read-only mode.
U The file is open in special file update mode.
L The file has been "placed" on the disk at a specific device cluster.
# The file has a block or range of blocks locked.
P The file is permanent and may not be killed or renamed.
```

There are two special cases of file opens that are seen by TTOPNF. These are UFD opens and "non-file structured" disk opens. They produce output similar to that below.

```bash
1 DBO Non-file structured 0 495513 8 inherits
2 DBO [1,2] User-file directory 0 112 16 PR
```

On the disk open the "size" becomes the number of virtual blocks the disk contains, and the clustersize is the device cluster size. UFD information is the same as file information. Note the "PR" flags on the UFD open.

4.0 Optional Patch

It may be desirable to allow non-privileged users the ability to use CTRL/F. A patch can be made to the built monitor to enable this. The patch procedure is shown below.

```bash
RUN $0NPAT
File to patch? (Monitor SIL with TTOPNF support)
Module name? TER
Base address? ..PRVF
Offset address? 0
Base Offset Old New?
137554 000000 001404 ? NOP
137554 000002 032761 ? C (UP-ARROW C TO EXIT)
```

5.0 Caution

Many letter quality printers use a CTRL/F sequence for acknowledgment. To prevent the monitor from intercepting this sequence set NO CTRL/R on these terminals.

6.0 Information

For any further help or comments contact:

Steven P. Davis
Director Software Engineering
Software Techniques, Inc.
5242 Katella Ave. Suite #101
Los Alamitos, CA 90720

7.0 Acknowledgements

The routines OUTPNT, OUTSIZ, OUTCHR, and DODIVS are slightly modified versions of the same routines found in the TTSYST support code of the RSTS/E monitor.

8.0 TTOPNF.MAC

```bash
TITLE TTOPNF, CTRL/F OPEN FILES

WRITE BY: STEVEN P. DAVIS
COPYRIGHT (C) 1980, 1981
SOFTWARE TECHNIQUES, INC., LOS ALAMITOS, CA 90720

THIS SOFTWARE IS PROTECTED FREE OF CHARGE TO MEMBERS OF THE NORTH
AMERICAN DISCUS ORGANIZATION AND MAY BE COPIED ONLY WITH THE
INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE, OR ANY
OTHER COPIES THEROF, 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.
```

For any further help or comments contact:

Steven P. Davis
Director Software Engineering
Software Techniques, Inc.
5242 Katella Ave. Suite #101
Los Alamitos, CA 90720

7.0 Acknowledgements

The routines OUTPNT, OUTSIZ, OUTCHR, and DODIVS are slightly modified versions of the same routines found in the TTSYST support code of the RSTS/E monitor.
We’ve got the best combination of word processing and data management available.

Other companies tried to overlay data processing onto existing word processing software systems. It didn’t work too well. There were just too many unreasonable restrictions: like small file sizes and cumbersome sorting limitations.

We’re a little hungrier, so we scrambled, wrestled, and finally arrived at the best word processing-data management system on the market.

We integrated our own proven QDMS data management system with our new QWORD word processing capability and came up with pure dynamite. Then we simplified it so that anyone can handle it.

You can buy a system that includes a PDP11/23, RSTS/E operating system, 192K bytes of memory, multiplexor, 20 Megabytes of removable disk storage on two drives, video console, QDMS and QWORD software for under $43,000. And the computer system is maintained by DEC.

In short, it’s the best.

Call, don’t write Tom Schreier at (203) 728-6777.
...
RABBIT-4 FILE SECURITY
GUNS DOWN DATA RUSTLERS
ON RSTS/E SYSTEMS

Do You Know Who The Pesky Varmints
Reading Your Confidential Information Are?

RABBIT-4 will help you track’em and catch’em,
cause it has a proven record of performance.
RABBIT-4 will let you:
☐ Log secured file accesses
☐ Signal OPSER of violations
☐ Roll-out the bandits
☐ Freeze system activities with
  6 levels of file security to keep your data safe
and secure, RABBIT-4 will also:
• Secure up to 64 data files
• Provide 32 user descriptions plus wild cards
• Restrict file access to specified programs
• Identify intrusions and intruders
• Recover automatically from system crashes

SEE OUR TALENTED RABBITS AT THE MAY DECUS MEETING
and pick up a free Rabbit Coloring Book Souvenir!
We’ll have a hospitality suite at the Eden Roc Hotel
where we’ll demonstrate Rabbit’s resourcefulness in
* Performance Analysis * Job Accounting * Resource Accounting
* Data Management * Financial Planning * File Security

RAXCO INC.
3336 N. Flagler Drive, West Palm Beach, FL 33407 (305) 842-2115
OUTCHR: HOV

OUTSIZ: 100$

OUTCHR - PRINT ONE OR TWO CHARACTERS

OUTSIZ - PRINT A NUMBER 0-255 WITH LEADING ZERO SUPPRESSION

FILNAM - PRINT OUT A FILENAME

RS - ADD

CALL OUTCHR, RS

R2-RS RANDOM

CALL OUTSIZ

R2-R4 RANDOM

BISB

R2

MOV

..EVEN

WORD 1000, 100, 10

BYTE BYTE1, BYTE2

OUTCHR - PRINT ONE OR TWO CHARACTERS.

OUTSIZ - PRINT A NUMBER 0-255 WITH LEADING ZERO SUPPRESSION.

FILNAM - PRINT OUT A FILENAME

RS - ADD

CALL OUTCHR, RS

R2-RS RANDOM

CALL OUTSIZ

R2-R4 RANDOM

BISB

R2

MOV

..EVEN

WORD 1000, 100, 10

BYTE BYTE1, BYTE2

OUTCHR - PRINT ONE OR TWO CHARACTERS.

OUTSIZ - PRINT A NUMBER 0-255 WITH LEADING ZERO SUPPRESSION.

FILNAM - PRINT OUT A FILENAME

RS - ADD

CALL OUTCHR, RS

R2-RS RANDOM

CALL OUTSIZ

R2-R4 RANDOM

BISB

R2

MOV

..EVEN

WORD 1000, 100, 10

BYTE BYTE1, BYTE2

OUTCHR - PRINT ONE OR TWO CHARACTERS.

OUTSIZ - PRINT A NUMBER 0-255 WITH LEADING ZERO SUPPRESSION.

FILNAM - PRINT OUT A FILENAME

RS - ADD

CALL OUTCHR, RS

R2-RS RANDOM

CALL OUTSIZ

R2-R4 RANDOM

BISB

R2

MOV

..EVEN

WORD 1000, 100, 10

BYTE BYTE1, BYTE2

OUTCHR - PRINT ONE OR TWO CHARACTERS.

OUTSIZ - PRINT A NUMBER 0-255 WITH LEADING ZERO SUPPRESSION.

FILNAM - PRINT OUT A FILENAME

RS - ADD

CALL OUTCHR, RS

R2-RS RANDOM

CALL OUTSIZ

R2-R4 RANDOM

BISB

R2

MOV

..EVEN

WORD 1000, 100, 10

BYTE BYTE1, BYTE2

OUTCHR - PRINT ONE OR TWO CHARACTERS.

OUTSIZ - PRINT A NUMBER 0-255 WITH LEADING ZERO SUPPRESSION.

FILNAM - PRINT OUT A FILENAME

RS - ADD

CALL OUTCHR, RS

R2-RS RANDOM

CALL OUTSIZ

R2-R4 RANDOM

BISB

R2

MOV

..EVEN

WORD 1000, 100, 10

BYTE BYTE1, BYTE2

OUTCHR - PRINT ONE OR TWO CHARACTERS.

OUTSIZ - PRINT A NUMBER 0-255 WITH LEADING ZERO SUPPRESSION.

FILNAM - PRINT OUT A FILENAME

RS - ADD

CALL OUTCHR, RS

R2-RS RANDOM

CALL OUTSIZ

R2-R4 RANDOM

BISB

R2

MOV

..EVEN

WORD 1000, 100, 10

BYTE BYTE1, BYTE2

OUTCHR - PRINT ONE OR TWO CHARACTERS.

OUTSIZ - PRINT A NUMBER 0-255 WITH LEADING ZERO SUPPRESSION.

FILNAM - PRINT OUT A FILENAME

RS - ADD

CALL OUTCHR, RS

R2-RS RANDOM

CALL OUTSIZ

R2-R4 RANDOM

BISB

R2

MOV

..EVEN

WORD 1000, 100, 10

BYTE BYTE1, BYTE2

OUTCHR - PRINT ONE OR TWO CHARACTERS.

OUTSIZ - PRINT A NUMBER 0-255 WITH LEADING ZERO SUPPRESSION.

FILNAM - PRINT OUT A FILENAME

RS - ADD

CALL OUTCHR, RS

R2-RS RANDOM

CALL OUTSIZ

R2-R4 RANDOM

BISB

R2

MOV

..EVEN

WORD 1000, 100, 10

BYTE BYTE1, BYTE2

OUTCHR - PRINT ONE OR TWO CHARACTERS.

OUTSIZ - PRINT A NUMBER 0-255 WITH LEADING ZERO SUPPRESSION.

FILNAM - PRINT OUT A FILENAME

RS - ADD

CALL OUTCHR, RS

R2-RS RANDOM

CALL OUTSIZ

R2-R4 RANDOM

BISB

R2

MOV

..EVEN

WORD 1000, 100, 10

BYTE BYTE1, BYTE2

OUTCHR - PRINT ONE OR TWO CHARACTERS.

OUTSIZ - PRINT A NUMBER 0-255 WITH LEADING ZERO SUPPRESSION.

FILNAM - PRINT OUT A FILENAME

RS - ADD

CALL OUTCHR, RS

R2-RS RANDOM

CALL OUTSIZ

R2-R4 RANDOM

BISB

R2

MOV

..EVEN

WORD 1000, 100, 10

BYTE BYTE1, BYTE2

OUTCHR - PRINT ONE OR TWO CHARACTERS.

OUTSIZ - PRINT A NUMBER 0-255 WITH LEADING ZERO SUPPRESSION.

FILNAM - PRINT OUT A FILENAME

RS - ADD

CALL OUTCHR, RS

R2-RS RANDOM

CALL OUTSIZ

R2-R4 RANDOM

BISB

R2

MOV

..EVEN

WORD 1000, 100, 10

BYTE BYTE1, BYTE2

OUTCHR - PRINT ONE OR TWO CHARACTERS.

OUTSIZ - PRINT A NUMBER 0-255 WITH LEADING ZERO SUPPRESSION.

FILNAM - PRINT OUT A FILENAME

RS - ADD

CALL OUTCHR, RS

R2-RS RANDOM

CALL OUTSIZ

R2-R4 RANDOM

BISB

R2

MOV

..EVEN

WORD 1000, 100, 10

BYTE BYTE1, BYTE2

OUTCHR - PRINT ONE OR TWO CHARACTERS.

OUTSIZ - PRINT A NUMBER 0-255 WITH LEADING ZERO SUPPRESSION.

FILNAM - PRINT OUT A FILENAME

RS - ADD

CALL OUTCHR, RS

R2-RS RANDOM

CALL OUTSIZ

R2-R4 RANDOM

BISB

R2

MOV

..EVEN

WORD 1000, 100, 10

BYTE BYTE1, BYTE2

OUTCHR - PRINT ONE OR TWO CHARACTERS.

OUTSIZ - PRINT A NUMBER 0-255 WITH LEADING ZERO SUPPRESSION.
Disc Structure Notes
By Dave Mallery

GOALS
SURVIVAL
PERFORMANCE

SURVIVAL—MFD’S all equal
DCN listings for contiguous files

PERFORMANCE—Seek/Latency/Transfer
Therefore, primary goal is
REDUCE HEAD MOVEMENT

Sources of Head Movement:
Directories—if left unchecked, destroy performance
a) by randomizing their location
b) by randomizing their links.
Non-contiguous files—window turns.

A PROPER DISC
Fully extended, non-deletable, centered, contiguous UFD’s
Centered swapfiles (if applicable)
Major data files: centered, contiguous, non-deletable
Recoverable due to
1) copies of MFD contiguous
2) DCN listings of major contiguous files for NFS access.

HOW TO DO IT
1) During DSKINT
a) Pack Clustersize
b) SATT.SYS base
c) MFD Clustersize 8/16
d) do not pre-extend [0.1] & [1.2]
e) do not locate [1.2]
2) At this point, you have a disc with one cluster MFD at
   beginning and nothing else but [0.1] first cluster. SATT.SYS is
centered.
3) Pre-extend the MFD [1.1]
   —based on expected total accounts. Plan your accounts —
give yourself enough extra.
4) Pre-extend [0.1]—you don’t need much. Dump the SATT.
5) In [0.1], create enough contiguous files to fill up the
   SATT to the center.
6) Run SREACT, creating your accounts.

NOTE: You are making MFD entries only. Use UFO
clustersize large enough. Create accounts in order of importance.
7) Pre-extend each UFO.
8) Make an image copy(s).
9) Create and Protect your swap files.
10) You will want to:
a) make sys-call to zero an acct. priv or
b) Place a null length, non-deletable file in each directory. The
   full UFD will remain contiguous.
11) Copy in your main data files. Hopefully, they can be con-
tiguous and protected.
NOTE: they are at the top of the directory! As you need it,
balance their locations around the center by deleting some of the
bulk files in [0.1].
12) Copy in your BAC and BAS, etc. files.
Make as many contiguous as possible.
Finally all the rest.
As you copy, do the BAC.TSK files first, so they are higher up in
the directory.
In [1.2] place login at the top.
Use separate libraries for groups of cusps.
NOTE: B + 2 spoolers really pay off.

CARE AND FEEDING
1) Run SREORDER (in B + 2) frequently.
2) Don’t create accounts unless you must—have extras
   created and extended.
3) Keep a stash of contiguous space in [0.1].
4) Use contiguous files as much as possible.
5) Print out and keep the DCN’s of all contiguous major files you
   worry about.
6) Make all your discs from the image copy, so your MFD’s are
   identical and interchangeable.
BEGINNER'S GUIDE TO
MACRO 11 PROGRAMMING IN RSTS/E
by Thomas Courtney, Mark/Ops, Inc., for the SENERUG Reference Library

Chapter 1
Introduction

MACRO-11 is the assembly language for the PDP family of computers. Recently, programming in MACRO-11 has become respectable in the RSTS/E community, and this guide is intended to help you get started. You are going to need the following publications:

2. PDP11 Processor Handbook for your machine.
3. RSTS/E System Directives Guide.

Actually, everything other than program listings can be found in these manuals somewhere, so the adventurous may wish to proceed to them directly. This guide is not intended to make you a wizard at programming in MACRO-11; it is intended to give you a notion of the structure of the machine you work on, and a firm understanding of how things (like I/O) get done.

Chapter 2
The General RSTS/E Environment

To program in MACRO successfully requires a different "world view" from higher level language programming. Instead of thinking of data and structure and programs as different entities, all three merge into what is hopefully a well ordered heap. After all, how can you expect to do a GET, or a PRINT or a FIELD in assembly language? For if assembly language programming is meant to put the programmer in direct contact with the machine, we would need a pretty sophisticated machine indeed to have such statements as these.

The answer, of course, is that the PDP11 doesn't do all this. [See: "Technical Notes", p. 66] However, it does some things pretty closely, and only waits for the program to put data in the proper locations, and make a call to the Monitor. These are called General Monitor Directives, and are things like .READ, .WRITE, .CCL, etc. Their locations are defined in the file COMMON.MAC, which comes in all RSTS/E kits, and an abbreviated form is in Appendix A.

To RSTS/E, a MACRO job can occupy up to 17777 bytes of memory (all numbers are octal). The monitor directives get their data, and return some too, from the first thousand bytes of this memory. A mapping of this region looks like this:

<table>
<thead>
<tr>
<th>BYTE RANGE</th>
<th>NAME</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000-0057</td>
<td>KEY, USRSP</td>
<td>User job image or runtime system.</td>
</tr>
<tr>
<td>0060-0107</td>
<td>FIRQB</td>
<td>used by Monitor to make the job swappable.</td>
</tr>
<tr>
<td>0110-0167</td>
<td>XRB</td>
<td>used by Monitor to make the job swappable.</td>
</tr>
<tr>
<td>0170-0377</td>
<td>CORCMN</td>
<td>default stack</td>
</tr>
<tr>
<td>0400-0401</td>
<td>USRPPN</td>
<td>defines job status</td>
</tr>
<tr>
<td>0402-0441</td>
<td>USRPRT</td>
<td>File Request Queue Block</td>
</tr>
<tr>
<td>0442-0457</td>
<td>USRPRT</td>
<td>Transfer Control Block</td>
</tr>
<tr>
<td>0460-0657</td>
<td>USRPPN</td>
<td>core common area</td>
</tr>
<tr>
<td>0660-0733</td>
<td>USRPRT</td>
<td>controlled solely by the job.</td>
</tr>
<tr>
<td>0734-0735</td>
<td>USRPPN</td>
<td>user project, programmer number.</td>
</tr>
<tr>
<td>0736-0737</td>
<td>USRPRT</td>
<td>user default protection code.</td>
</tr>
<tr>
<td>0740-0777</td>
<td>USRPRT</td>
<td>user logical device name table.</td>
</tr>
</tbody>
</table>
MIS Professionals
Seeking A Fruitful Career Opportunity?

Consider Apple.

At Apple Computer, we sincerely believe that our product—the personal computer—is the key to providing the 21st century individual with the personal power to deal with an increasingly complex society. In the four short years since our inception, we've developed the world's leading personal computers. And we've certainly been fruitful in doing so. In fiscal 1980 we grew 300%, and the future looks even better.

Projected to reach sales of $5 billion by 1985, the personal computer industry promises unlimited growth, long-range stability and a wide variety of innovative challenges in a growing industry.

Systems Analysts
Programmer/Analysts

The career implications for MIS professionals at Apple are limitless. We're offering solid opportunities to motivated analysts and programmers who are eager to operate in a dynamic, high growth environment.

We have current openings at various levels, requiring one to five years experience and a Bachelor's degree or equivalent. We're especially interested in individuals familiar with DEC PDP-11/70, RSTS/E, and NCA software.

Your career can bear fruit with Apple. We pay top salaries and offer one of the best fringe benefit programs in the San Francisco Bay Area. Please send your resume to Professional Employment, Apple Computer, Inc., 10260 Bandley Drive, Cupertino, California 95014.

An Equal Opportunity Employer
The sections of this area we are most interested in are the FIRQB and the XRB, since these areas are used in most Monitor-job communications.

2.1 The FIRQB.

The FIRQB (File Request Queue Block) contains the information used by the Monitor for I/O requests. Before you do most I/O requests, you have to load the FIRQB area with a variety of values. The general form of the FIRQB is:

<table>
<thead>
<tr>
<th>OFFSET</th>
<th>NAME</th>
<th>HIGH BYTE</th>
<th>LOW BYTE</th>
<th>OFFSET</th>
<th>NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>FQFUN</td>
<td></td>
<td>returned status</td>
<td>0</td>
<td>FIRQB</td>
</tr>
<tr>
<td>3</td>
<td>FQSIZM</td>
<td>calfip/.uuo</td>
<td>job number * 2</td>
<td>2</td>
<td>FQJOB</td>
</tr>
<tr>
<td>5</td>
<td>project</td>
<td>MSB file size</td>
<td>channel number * 2</td>
<td>4</td>
<td>FQFIL,FQERNO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>file name (2 words in RADIX-50 form)</td>
<td>6</td>
<td>FQPPN</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>FQPROT</td>
<td></td>
<td>10</td>
<td>FQNAME1</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
<td>12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td>file ext. (1 word in RADIX-50 form)</td>
<td>14</td>
<td>FQEXT</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
<td>least significant bits of file size</td>
<td>16</td>
<td>FQSIZ</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td></td>
<td>buffer length</td>
<td>20</td>
<td>FQBULF,FQNAME2</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td></td>
<td>mode</td>
<td>22</td>
<td>FQMODE</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td></td>
<td>status flags</td>
<td>24</td>
<td>FQFLAG</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td></td>
<td>prot. code</td>
<td>26</td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td></td>
<td>device name (a ASCII characters)</td>
<td>30</td>
<td>FQDEV</td>
<td></td>
</tr>
<tr>
<td>33</td>
<td></td>
<td>0, unit real device unit number</td>
<td>32</td>
<td>FQDEVN</td>
<td></td>
</tr>
<tr>
<td>35</td>
<td></td>
<td>cluster size</td>
<td>34</td>
<td>FQCLUS</td>
<td></td>
</tr>
<tr>
<td>37</td>
<td></td>
<td>number of entries in directory lookup</td>
<td>36</td>
<td>FQNENT</td>
<td></td>
</tr>
</tbody>
</table>

Most of these entries should be self explanatory. The major idea is that the programmer puts in information needed by whatever Monitor directive, makes the call, and retrieves whatever information the Monitor gave back.

2.2 The XRB.

The XRB is the other major area used for Monitor-job communications. Information specified here are things like the location and size of strings to be processed. The general format of the XRB is:

<table>
<thead>
<tr>
<th>OFFSET</th>
<th>NAME</th>
<th>HIGH BYTE</th>
<th>LOW BYTE</th>
<th>OFFSET</th>
<th>NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>XRBLKM</td>
<td>buffer size in bytes</td>
<td>0</td>
<td>XRLEN</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>bytes transferred</td>
<td>2</td>
<td>XRC</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>buffer address</td>
<td>4</td>
<td>XRL</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>MSB of block #</td>
<td>6</td>
<td>XRCI</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td>LSB of block #</td>
<td>10</td>
<td>XRBLK</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td>wait time for terminals</td>
<td>12</td>
<td>XRTIME</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td>device modifier</td>
<td>14</td>
<td>XRMOD,XRBSIZ</td>
<td></td>
</tr>
</tbody>
</table>

Again, almost everything is straightforward. XRTIME is set for terminal input only. If it is 0, the system will wait forever. If it is positive, that is the time it will wait for a response. If negative, it goes into "keyboard Monitor wait" state, which I have never used, but the Directives manual says is used for indirect command files by the Monitor.

Chapter 3

MACRO-11 Programming Basics

MACRO-11 is a fairly complex assembly language, with over 400 instructions in the instruction set, eight register modes, and a variety of assembly time instructions. I am not about to explain all of it, quite
simply because I do not know all of it. Instead, I will attempt to explain the small subset of instructions I am going to use.

A statement in MACRO-11 has the form

```
label: operator operand, operand ; comment
```

the label and comment are not necessary to the legality of the statement. Some operators require one or no operands. The types of assembly statements fall into the following general categories:

1. Instructions: these are from the PDP11 instruction set.
2. Directives: these are calls to the RSTS/E Monitor
3. Data formatting: these reserve space for data.
4. Output formatting: these control listing results.
5. Logical control: these instructions allow pieces of a program to be assembled independently.

I will admit there are statements that do not fall into any of these categories. However, I will try not to use any of them here, and firmly believe that for the first shot, we will have enough to chew on.

### 3.1 The Instruction Set.

These are the nitty-gritty machine instructions. As they say in the processor handbook of your choice, instructions come in three flavors:

1. single operand — these are the operator operand instructions. The ones we will be using are:

   **MNEMONIC**   **ACTION**

   - CLR dst clears dst (destination) word
   - CLRB dst clears dst byte.
   - TSTB tests dst.

   Basically, CLR and CLRB put a 0 in a location specified by the operand. TSTB (and oddly enough, TST) test a location and set the Processor Status word. The Processor Status word contains information on the current status of the machine. Instructions operate on bits 0-3 of this word, called C, V, Z and N respectively. The C bit is set whenever an instruction caused a carry out of the most significant bit of the result. The V bit indicates an operation overflow. The Z bit is set whenever the result of an instruction is cleared. The N bit is set whenever the result of an operation is negative. In general, if the condition does not occur, then the appropriate bits are cleared.

2. double operand — these are the operator operand operand instructions. The ones we will use are:

   **MNEMONIC**   **ACTION**

   - MOV src, dst move source to dst.
   - SUB src, dst subtract src from dst.

3. program control — specifies specific actions to be taken by the processor. The ones we will use are:

   **MNEMONIC**   **ACTION**

   - BNE dst branch to dst if Z bit clear.
   - JSR R, nam jump to subroutine nam
   - RTS R return from subroutine.

JSR and RTS also use a register, called the LINKAGE REGISTER. Upon executing the subroutine call, the
current contents of the register are pushed onto the stack, and the return address is put into the register. Upon returning from the subroutine, the contents of the register are used for the return address, and its original contents are restored and popped off the stack.

3.2 Directives.

As you now already know, the Monitor Directives interface between the Monitor and your job. We are only going to use five in the following examples: .READ, .WRITE, .FSS, CALFIP and .EXIT.

1. .READ — this directive reads data from a file or a device previously opened on a channel. Since we’re going to use the keyboard, there will be no need to open it, since it defaults to being channel 0. We will have to fill in the information on the XRLEN, XRBC, XRLOC, XRCI and XRBLK. We will get information back on .XRBLK.

2. .WRITE — This directive writes to a file from a user buffer specified in the XRB. The data we must pass it are XRLEN, XRBC, XRLOC, XRCI and XRBLK. I’m not going to talk about what can be returned, since they are errors, and by the time you are ready to deal with them, you can look them up without much hassle.

3. .FSS — the File String Scan directive. This allows you to set up the FIRQB automatically when opening a file. You can also use it to convert strings to RAD50 format.

4. CALFIP — this command does all sorts of I/O related functions, like opening or closing files, assigning/deassigning devices, directory lookup and quite a bit more. We will use it to create a file, as a demonstration of how to set up the FIRQB using the .FSS directive.

5. .EXIT — This returns control to the default runtime system. You need pass nothing to the call, and nothing is returned.

3.3 Data Formatting.

These statements reserve space for data, and sometimes put data into specific locations. The ones we will use are:

<table>
<thead>
<tr>
<th>MNEMONIC</th>
<th>ACTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>.BLKW 1000</td>
<td>reserves 1000 (octal) words of storage.</td>
</tr>
<tr>
<td>.ASCII /Test/</td>
<td>puts the ASCII characters T, e, s, t and &lt;cr&gt;, If into storage.</td>
</tr>
<tr>
<td>.ASCIZ /Test/</td>
<td>same as .ASCII but appends a null character to the end.</td>
</tr>
<tr>
<td>.EVEN</td>
<td>puts in an extra byte if program counter is odd.</td>
</tr>
</tbody>
</table>

3.4 Output Formatting.

These statements take care of how the assembly listing looks.

<table>
<thead>
<tr>
<th>MNEMONIC</th>
<th>ACTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>.TITLE Test</td>
<td>causes the title “Test” to appear on every output page of the assembly listing.</td>
</tr>
<tr>
<td>.IDENT /v07.1/</td>
<td>causes a version number to be added to the title.</td>
</tr>
<tr>
<td>.PAGE</td>
<td>causes a page eject in the assembly listing.</td>
</tr>
<tr>
<td>.END nam</td>
<td>end of source file input. If “nam” is not null, then this is the entry point procedure. There can be only one entry point procedure built at task building time.</td>
</tr>
</tbody>
</table>
Your Second Source To DEC.*

IMMEDIATE DELIVERY
11/70, 11/34, 11/23, LSI

We Provide Systems, Add-ons, Installations and Maintenance
Call and ask us about our DEC equivalent systems for:

LSI, PDP11/34, and 11/70
RM02, RM03, RM04
RP02, RP03, RP06, RP07
TE-10, TE-16, TU-45

(408) 732-4523

West Coast Computer Exchange, Inc.

*DEC is a registered trademark of Digital Equipment Corporation

248 Sobrante Way, Sunnyvale, CA 94086
3.5 Logical Control.

These instructions allow the deferring of several actions until link time. We will only use one such statement:

```
.GLOBL NAM1, NAM2
```

causes NAM1 and NAM2 to be external global references. A label is declared GLOBL by using a double semicolon, e.g. label::. A mnemonic variable is made GLOBL by using a double equal sign, e.g. XRLEN = = 0.

Chapter 4

Registers and Addressing Modes.

The PDP11 has eight general purpose registers for use in memory reference addressing. These registers can serve a variety of functions, from holding data to be played with, to containing pointers to pointers to addresses. Two of the general registers, R6 and R7 (the others are R0—R5) have special functions. Register 6, called SP, is the hardware stack pointer. Register 7 contains the location of the next address to be executed, and is called PC or "". The secret of how all the registers are used is in their addressing modes.

There are eight addressing modes, and these are all explained in the Processor Handbook. Even though they do it better than I, here is a synopsis of what the story is:

1. Mode 0, Register Mode. This uses the register as a simple accumulator. It is the fastest way to do things, whenever possible.

   INC R3 ; adds 1 to the contents of R3.

2. Mode 1, Register Deferred Mode. The address of the operand is stored in the register.

   INC (R3) ; adds 1 to the contents of the address specified by R3.

3. Mode 2, Autoincrement Mode. Like Mode 1, but the register is incremented afterwards. It is incremented by 1 if you are using a byte instruction, and by 2 if you are using a word instruction.

   INC (R3) + ; adds 1 to the contents of the address specified by R3, then increments R3 by 2.

4. Mode 3, Autoincrement Deferred Mode. The contents of the address specified by the contents of the register are used as the address of the operand. The register is then incremented 1 or 2, for byte or word instructions respectively. The only way I can remember all of this one is to call it "pointer to a pointer" mode.

   INC @(R3) + ; adds 1 to the contents of the address specified by the contents of the address specified by R3, then increments R3 by 2.

5. Mode 4, Autodecrement Mode. This is the same as Autoincrement Mode, except it decrements the register before using its contents.

   INC -(R3) ; contents of R3 are decremented by 2.

   ; the address specified by this number are then incremented by 1.

6. Mode 5, Autodecrement Deferred Mode. Similar to Autoincrement Deferred, but the register is decremented before anything else happens.
INCM @-(R3) ; R3 is decremented by 2. Adds 1 to the contents of the address specified by R3.

7. Mode 6, Index Mode. The base address is added to the index word to get the address of the operand.

INC 1000(R3) ; 1000 is added to R3, and the contents of that address are incremented by 1.

8. Mode 7, Index Deferred Mode. The base address is added to the index. The contents of the index is then used as a pointer to the address of the operand.

INC @1000(R3) ; 1000 is added to R3; and the address pointed to by the contents of the address in R3 are incremented by 1.

There are also 4 modes associated with the use of the PC register.

9. Mode 2, PC Immediate Mode. This is equivalent to autoincrement with the PC.

ADD #10, R0 ; the value 10 is added to R0.

10. Mode 3, PC Absolute Mode. This is equivalent to autoincrement deferred mode with the PC.

ADD @#10, R0 ; adds the contents of location 10 to R0.

11. Mode 6, PC Relative Mode. This is the Index mode with the PC.

ADD MSG, R0 ; adds the contents of MSG to R0.

12. Mode 7, PC Relative Mode Deferred. This the Index deferred mode with the PC.

ADD @MSG, R0 ; the contents of the address specified by the contents of MSG are added to R0.

... continued on page 88
SYSTEM FILES

If the installation of QUE.11 has been successful you will have these files in your system ([1,x] is the QUE.11 account):

Queue control package: (The protection codes shown must be used)

- [1,x]QUE11.TSK <124>
- [1,x]OP.TSK <124>
- [1,x]TEST.TSK <104>
- [1,x]LOAD.TSK <124>
- [1,x]SHOW.TSK <104>
- [1,x]SUBMIT.TSK <104>
- [1,x]DO.TSK <104>
- [1,x]CANCEL.TSK <104>

Spooling package:

- $PRINT.TSK <104>
- $SPOOL.TSK <232>
- $QUE.TSK <232>

Print control files: (required for each print device)

- $NORMAL.LP <40>
- $NORMAL.LPO <40>

In some systems the PRINT, SPOOL and QUE programs are held in the [1,x] library instead of the $ account. This requires a minor alteration to the control files NORMAL.LP etc. Note that these control files must be kept in the $ library.

SYSTEM STARTUP

The only job which must be started is QUE11.TSK. This may be run in any privileged account but the old (if any) QUE11.DAT file should be in the same account.

When QUE.11 starts it prints a ?-mark. This can be answered in two ways:

1. device specification
2. detach statement (always last).

Device Specification

QUE.11 has an internal table with a list of all the physical devices configured in your system. This table is used to schedule queued jobs which reserve devices and associated volumes. You may disallow the use of a class of devices by a command of this type:

`? MT:/bar`

You may add to the list of devices by giving the mnemonic of a new 'dummy' device, like this:

`? CLASS:
Dummy devices are a very powerful aid in job scheduling; they are described in detail later.

Switches: /single /bar /clock

Detach Statement

This statement ends the initialization. After it is given QUE.11 searches for a file called QUE11.DAT which holds a list of any previously queued jobs. If the file does not exist in the account under which QUE.11 is running then a new one is set up automatically.

Two switches may be used on the DETACH statement these are /LIST which prints the internal device table and /PRi:n which sets the QUE.11 priority.

These two switches must be given in the order shown here:

`?DETACH/LIST/PRi:0`

Note that if you defined a dummy device at a previous set-up which is now omitted, an error may occur when the QUE11.DAT file is scanned. Any job which reserved the omitted device will be omitted; a message will be printed to warn you.

CONCISE COMMAND LANGUAGE (CCLS)

You should add this set of commands to the Concise Command Language (remember to alter the START.CTL files for INIT):

```
CCL SHO=W = [1,x]SHOW.TSK
CCL SUB-MIT = [1,x]SUBMIT.TSK
```
OUTLINE OF OPERATION

QUE.11 processes requests by using pseudo-keyboards as the job consoles. It can process up to four batch streams at once if there are four PK:s free in the system.

Job scheduling is done by testing each job against the following list of requirements which must be satisfied before the job will start:

1. has the system room for another job?
2. has QUE.11 a job slot free (10 available)?
3. is there a pseudo-keyboard free?
4. has QUE.11 enough free I/O channels (1 for immediate mode, 2 for batch or 3 for batch with a log file)?
5. are the reserved devices free (not assigned and not opened)?
6. are the required volumes (paper) loaded?

CLOSE DOWN

QUE.11 is ended by giving the OP CLOSE command. This serves two purposes it shuts down the queue only when all the current jobs have finished and it saves outstanding job requests in the QUE.11.DAT file. If QUE.11.DAT is stopped in any other way (by a KILL command, for instance) it is likely that some items in the queue will be lost.

The SHUTUP program will not end QUE.11 properly; the OP CLOSE command should be given just before SHUTUP is started.

The SYSTAT job table should be checked to make sure that QUE.11 has stopped. A typical closedown might look like this:

```
SYS/2
2 [1.5] Det QUE.11 19/28K SL D02...
Ready
OP CLOSE
Ready
SYS/2
Ready
Run $SHUTUP
```

Announcing:

QUE-11

ONE JOB SPOOLER FOR RSTS/E CONTROLS ALL SPOOLING

REPLACES:

- SPLIDL, SPLRUN ... etc.
- BAT IDL, BAT RUN ... etc.
- QUEMAN, QUMRUN ... etc.
- OPSER, OPSRUN ... etc.
- ATPRO
- ATPK

QUE-11:

- One job controls all spooling
- Saves small buffers and job slots
- Spawns jobs as needed
- Handles line printer and keyboard spooling
- Controls as many BATCH JOBS as pseudo-keyboards
- Full parameter replacement in QUE calls
- "DO" command replaces indirect processors
- Manuals available now
- Program deliveries early in 1981
- Only $995 single cpu license
- Trial Version $100

For more information contact:
On Track Systems, Inc., P.O. Box 245, Ambler, PA 19002
Phone: 215/542-7008

STATE CODES

The state column in the QUE.11 status report (SHOW command) uses the codes described here to indicate the reason why a job is waiting to run. The normal codes that may appear are:

- Q Wait
  request not yet scanned because some essential resource is unavailable.
- V Wait
  job requires one or more volumes to be mounted
- D Wait
  job requires a device that is in use; it will run when the device becomes free.

Abnormal codes are:
Exxx
an unrecoverable error occurred in starting the
job (xxx is the RSTS/e error code).

The operator should cancel requests which appear
with the abnormal codes because they prevent later re­
quests from running.

DEVICE TABLE

As explained already each job request may reserve up
to five devices and associated volumes. These may be any
of the physical or dummy devices. The job will not be
started until all the devices are free and until they each
have the required volume loaded on them.

If a job reserves a device without a specific unit
number (e.g. MT:) the job will run as soon as any device of
the type required becomes available.

The list of currently loaded volumes is altered by the
load command which has this form:

    LOAD dev:volume

e.g
    LOAD LPO:WIDE
    LOAD MT2:TAPE79

No checking (except for discs) is done to see if the
volume is actually the one loaded and it is the responsibility
of the operator to ensure that the QUE.11 device list ac­
curately reflects the state of the peripheral devices on the
system.

The current status of the device list may be found by
typing

    SHOW D[evices]
or
    SHOW V[olumes]

Only the devices which are currently loaded will ap­
pear.

DUMMY DEVICES

Dummy devices are treated by QUE.11 in almost ex­
actly the same way as the real devices on the system. For
example, suppose that CLASS: was set up as a dummy
device. It could be used to schedule large, slow jobs to run
at night or at some other convenient time. The request to
QUE.11 would be:

    SUBMIT test for CLASS:NIGHT

which might show in the status report as:

<table>
<thead>
<tr>
<th>Queue index</th>
<th>Owner</th>
<th>State</th>
<th>Reserved Devices</th>
</tr>
</thead>
</table>

When the operator was leaving in the evening the
device CLASS: would be loaded with the 'volume' night:

    LOAD CLASS:NIGHT

Remote job entry is an application where a real device
may be loaded with a dummy volume. The distant receiver
might be given a mnemonic name which would be used
when submitting jobs for transmission. These jobs would
be released by loading the device RJ: with the mnemonic
name.

The main distinction between dummy devices and the
real ones is that the dummy devices may be used by more
than one job at the same time. Where this is undesirable it
can be prevented by appending the /SINGLE switch to the
device when QUE.11 is being started.

SPOOLING OF PRINT FILES

Print files are queued to the printer(s) by a package of
programs (stored in the 'S' account) which make use of the
QUE.11 batch control facilities. The files are transferred to
the printer by a program called SPOOL.TSK which is run
under the control of a command file (also stored in the 'S'
account). The name of this file is of special significance; it
represents the type of paper required while the file name
extension indicates the printer device.

This filename convention is important; it the method
by which the System Manager controls which devices may
be used as queuable printers and the forms may be used on
them.
For example, if your system has a printer LPO: and a
terminal KB3: which is also to be used as a printer, you
might provide these files:

\[\begin{align*}
\$\text{NORMAL.LP} \\
\$\text{NARROW.LP} \\
\$\text{NORMAL.LPO} \\
\$\text{NARROW.LPO} \\
\$\text{NARROW.KB3} \\
\$\text{FORMX.KB3}
\end{align*}\]

These files would allow normal and narrow paper to
be used in the printer LPO: and narrow and FormX in KB3:.
Notice that the file name extensions were given in two
forms for LP: — this allows the user to specify the printer
exactly (LPO:) or by the generic name (LP:) if the unit is
unimportant. This does not matter if there is only one
printer on your system. On a two printer system the .LP
extension would allow files to be queued for either LPO: or
LP1: (determined by QUE.11 when the job is scheduled).

When a file is queued for printing with the PRINT or
QUE programs a request is submitted to QUE.11 in the
same way as a normal batch job request. However the con­
trol file is one of the special files described above. The
printer and paper are reserved automatically and appear in
the QUE.11 status report as for a batch job.

The name of the file(s) to be printed are given as a
list of ‘parameters’ which cause some of the words in
the command file to be replaced. The three words af­
tected are:

1. #FS — the filename string (which may contain
switches)
2. #SK — replaced by Y if page skip is required;
3. #RQ — used for REQUED information if
necessary.

SHOW n may be used to see the parameters for job
'n'.

PRINT COMMAND FILES

The $\text{NORMAL.LP} file which is supplied should be
used as a basis for new control files. The listing below
gives the purpose of the lines in the control file.

\[\begin{align*}
1 & \text{RUN } \$\text{SPOOL.TSK} \\
2 & \text{#FS} \\
3 & \text{&VOL} \\
4 & \text{133} \\
5 & \text{66} \\
6 & \text{8192} \\
7 & \text{#SK} \\
8 & \text{&PPN} \\
9 & \text{#RQ}
\end{align*}\]

The following items are all substituted when the file
is read by QUE.11 at run time:

\[\begin{align*}
\text{FS}, \&\text{VOL}, \text{#SK}, \&\text{PPN}, \text{#RQ}
\end{align*}\]

At line 7 a file name may be given; the contents of
this file will be printed as part of a separator page at the
begining of the printout. If this line is left blank no
separator page appears.

At line 6 the value 8192 stops a form-feed when the
printer is closed at the end of the job. See the RSTS/E
programmers guide for information on other values.

If the operator cancels a job that has already started
the job is stopped and the paper is realigned. SPOOL
assumes that the printer can handle form-feeds cor­
correctly.

SPOOL COMMANDS

The operator can control printing by means of these
commands:

1. CANCEL n
CANCEL dev: (e.g CANCEL LP:)
Cancel the job before or during printing. The
second form may only be used when the job has
started to print.

2. LOAD dev:paper
used to tell QUE.11 that the paper has been
changed. If the paper is changed while a job is
actually running there is no effect until it finishes and the next job is selected.

3. OP STATUS n
   OP STATUS LP:
   used to discover the state of the current job. This command and the following ones will only work while the specified job is actually using the printer, otherwise one of the errors
   ?Job not in receive state
   ?Device not controlled by QUE.11
   will occur.

4. OP RESTART n B
   OP RESTART dev: B
   May be used to restart a printout at the beginning (omit B) or at a specified block. This may upset the page numbering if the /SKIP option was chosen.

The SPOOL.TSK program waits when the print buffers are full. This can cause a delay when some of the commands above are given before a response is returned to the operator and action is taken by the program. The most affected commands are the OP commands.

---

**DATA ENTRY/INQUIRY POWER**

**MULTI-USER DATA ENTRY**
- Create, modify or verify data files
- Extensive field editing
- User friendly design
- Extracts ISAM file data

**POWERFUL FEATURES**
- Dictionary-driven
- Full logic support
- CRT features supported (blink, bold, underline, reverse video & cursor)

**ON-LINE INQUIRY**
- Inquiry and/or update for ISAM files
- Multiple ISAM files
- User access control
- Browse feature

**DEC INTERFACE**
- RSTS/E, RSX11M, RT11
- RMS11K with secondary index support
- Optional support for resident libraries

**RESULTS**
- Increased productivity
- Macro-coded - FAST!!!
- Single DEUS program replaces user data entry inquiry programs

**STRATEGIC DATA SYSTEMS**
4N310 Knoll Creek Drive
St. Charles, IL 60174
(312) 377-3396

---

From $750
PDP II USERS

WANT AN ALTERNATIVE?
YOU CAN HAVE ONE

with computer maintenance

from Tymshare®

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

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

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

I would like to be contacted by Tymshare to discuss my maintenance needs in more detail.

I would like a price quote on the cost to maintain my computer system.

Type of maintenance coverage you desire (CHECK ONE)
- 8 am to 5 pm Monday through Friday
- 8 am to 8 pm Monday through Friday
- 24 hour Monday through Friday
- 24 hour, 7 days per week
- Per Call Only
- Other — Please indicate below:

MY PRESENT CONTRACT EXPIRES

EQUIPMENT TYPE QUANTITY EQUIPMENT TYPE QUANTITY

PHONE NO. ___________
**ON20FF.B2S**

By Dave Schott

---

**Program:** ON20FF.B2S

**Version:** 7.0

**Edit:** 07

**Edit Date:** 10-Nov-80

**Author:** Dave Schott

---

This software is furnished free of charge to RSTS/E sites. It may be distributed and copied to any site as long as there is no direct commercial profit derived from such distribution.

Although this software has been tested and is believed to be error-free, neither ACS! nor the author assume responsibility for the use or reliability of the software.

Any problem with the software should be reported to the author at the address above, although support of the software is not guaranteed.

---

**MODIFICATION HISTORY**

---

**GENERAL DESCRIPTION**

ON20FF will turn On/Off the modem bit on the requested keyboard line. This is accomplished by re-writing the TTINTF (terminal interface flag) of the requested keyboard line. Since this program does poke memory it can only be run successfully from [1,1].

---

**MAIN CODING**

---

**ERROR GOTO 19000**

1 establish the standard error handling

\> PRINT IF CCPOS(04) <> 0
\> return KB to left margin
\> IDS = "07,07"
\> set up the version ID tag

2010 PRINT "ON20FF*CHAR(93)+ID$+CHAR(94)+CVTS(RIGHT(SYS(CHAR(64)+CHAR(93)+CHAR(94)+CHAR(08)+CHAR(31),4),4)+"
\> indent yourself w/version ID tag

2020 DIM MONITOR1(30%),MONITOR2(30%)
\> CHANGE SYS(CHAR(64)+CHAR(31)) TO MONITOR1
\> first part of the monitor table
\> MONITOR1(I%) = MONITOR1(I%)+SWAP(MONITOR1(I%+I%))
\> FOR 1% = 1% TO 29% STEP 2%
\> swap those bytes that need to be
\> CHANGE SYS(CHAR(64)+CHAR(31)) TO MONITOR2%
\> second part of the monitor tables
\> MONITOR2(3%(I%+I%)) = MONITOR2(3%(I%+I%)) + SWAP(MONITOR2(3%(I%+I%)))
\> FOR I% = 1% TO 29% STEP 2%
\> swap here also

3030 MAX. KB% = MONITOR1(31)
\> maximum # of keyboard on this system
\> DEVKBR = MONITOR1(73)
\> KB.DDB = POKE(DEVKBR+MONITOR1(73)
\> KB.DDB = POKE(DEVKBR+DEVKBR)
\> KB0.DDB = PEEK(MONITOR1(73))
\> KB0.DDB = PEEK(DEVKBR+DEVKBR)
\> KB0.DDB = POKE(DEVKBR+DEVKBR)

4040 PRINT "Keyboard #: ":
\> PRINT a keyboard line
\> KB$ = PRS(UNLESS ERRFLG)
\> GOTO 10600 UNLESS ERRFLG
\> GOTO 32767

1050 ERR$ = VAL(KB$) UNLESS ERRFLG
\> get the keyboard into correct form
\> GOTO 10650 UNLESS ERRFLG
\> PRINT ERR$ (ERRFLG)
\> let the user know of an error

---

March 1981

---

**RSTS/E Professional User's Conference**

---

**Able Communications Systems, Inc.**

1197 North Tustin Avenue

Anaheim, CA 92807

(714) 632-0202

---

**NEW TTINTF = TTINTF OR 16384%**

---

**ASCII(CVT$(I$)) = 76%**

---

**ERR$ = ERR$ OR ERR$**

---

**ERR$ = ERR$**

---

**ERR$ = 0%**

---

**ERR$ = 0%**

---

**ERR$ = 0%**

---

**ERR$ = 0%**

---

**ERR$ = 0%**
FICHE.BAS

By Scott Banks and Dave Mallery
Fully compatible systems and peripherals

<table>
<thead>
<tr>
<th>Model</th>
<th>Capacity (MB)</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>NRM05</td>
<td>300</td>
<td>$20,500</td>
</tr>
<tr>
<td>NRJ05</td>
<td>300</td>
<td>$18,800</td>
</tr>
<tr>
<td>NRM06</td>
<td>200</td>
<td>$13,600</td>
</tr>
<tr>
<td>NRJ06</td>
<td>200</td>
<td>$19,600</td>
</tr>
<tr>
<td>NRP06</td>
<td>200</td>
<td>$17,900</td>
</tr>
<tr>
<td>NRM03</td>
<td>80</td>
<td>$12,600</td>
</tr>
<tr>
<td>NRJ03</td>
<td>80</td>
<td>$14,800</td>
</tr>
<tr>
<td>NRM03</td>
<td>80</td>
<td>$13,100</td>
</tr>
<tr>
<td>NRP03</td>
<td>80</td>
<td>$7,900</td>
</tr>
<tr>
<td>NTJ45</td>
<td>45</td>
<td>$7,300</td>
</tr>
<tr>
<td>NTJ45</td>
<td>75</td>
<td>$8,600</td>
</tr>
<tr>
<td>NT125</td>
<td>125</td>
<td>$9,800</td>
</tr>
</tbody>
</table>

For complete details, contact us at:

Plycom services, inc.
P.O. Box 160
Plymouth, IN 46563
(219) 935-5121

From one of the pioneers in commercial data processing using RSTS. Off the shelf software ready for immediate delivery. Completely interactive. Extensively documented. Fully supported. Ideal for OEM's, service bureaus or end users. Cost effective solutions including:

- Accounts Payable
- General Ledger
- Financial Reporting
- Accounts Receivable
- Payroll
- Fixed Assets

For more information, contact Plycom.
RSTS/E MONITOR INTERNALS
PART 1

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

For many years RSTS users have been pleading with DEC to produce a manual describing the internal workings of the RSTS/E monitor. However, DEC is in a difficult position. If they published such a manual some people would complain if DEC later made something in the monitor incompatible with the current version.

This is where we in the user community come in. I'm willing to share some of the information I've gathered in my work with RSTS over the past eight years. I hope this information proves useful.

This is the first of four articles that will describe the internals of RSTS/E V7.0. All the major functions will be described, including job control, memory control and file and device handling.

The articles will be broken into four parts as follows:

Part 1: 1. Job Control
   - JOBTBL—Job table
   - JDB—Job data block one
   - JDB2—Job data block two
   - IOB—I/O data block
   - WRK—Work block
   - Fixed memory locations
   - JBSTAT, JBWAIT—Job status tables

Part 2: 2. Memory Control
   - MCB—Memory control sub-block
   - MEMLIST, RESLIST—Memory control lists
   - RTS—Runtime system descriptor block
   - NULRTS—Disappearing RSX RTS
   - RTSLST—RTS control list
   - LIB—Resident library descriptor block
   - LIBLST—Library control list
   - WDB—Window descriptor block

Part 3: 3. File Control
   - SCB—Small file control block
   - FCB—Large file control block
   - FCBLST—File control list
   - WCB—Window control block

Part 4: 5. Send-Receive
   - RIB—Receiver ID block
   - ERLRIB—ERRLOG RIB block
   - SNDLST—Receiver control list
   - PMB—Pending message block

Part 5: 6. Concise Command Language
   - CCL—CCL definition block
   - CCLLST—CCL control list

Part 6: 7. Miscellaneous

Since the information described in these articles is intended primarily for systems programmers it will be presented in the form of a reference manual. My apologies to those of you who are just looking for a general understanding of RSTS and come away bleary-eyed.

If you plan to use this information to access monitor table information from an application program please remember that DEC may change this information at any time, so write your programs accordingly. They've changed things in every release so far. There's no reason to think they won't do it again.

Enough said. Let's get inside RSTS.

1.0 JOB CONTROL

RSTS can support up to 63 simultaneous jobs. Each of these jobs can be either a user at a terminal or a detached program. The job control structures allow RSTS to share resources (such as CPU time) properly among all users. In addition they provide the means to access the information necessary for almost every other service provided by the monitor, such as device and file handling.

For example, the scheduler uses the information in the job descriptor block to determine which job to run next and what run burst to assign it. The memory manager uses the memory control sub-block and the residency quantum to set up the memory mapping registers and perform swapping, if necessary.

The job control structures consist of a combination of tables and blocks. The size of the tables is determined by the number of jobs specified at sysgen time. The size of a block is typically 16 words.

The tables contain one word for each possible job on the system. As new jobs are created and removed, infor-
motion in the tables is changed but the size of the tables remain the same.

The job control blocks, on the other hand, exist only while a job exists. They are created (from small buffers) when a job is first created and deleted when the job is removed (i.e., by logging out).

The location of specific tables can be determined using the GET MONITOR TABLES SYS calls. Once you know where a table starts, the values within the table can be accessed by adding the required offset to the starting address of the table. The following example gets the address of the Job Data Block (JDB) for job 5 (see section 1.1 and 1.2 for information about JOBTBL and JDB):

1.1 JOBTBL—JOB TABLE

The Job Table is the root of the job control structures. It points to the Job Data Block which, in turn, points to the other job related blocks.

JOBTBL contains one entry for each possible job on the system. The job control information for each job can be accessed by using the job number times two as an offset from the beginning of JOBTBL. The value found at that location will be the address of the Job Data Block (JDB) for that job. If a 0 is found at that location there is currently no job with that job number.

The first word of JOBTBL (at offset 0) is always a 0. This word corresponds to the entry for the null job, job 0. The last word in the table will always be -1 to signify the end of the table. Thus, the total length of JOBTBL, in words, is JOBMAX (the maximum number of jobs, specified at sysgen time) plus 2.

1.2 JDB—JOB DATA BLOCK ONE

The Job Data Block (JDB) contains the most commonly used information about a job. It is pointed to by the entry in JOBTBL corresponding to its job number. The JDB, in turn, points to the three other job control blocks for the job: JDB2, IOB and WRK.

The scheduler uses JDPRI and JDBRST to determine which job to run next. The memory manager uses JDMCTL, JDSIZE, JDSIZM and JDSIZN to set up the memory mapping registers and to schedule a swap-in. The swapper uses JDRESQ, JDSWAP, JDMCTL and JDSIZE to swap jobs in and out of memory. The EMT handler uses JDIOST, JDILG and JDWRK to process system and I/O calls.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>JDIOB</td>
<td>This word contains the address of the I/O data block (IOB) for this job (see section 1.4).</td>
</tr>
<tr>
<td>2</td>
<td>JDFLG</td>
<td>This word contains the main job status flags (see section 1.2.1).</td>
</tr>
<tr>
<td>4</td>
<td>JDIOST</td>
<td>This byte contains the error code to be returned to the program after the completion of the current monitor call. If the bit JFIOST is set in the job status flags (JDFLG), JDIOST is moved to the location FIRQB+0 in the user program area (accessed by the variable ERR in BASIC-PLUS). A value of 0 indicates that no error occurred. (See the System Directives Manual).</td>
</tr>
<tr>
<td>5</td>
<td>JDPOST</td>
<td>If the bit JFPOST is set in the job status flags (JDILG) this byte is used to specify which information in the job's work block (WRK) is to be posted to the job's FIRQB or XRB. If the value in JDPOST is positive it is used as an index into a table of bit masks for posting to the FIRQB. If it is negative the low order 7 bits are used as a bit mask for posting to the XRB.</td>
</tr>
<tr>
<td>6</td>
<td>JDWORK</td>
<td>This word contains the address of the work block (WRK) for this job (see section 1.5).</td>
</tr>
<tr>
<td>8</td>
<td>JDB2</td>
<td>This word contains the address of the second job data block (JDB2) (see section 1.3).</td>
</tr>
<tr>
<td>10</td>
<td>JDILG</td>
<td>This byte contains the secondary job status flags (see section 1.2.2).</td>
</tr>
<tr>
<td>11</td>
<td>JDSIZN</td>
<td>This byte contains the size (in K words) to make this job the next time it is swapped in. The memory manager uses this location when a job must be swapped out to find additional memory as it attempts to grow in size.</td>
</tr>
</tbody>
</table>
12 JDRTS  This word contains the address of the runtime system descriptor block (RTS) associated with this job. (RTS blocks will be discussed in the second part of this series.) Every job has a runtime system associated with it at all times. If the disappearing RSX runtime system is associated with this job, JDRTS will contain the address of the null runtime system descriptor block (NULRTS).

14 JDRESQ  This word contains the current residency quantum for the job. As long as the job’s residency quantum is non-zero the job is not eligible to be swapped out. This is used to reduce memory thrashing. When a job is first swapped into memory it is given a residency quantum. Each time the job runs or does disk I/O the residency quantum is reduced in proportion to the amount of time it ran and the estimated time to complete the I/O. When the job goes into terminal input wait the residency quantum is set to zero to allow the job to be swapped if necessary.

16 JDMCTL  These five words are the memory control sub-block (MCB) for the job. The memory control sub-block will be discussed in the second part of this series.

20 JDSIZE  This word (within the memory control sub-block) contains the size of job in K words.

26 JDRESB  This word contains the bit mask to be posted to the level three queue word (L3Q) when the job is made resident. It is used by the monitor to notify itself when the job is made resident so that a function that required the job to be resident may be continued.

28 JDPRI  This byte contains the job’s priority. It can range from -128 to +127. The scheduler uses this byte to determine which job to run next.

29 JDBRST  This byte contains the job’s run burst. The run burst is the amount of time (in clock ticks) the job may execute compute-bound before the scheduler is called.

30 JDSIZM  This byte contains the job’s private memory size maximum. A value of 255 indicates that the job may use up to the system job size maximum.

31 JDSWAP  If this byte is non-zero it contains the slot number within the swapping files that the job is swapped out to. If it is zero, the job is either resident in memory or has no job image.

## RSTS/E SOFTWARE PACKAGES

- **KDSS**, a multi-terminal key-to-disk data entry system. (Also available for RSX-11M.)
- **TAM**, a multi-terminal screen-handling facility for transaction-processing applications. (Also available for RSX-11M.)
- **FSORT3**, a very fast sort. Directly sorts RSTS/E files containing up to 16 million keys or records. Up to 70 times as fast as the RSTS-11 Sort package in CPU time.
- **SELECT**, a convenient, very quick package for extracting records that meet user-specified selection criteria.
- **BSC/DV**, a device driver for the DEC DV11 synchronous multiplexer that handles most bisynchronous protocols.

- **COLINK**, a package that links two RSTS/E systems together using DMC11s. Supports file transfers, virtual terminals, and across-the-link task communication.
- **DIALUP**, a package that uses an asynchronous terminal line to link a local RSTS/E system to a remote computer system. Supports file transfers, virtual terminals, and dial-out through a DN11.

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

Evans Griffiths & Hart, Inc.
55 Waltham Street
Lexington, Massachusetts 02173
(617) 861-0670
### 1.2.1 JDFLG — Primary Job Status Flags

The job status flags contained in the word JDFLG are defined as follows:

<table>
<thead>
<tr>
<th>Bit</th>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>JFPOST</td>
<td>The monitor checks this bit before making a job runnable. If it is set the information in JDPOST is used as a mask for updating the job’s FIRQB or XRB. The information in J2PPTR and J2PCNT in JDB2 (see section 1.3) may also be used to post large amounts of data to a user-defined buffer.</td>
</tr>
<tr>
<td>1</td>
<td>JFIOKY</td>
<td>If this bit is set when a job is made runnable the job’s keyword is copied into location 400(octal) of the job’s image and the contents of JDIOST are posted to the job’s FIRQB.</td>
</tr>
<tr>
<td>2</td>
<td>JFRSX</td>
<td>If this bit is set the monitor resident RSX support is used to post the job information indicated by JFIOKY, rather than using the standard monitor routines.</td>
</tr>
<tr>
<td>3</td>
<td>JFCC</td>
<td>This bit is set when a (\uparrow) C is typed at the job’s terminal. When the job becomes runnable the pseudo-vector P.CC (see the System Directives Manual) will be taken unless JF2CC is also set.</td>
</tr>
<tr>
<td>4</td>
<td>JF2CC</td>
<td>This bit is set when a (\uparrow) C is typed and at least one (\uparrow) C has already been typed since the job was last run. When the job becomes runnable the pseudo-vector P.2CC will be taken.</td>
</tr>
<tr>
<td>5</td>
<td>JFPPT</td>
<td>If this bit is set when the job is made runnable the floating point trap pseudo-vector, P.FPP, will be taken.</td>
</tr>
<tr>
<td>6</td>
<td>JFGO</td>
<td>If this bit is set when the job is made runnable the floating point trap pseudo-vector, P.FPP, will be taken.</td>
</tr>
<tr>
<td>7</td>
<td>JFREDO</td>
<td>If this bit is set when a job appears runnable (and JFGO is not set) then the job is stalled waiting for an I/O completion and is not really runnable.</td>
</tr>
<tr>
<td>8</td>
<td>JFSYST</td>
<td>This bit is set if the job can use temporary privileges.</td>
</tr>
<tr>
<td>9</td>
<td>JFFPP</td>
<td>If this bit is set the contents of the floating point hardware (if any) are saved and restored along with the job image. This bit is one of the keyword bits (see the System Directives Manual).</td>
</tr>
<tr>
<td>10</td>
<td>JFPRIV</td>
<td>This bit is set if the job is logged into a privileged account. It is one of the keyword bits.</td>
</tr>
<tr>
<td>11</td>
<td>JFSYS</td>
<td>This bit is set if the job is currently running with temporary privileges. It is one of the keyword bits.</td>
</tr>
<tr>
<td>12</td>
<td>JFNOPR</td>
<td>This bit is set if the job is running non-logged in. It is one of the keyword bits.</td>
</tr>
<tr>
<td>13</td>
<td>JFBIG</td>
<td>If this bit is set the job can exceed its private memory size (as defined in JDSIZM). It is one of the keyword bits.</td>
</tr>
<tr>
<td>14</td>
<td>JFLOCK</td>
<td>This bit is set if the job is locked in memory. It is one of the keyword bits.</td>
</tr>
<tr>
<td>15</td>
<td>JFSPCL</td>
<td>This bit is set if some special processing is required before running the job. The flag bits in JDFLG2 specify the special action to be performed.</td>
</tr>
</tbody>
</table>

### 1.2.2 JDFLG2 — Secondary Job Status Flags

The job status flags contained in the byte JDFLG2 are defined as follows:

<table>
<thead>
<tr>
<th>Bit</th>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>JFCTXT</td>
<td>This bit is set if the job’s context is to be saved.</td>
</tr>
<tr>
<td>1</td>
<td>JFPRTY</td>
<td>This bit is set if the special condition shown by JFSPCL is a memory parity error.</td>
</tr>
<tr>
<td>2</td>
<td>JFRUN</td>
<td>This bit is set if the special condition shown by JFSPCL is a new program run entry request.</td>
</tr>
<tr>
<td>3</td>
<td>JFSWPR</td>
<td>This bit is set if the special condition shown by JFSPCL is a runtime system or resident library load failure.</td>
</tr>
<tr>
<td>4</td>
<td>JFSTAK</td>
<td>This bit is set if the special condition shown by JFSPCL is a stack overflow.</td>
</tr>
<tr>
<td>5</td>
<td>JFSWPE</td>
<td>This bit is set if the special condition shown by JFSPCL is a swap error.</td>
</tr>
<tr>
<td>6</td>
<td>JFKIL2</td>
<td>This bit is set when the job is being killed and the logout phase of killing a job has completed. The second phase of a logout, the removal of job control information, should be done now.</td>
</tr>
<tr>
<td>7</td>
<td>JFKILL</td>
<td>This bit is set if the job is to be killed.</td>
</tr>
</tbody>
</table>

If JFSPCL is set in JDFLG but no bits are set in JDFLG2, the current runtime system is entered at its P.STRT or P.CRAS entry point.

### 1.3 JDB2 — JOB DATA BLOCK TWO

The secondary job data block (JDB2) contains information about the job that is used less often or is less...
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 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.

LSI-11 EMULATING MASS STORAGE CONTROLLER, Model DQ 202. Cost effective interface of two 8 and/or 14-inch Winchester, 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.

PDP-11 MAGNETIC TAPE COUPLER, Model DU 130, offers features of Model DO 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.

PDP-11 EMULATING MASS STORAGE CONTROLLER, Model DU 202, offers same features as Model DU 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 GGV
Improve your communication
We've earned our reputation by designing and delivering dependable, easy to live with tape and disk controllers for DEC cpus. Products that really showed the industry how to do it.

Now we've cleaned up the communications I/O morass, with a new DH-11 type multiplexer for PDP-11 and VAX systems. It duplicates all the features of DEC's DH-11. And it goes well beyond, delivering improved line handling capabilities in a far smaller package and at significantly lower cost.

You might say we're now doing for communications controllers what we've already done for tape and disk controllers — applying the latest microprocessor technology and some sexy design to give our customers the performance and reliability they deserve.

Our newest achievement proves the point. To wit: just one hex size board, occupying a single Unibus slot, provides up to 64 lines for terminals, printers or other communications I/O. That's eight times more efficient than the DEC approach.

And our unique system organization means maximum flexibility. You can daisy-chain up to four remote distribution panels, expanding in 8 or 16 line increments all the way to 64 lines. You can configure any 8 line group for an RS-232 interface, with modem control a standard feature. Or you have the option of selecting a current loop interface. And because all your interface variations are made at the distribution panels, no additional hardware or cpu restructuring is needed when you add channels.

You get everything you got from DH-11, such as DMA on transmit operation and line format flexibility. Naturally, we're fully software transparent and run existing DEC diagnostics.

But we like to do everything we do just a little bit better than the other guy. So we boosted performance to 19.2 K baud per line. For peak traffic, we can 'double fifo' to 128 characters per 16 lines. Multiple controllers can be added for increased capacity. You get full 16-bit word transfers on DMA operation — not just byte transfers. And, of course, we've taken advantage of our system's high speed bipolar microprocessor to build-in extensive controller and channel self-test and fault indicators.

We're concentrating on DEC, making the world's best cpus even better with some ingenious controllers. We've done it for tape and disk interfacing. Now for communications I/O. We'd like to send you all the details we didn't have room to include here. So write or call Emulex Corporation. Or, if you just can't wait, contact the Emulex representative, dealer or distributor nearest you.

For immediate, off the shelf delivery contact our national distributor: First Computer Corporation; Corporate Square, 825 North Cass Avenue; Westmont, IL 60559; 312/920-1050.

EMULEX CORPORATION
2001 E. Deere Ave.
Santa Ana, CA 92705
714/557-7580
TWX 910-595-2521
time critical than the information in JDB. Its primary use is for accounting and directory information.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Offset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>J2CPU</td>
<td>0</td>
<td>This word contains the low order 16 bits of the total CPU time used by this job through the last time J2TICK was posted. The units of this value are 1/10th seconds.</td>
</tr>
<tr>
<td>J2CPU</td>
<td>2</td>
<td>This word contains the low order 16 bits of the CPU time for this job in device-minutes. A device-minute is the use of one device for one minute. Using two devices for one minute is two device-minutes, etc.</td>
</tr>
<tr>
<td>J2CON</td>
<td>4</td>
<td>This word contains the total connect time, in minutes, for this job. Connect time is only computed while a job is logged in.</td>
</tr>
<tr>
<td>J2KCT</td>
<td>6</td>
<td>This word contains the low order 16 bits of the job’s kilo-core-ticks. One kilo core tick is the use of 1 K-word of memory while executing for 1/10th second. Using 2 K-words for 1/10th second is two kilo-core-ticks. This word contains the total device time for this job in device-minutes.</td>
</tr>
<tr>
<td>J2DEV</td>
<td>8</td>
<td>This word contains the total device time for this job. The low byte contains the group number and the high byte contains the user number. If the job is not logged in this word will be 0.</td>
</tr>
</tbody>
</table>

- **J2TICK**: This word is incremented at each clock interrupt when the job is executing. When the job is descheduled this value is converted to the equivalent number of 1/10th seconds and added to J2CPU. Any amount less than 1/10th second is left in this word. The units of this word depend on how fast the clock is interrupting. For a KW11L clock, running at 60 hertz, the units are 1/60th seconds.

- **J2CPU**: This byte is the high order 8 bits of the kilo-core-ticks (see J2KCT above).

- **J2CON**: This byte is the high order 8 bits of the CPU time (see J2CPU above).

- **J2NAME**: These two words contain the program name in RADSO. The program name is specified using the .NAME SYSTEM call. All the standard runtime systems issue this call to post the program name each time a program is run. The contents of this word are for information only, and are ignored by RSTS.

- **J2DRTS**: This word is the address of the runtime system descriptor block (RTS) of the default RTS for this job. When a running program exits, control returns to this default RTS. If the default RTS is no longer installed when the program exits the system default RTS is used instead. RTS blocks will be described in the fourth article of this series.

- **J2MPTR**: If this job is a message receiver, this word contains the address of its receiver ID block (RIB). If this job is not a receiver this word is 0. RIB blocks will be described in the fourth article of this series.

- **J2PCNT**: This word is used as a pointer to a monitor buffer to be used to transfer information to or from a user program buffer. It is normally used for large message send/receive buffer transfers. If the lower 5 bits of the pointer are 0 the pointer is an address into the small buffer area. If the lower 5 bits are not 0 the pointer is an address into the large buffer area that has been rotated left 7 bits.

- **J2PPN**: This word contains the job’s PPN. The low byte contains the group number and the high byte contains the user number. If the job is not logged in this word will be 0.

- **J2UFDR**: This word contains the device cluster number (DCN) of the first cluster on SYO: of the UDP specified in J2PPN above. The value of this word is undefined if the job is not logged in.

- **J2WPTR**: If the job is attached to any resident libraries this word contain the address of the job’s first window.

... continued on page 58
INSIDE:

☐ The Other Story About VAX and VMS
The Other Story About VAX and VMS

By Ross W. Miller
President, Online Data Processing, Inc.

In 1977 just prior to the VAX being announced, I was discussing with a group of people some short-falls of the PDP11 computer. (1) Memory Management Concepts; (2) Memory Addressability; (3) Shared Memory Protection.

We also discussed different data types that are required to do a good job of data processing and the type of instructions one should have. I was greatly surprised and pleased when I went to the Fall DECUS of that year and DEC announced the VAX computer and described what was to me an almost perfect hardware design.

The design of the VAX is one of the best I've seen for current technology and it will give a considerable amount of flexibility in incorporating new concepts which have yet to be developed. Overall I would give DEC very high marks for the VAX machine. However, there are some problems. The purpose of this article is to provide some insight as to where those problems might be found, especially since many users are looking to the VAX to alleviate through-put problems and avoid costly conversions. Rather than spending a lot of time on the positive details of the VAX which you can read from all sorts of DEC literature, I'll take a few moments to enlighten you on some of the surprises and points of interest that are not discussed by DEC.

The single largest problem with the VAX computer is the fact that the DEC commercial product line people didn't want to become involved in the early design of the VAX computer and its operating system. I am sure there are going to be some people jumping up and down and saying, "Miller you are all wrong!... we were there," but unfortunately, they weren't there when they needed to be. Many things were designed into the VAX computer attempting to be sensitive to commercial interest in areas such as the ability to handle strings more efficiently, binary coded decimal data types and formatting instructions. However, DEC missed the boat in designing the operating system for commercial applications in such areas as operator interface, facilities for handling printers, record management efficiencies and what I would consider some significant oversights in the primary migration language for DEC's current customer base. This being VAX BASIC.
VAX BASIC — There are some good things and some bad things about it. First the good things... they have tried to maintain the interactive mode of programming which is a very positive feature of BASIC PLUS. Now the bad things... they didn’t go far enough with the interactive features; secondly, it suffers greatly in the area of performance; and thirdly, by trying to give the impression that you can make minor modifications to your BASIC PLUS code and begin executing it in VAX BASIC, you could be misled into doing things that will be detrimental in the areas of performance and effectiveness use of the VAX machine.

Unwittingly, many of us who have been exposed to BASIC PLUS programming have developed several shortcuts and concepts on resolving data processing problems by automatically taking into account the restrictions that BASIC PLUS imposes. By using the same approach to develop your VAX BASIC program as you used with the BASIC PLUS program, you could be hurting yourself; i.e. structuring your program, data types used and, segmenting functions that should not be segmented. However, you automatically do it to circumvent BASIC PLUS restrictions.

With regard to VAX BASIC performance, we found that it is indeed fast in most bench marks, and specifically, in doing computational type operations. My point is: In a commercial application, a very small portion of your application program is actually involved in doing computation. Commercial applications deal mostly with string manipulation and record management functions; searching through a file, doing compares, changing data, updating information in records, etc. Therefore, it is my opinion that most bench marks, that have been run, do not reflect the true commercial environment. We have found that when applications written in VAX BASIC are compared to BASIC PLUS application of the same type (same concepts used) running on the 11/70, surprisingly the CPU time is comparable and this is disappointing.

In trying to determine exactly where the problems occur in VAX BASIC performance, we have traced through listings and micro-fiche. Many of the specific commercial routines that do string manipulation, formatting of data, and record management will do a call to a routine to perform a function which could have been executed in one to three machine instructions. As you trace these calls into the run time library, you find that these routines do calls to other routines, which do calls to other routines, etc. until finally, it gets down to the routine which actually performs the function and does the two or three instructions. It then starts to do the returns and unwinds. Naturally, all this is overhead and can increase the possibility of page faults as you wander your way through the mazes of twisty little passages, all alike, in the run time library.

It appears that a PDP11 programmer wrote the VAX BASIC programs and got carried away with the idea of modular programming forgetting that the VAX had built in instructions to accomplish what you want in machine instructions. It’s not as fast as advertised, but it could be much faster.

DATA BASE MANAGEMENT — The VAX machine is the right machine to implement a good DATA BASE management system. As of this time, I have not seen a good DATA BASE management system for the VAX but nothing in the design of the operating system prevents it from being an excellent DATA BASE system. One of the major requirements for DATA BASE management is to have a large address space and do a reasonable job of caching frequently used sections of the DATA BASE. A 16 bit machine does not have the address space necessary and can cause you to do excessive thrashing of the disk by using DATA BASE system. A good DATA BASE system should reduce disk access for you.

RMS — Another significant point, is the use of RMS on the VAX computer. RMS is built into the operating system and this has its good points and its bad points. The positive aspect is they have reduced the overhead for RMS significantly from the PDP11 implementation. Those who are familiar with RMS on the 11 will be pleased with the VAX. However, there is still a significant amount of overhead in RMS due to the conservative implementation. An example of this is that whenever you do a put of a record which is 50 or 60 characters long, it will cause a write of the entire block back to the disk. To get better performance, it is preferable to keep mass storage I/O to an absolute minimum since that is typically the most time consuming operation on a computer. Because of this, it is easy to get into I/O bottlenecks with RMS under VAX.

I am confident that performance improvements are being made to the RMS facilities. My biggest concern is that the VAX BASIC people do not seem to be concerned about performance issues; nor are they interested in providing additional features which will allow the user to control performance issues such as the many things that RMS will allow you to do but VAX BASIC doesn’t provide a clean way of accessing.
VMS — Furthermore, there are some things in the operating system which definitely need to be improved.

1. If you are connected to a dial up line and you lose carrier on that line, your job gets killed. There is no way of preventing it. This isn't very convenient when you are updating multiple files simultaneously.

2. There isn't any way of doing a detach of a job and attaching to a job.

3. There is no way to do a force to a terminal for the system manager to help naive users or to get an X-offed LA 180 started again.

4. It is not possible to mount a tape under program control. The only way you can mount a tape is at VMS DCL level.

If you can imagine a situation in a commercial environment where you need to process for several hours and then mount a tape, do further processing, change to a different tape to continue processing, you will definitely have problems doing this currently under VMS.

On the positive side, spooled printers and terminal are great. There needs to be additional improvements in forms management and controlling who can be an operator for a specific printer, such as a remote printer, but these again can be resolved.

Things such as being able to call routines written in any language from any other language is a beneficial feature. You can optimize certain functions that are not in the native language that you are using but are executed frequently. Even though I discourage mixing macro code and other such routines in business applications, we have used a few macro coded routines very effectively in business applications for forms management and screen handling routines which we developed prior to DEC having anything available.

In addition, we found there were some DCL commands that were not provided so it was very easy for us to write the routines the way we wanted them and provide them for our clients as though they were part of the standard DCL set.

While you may think this has been a negative report, I wish to assure you that this is not my intent. However, I do want you to be aware of the things that are not advertised or widely known. Most of these problems are a matter of presenting them to the proper individuals to get them fixed or modified. Currently, DEC definitely needs strong input from the commercial world. DEC's turnaround time however, is 1½ to 2 years. That means that if you put in a request right now for a new feature or a change and DEC accepts it, you won't see it for 1½ to 2 years. That is the time it takes to implement, test and distribute the new feature or change. So . . . if you wait, you can calculate for yourself when it will be available.

**In conclusion,** I'm 100% behind VAX/VMS, but I would caution you to be extremely careful about assuming anything about the VAX system, and especially how to get performance out of it. We have been bit the hardest by using functions that worked extremely well on the PDP 11 but when we implemented them on the VAX we were actually working against VMS and thereby increasing system overhead.
ROSS/V is a software package, written in VAX-11 MACRO, which provides a RSTS/E monitor environment for programs running in PDP-11 compatibility mode on DEC's VAX-11.

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

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

ROSS/V is available from:

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

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

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

AND NOW, FOR THE 64K QUESTION!

CONTESTANT A

CONTESTANT B

CONTESTANT C
SCENES FROM CANADIAN DECUS 1981

VAX-11/750.

Starting today, the size of your program no longer determines the size... or price... of your system.

The VAX-11/750 fits countless applications waiting for a new kind of computer system.

The architectural breakthrough: 2 gigabytes of program space for every user.

The software you need to put large jobs onto a small computer is already written.

How the VAX-11/750 redefines distributed processing to help you control costs, increase productivity.

The technological benefit: Price/performance and reliability.

VAX-11/750.

Enfin un système dont la dimension... et le prix... ne dépendent plus de l'ampleur du programme.

Le VAX-11/750 répond à une multitude de besoins qu'aucun système n'a pu satisfaire jusqu'ici.

Une architecture révolutionnaire: 2 milliards d'octets d'espace-programme par utilisateur.

Un logiciel qui vous permet d'exécuter de gros travaux sur un petit ordinateur.

Une nouvelle décentralisation du traitement des données qui vous aide à mieux contrôler les coûts et à augmenter la productivité.

The Supermarket of Computer Products
PRESENTS
Disk Sub Systems Ala Carte

DEC, DATA GENERAL and Others
For LSI11*, PDP11*, PDP11/70*, VAX*, UNIBUS*, & MASSBUS*:
*Tradenames of Digital Equipment Corp.

Featuring:

- CONTROL DATA's
  SMD, CMD, MMD, FMD,
  HAWK, LARK, PRINTERS,
  TAPE DRIVES & MEMORIES

- KENNEDY's
  WINCHESTERS
  & TAPE DRIVES

- DATA PRODUCT's
  PRINTERS

- CENTURY DATA's
  TRIDENTS

Plus a Menu of CONTROLLERS:
- EMULEX • SYSTEM INDUSTRIES
- WESTERN PERIPHERALS
- DILOG • BYTRONIX • RIANDA
- MCT • DIGIDYNE

ONLY GF OFFERS THIS UNIQUE COMBINATION:
- CUSTOM-TAILORED TO YOUR EXACT REQUIREMENTS
- FULL EMULATION
- COST SAVINGS OF 50%
- FAST DELIVERY
- EXPERT INSTALLATION & SERVICE
- OEM DISCOUNTS AVAILABLE

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

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

GF: I am interested in:
[ ] Printers [ ] Controllers [ ] Memories
[ ] Tapes [ ] CRTs [ ] Media

I am an:
[ ] End user [ ] Institutional [ ] OEM

My requirements are:
[ ] Immediate [ ] 3-6 months [ ] Information only

Name ________________________________
Title ________________________________
Company ____________________________
Address ______________________________
City ______________________ State ______ Zip ______
Telephone (______) ____________________

My computer is a ____________________
1.4 IOB — I/O BLOCK

The I/O block is used to access information about each open channel. It contains one entry for each of the possible 16 channels. (Note: BASIC-PLUS only allows 12 channels because channel 0 is the user's terminal and channels 13-15 are used internally by the BASIC-PLUS interpreter).

The IOB is 16 words long. Each word is 0 if the corresponding channel is closed and non-zero if it is open. If the channel is open this word contains the address of the device data block (DDB) for non-disk devices, the window control block (WCB) for disk files on large-file systems or the small file control block (SCB) for disk files on small-file systems.

One exception is the entry for channel 0. This entry corresponds to the job's terminal. If the job is running detached this entry may still point to a terminal DDB. However the ownership byte within the DDB (DDJBNO) will not contain this job's job number.

1.5 WRK — WORK BLOCK

The work block is a scratch work area used to hold information that is normally contained only in the user program area. This allows the job to be swappable during certain long monitor calls. The work block normally is used to store the FIRQB during FIP calls and the XRB during I/O transfers.

1.6 FIXED MEMORY LOCATIONS

Several fixed locations in low memory within the monitor are used to store information about the currently executing job. This information provides a shortcut for accessing the most used job control information. The information is as follows:

<table>
<thead>
<tr>
<th>Address</th>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
</table>
| 518     | JOB      | This byte contains the job number (times 2) of the currently executing job. If this byte is 0 the null job is running. In that case the words that follow have special meanings, as noted. This byte is accessed in BASIC-PLUS by PEEK(518%) AND 255%.
| 519     | NEXT     | If this byte is non-zero it contains the job number (times 2) of the job that was scheduled to be the current job but is not yet swapped into memory. This job will start execution immediately upon gaining memory residency. In this case the job shown in JOB is "sub-scheduled" to make use of the available CPU time while waiting for the next job to swap in. This byte is accessed in BASIC-PLUS by SWAP%(PEEK(518%)) AND 255%.
| 520     | JOBDA    | This word contains the address of the job data block (JDB) for the currently executing job. If JOB contains a 0 this word will be 0. This word contains the address of the flag word, JDFLAGS, in the current job's JDB. If JOB contains a 0 this word will contain the address of a dummy value of 0.
| 522     | JOBF     | This word contains the address of the data block (JDB) for the current job. If JOB contains a 0 this word will contain the address of a dummy value of 0.
| 524     | IOSTS    | This word contains the address of the JDB (JDB2) for the current job. If JOB contains a 0 this word will contain the address of a dummy value of 0.
| 526     | JOBWRK   | This word contains the address of the work block (WRK) for the current job. If JOB contains a 0 this word will be 0. This word contains a pointer to the second job data block (JDB2) for the current job. If JOB contains a 0 this word will contain the address of a scratch location to be used for J2TICK.
530 JOBRTS This word contains the address of the runtime system block (RTS) for the current job. If the current job is using the disappearing RSX runtime system this word will contain the address of the null RTS descriptor, NULRTS. If JOB contains a 0 this word will be 0.

532 CPUTIM This word is the address of the value J2CPU in the second job data block (JDB2) for the current job.

534 JOBWDB This word contains the address of the first window descriptor block (WDB) used by the current job. If the current job is not attached to any resident libraries, or if JOB contains a 0 this word will be 0.

1.7 JBSTAT, JBWAIT — JOB STATUS TABLES

Two tables, JBSTAT and JBWAIT, are used to determine if a job is runable. These tables each have one word for every possible job on the system, including the null job (job 0). They are accessed by using the job number times two as an offset.

When the scheduler wants to determine whether a job is runable it performs a logical AND of the bits in the job's JBSTAT entry with the job's JBWAIT entry. If the result is non-zero the job is runable. If the result is 0 the job is stalled waiting on something and cannot be run.

When a job stalls for any reason (such as doing I/O) the JBSTAT entry corresponding to its job number is set to 0 and a bit is set in the corresponding JBWAIT entry to show what the job is waiting for.

Stalling a program for hybernation clears both the JBSTAT and the JBWAIT entry. When the job is later re-attached to a terminal JBWAIT will be updated to some non-zero value.

When an I/O completes or some asynchronous event occurs, such as a message receive or sleep timeout, the monitor sets an appropriate bit in the job's JBSTAT word. Depending on what the program was stalled for, this may or may not make the job runable. However, sooner or later the event the job is waiting for will complete and the job will be made runable again.

The bit values within the JBSTAT and JBWAIT entries have the following values:

<table>
<thead>
<tr>
<th>Bit</th>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>JS.SY</td>
<td>This bit is used for I/O on all synchronous devices. This includes disk, floppy disk, magtape, etc. but does not include terminals.</td>
</tr>
<tr>
<td>1</td>
<td>JS.KB</td>
<td>This bit is used for terminal input. This bit is assigned at sysgen time for devices (such as LP:) that can stall and de-stall a job while still processing its I/O.</td>
</tr>
<tr>
<td>2</td>
<td>JS.FIP</td>
<td>This bit is assigned at sysgen time for devices (such as LP:) that can stall and de-stall a job while still processing its I/O. This bit is assigned at sysgen time for devices (such as LP:) that can stall and de-stall a job while still processing its I/O.</td>
</tr>
<tr>
<td>3</td>
<td>JS.FIP</td>
<td>This bit is assigned at sysgen time for devices (such as LP:) that can stall and de-stall a job while still processing its I/O. This bit is assigned at sysgen time for devices (such as LP:) that can stall and de-stall a job while still processing its I/O. This bit is assigned at sysgen time for devices (such as LP:) that can stall and de-stall a job while still processing its I/O.</td>
</tr>
<tr>
<td>4</td>
<td>JS.SYM</td>
<td>This bit is assigned at sysgen time for devices (such as LP:) that can stall and de-stall a job while still processing its I/O.</td>
</tr>
<tr>
<td>5</td>
<td>JS.SYM</td>
<td>This bit is assigned at sysgen time for devices (such as LP:) that can stall and de-stall a job while still processing its I/O.</td>
</tr>
<tr>
<td>6</td>
<td>JS.SYM</td>
<td>This bit is assigned at sysgen time for devices (such as LP:) that can stall and de-stall a job while still processing its I/O.</td>
</tr>
<tr>
<td>7</td>
<td>JS.SYM</td>
<td>This bit is assigned at sysgen time for devices (such as LP:) that can stall and de-stall a job while still processing its I/O.</td>
</tr>
<tr>
<td>8</td>
<td>JS.SYM</td>
<td>This bit is assigned at sysgen time for devices (such as LP:) that can stall and de-stall a job while still processing its I/O.</td>
</tr>
<tr>
<td>9</td>
<td>JS.SYM</td>
<td>This bit is assigned at sysgen time for devices (such as LP:) that can stall and de-stall a job while still processing its I/O.</td>
</tr>
<tr>
<td>10</td>
<td>JS.SYM</td>
<td>This bit is assigned at sysgen time for devices (such as LP:) that can stall and de-stall a job while still processing its I/O.</td>
</tr>
<tr>
<td>11</td>
<td>JSTEL</td>
<td>This bit is used for terminal output.</td>
</tr>
<tr>
<td>12</td>
<td>JSFIP</td>
<td>This bit is used for FIP (SYS call) waits.</td>
</tr>
<tr>
<td>13</td>
<td>JSTIM</td>
<td>This bit is used for timeouts from various time restricted events, such as SLEEP, terminal input timeout, message receive timeout, etc. This bit is used to wait for small buffers when no buffer space is available for I/O buffers.</td>
</tr>
<tr>
<td>14</td>
<td>JSBUF</td>
<td>This bit is not currently used but is reserved for future use.</td>
</tr>
<tr>
<td>15</td>
<td>JSTEL</td>
<td>In the next issue: MEMORY CONTROL</td>
</tr>
</tbody>
</table>

SPECIAL OFFER

A wall chart showing the linkages between all tables in the RSTS/E monitor is available for $1.00. Send your requests to Northwest Digital Software, Box 2-743, Newport, WA 99156.
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

IT

COMPU-TOME, INC.
668 S. Sunset Avenue
West Covina, CA 91790
(213) 960-2895
All Things BRIGHT and Beautiful

By Michael J. D. Mowat, B.Sc., M.B.C.S.
Department of Agriculture and Fisheries for Scotland

Creating a ‘Proper’ RSTS Disk with Contiguous Pre-extended Directories

If you do a weekly backup copy of your system disk, you could start the week with a restructured disk to get the most out of your system.

The advantages of having a well constructed system disk on a RSTS system are great. The features required on such a disk are that the directories should be contiguous and pre-extended and should be placed near the centre of the disk along with the SATT.SYS and the swapfiles, and that all other files should be physically contiguous as far as possible.

Such a disk is very fragile because user files which are not logically contiguous will fragment as they are modified, and if the system requires that accounts be removed or added the contiguity of the directories will gradually deteriorate. Also if a directory is zeroed its contiguity will be lost. It is possible to build a disk with contiguous directories by starting with a newly initialized disk and creating all the accounts, then pre-extending all the directories by opening null files and closing them.

With a system with a large number of accounts, this is an extremely time consuming process and it is not reasonable to repeat the process if accounts are created or deleted. System disks do however require to have back up copies made and if a faster method of producing a disk with a proper structure were available it would be possible to rebuild the disk at every backup session, taking account of any alteration in the account structure. Provided that the system can be taken off time sharing for the time necessary to copy the files using PIP the program BRIGHT.BAS described here can build the skeleton disk very quickly (about 10 minutes for an 80 megabyte disk with 100 accounts).

The new disk is initialized as a system disk with the directories pre-extended and the SATT, swapfiles and the directory of [1,2] are placed. The program BRIGHT as it stands assumes that these are placed contiguously from cluster 20000 with a pack cluster size of 4. The program must be changed for other settings. When the disk has been initialized and any necessary files added it may be saved to another medium, or an image copy may be made to use the next time the system is rebuilt, to save having to go through the initialization dialogue again. Typically the skeleton disk would have the following.

- Bootstrap
- INIT.SYS
- A SIL
- BASIC.RTS
- RT1I.RTS
- SWAP files
- CRASH.SYS
- INIT.BAC
- PIP.SAV
- UTILITY.BAC
- BRIGHT.CTL (Command file to build the disk)
- ERRINT.BAC
- ATPK.BAC
- LOGIN.BAC
- LOGOUT.BAC
- BRIGHT.BAS (or BAC)

The new pack is booted on DBO: and the old pack is mounted (physically and logically as a private pack) on DB1:. The program BRIGHT.BAS is then run. The first thing this program does is to build an MFD containing all the accounts from the old system disk. It then opens a contiguous file into which to build the directories and then pre-extending all the directories by opening null files and closing them.

The old disk is kept as a backup volume and it can be remounted in the event of a disaster. Recovery of a single file from the old volume is very quick and easy using PIP.

The prerequisites are:

1. An initialized system disk with the system directories pre-extended and with swapfiles included. This disk must contain the program BRIGHT.BAS and may in addition contain a startup command file (using ATPK) to make the operation automatic. In this case it must also contain all cusp programs and command files used before account [1,2] is copied to the new disk.

2. The previous system disk with all changes made to the structure of the accounts.
When the directories have been built the files are copied from DB1: to DB0: using PIP.SAV. If your system uses a new files first directory structure, the disk should be initialized as NOT new files first so that a PIP ** will not reverse the order. The program INVERT.BAS is then run to change to new files first. This program must run in account [1.1] as it must tell the monitor that the change has happened (since the disk descriptor is read only on mount).

The listings of BRIGHT.BAS and INVERT.BAS are as follows:

**BRIGHT.BAS**

I EXTEND
180 THIS PROGRAM WILL RUN ON A SYSTEM DISK AND WILL CREATE THE SAME ACCOUNTS AS ON AN OLD SYSTEM DISK WHICH MUST BE MOUNTED AS A PRIVATE DISK ON ANOTHER DRIVE.
190 THE PROGRAM IS WRITTEN TO HANDLE THE NEW DISK ON DB0: AND THE OLD DISK ON DB1:
200 THE PROGRAM WILL EXTEND THE DIRECTORIES OF ALL ACCOUNTS TO 1000 CLUSTERS AND THEN REARRANGE THE ACCOUNTS IN ALPHABETICAL ORDER.
220 THE PROGRAM ASSUMES A CLUSTER SIZE OF 4.
240 PRINT "THE PROGRAM WILL RUN ON A SYSTEM DISK AND WILL CREATE THE SAME ACCOUNTS AS ON AN OLD SYSTEM DISK WHICH MUST BE MOUNTED AS A PRIVATE DISK ON ANOTHER DRIVE.
250 THE PROGRAM IS WRITTEN TO HANDLE THE NEW DISK ON DB0: AND THE OLD DISK ON DB1:
260 THE PROGRAM WILL EXTEND THE DIRECTORIES OF ALL ACCOUNTS TO 1000 CLUSTERS AND THEN REARRANGE THE ACCOUNTS IN ALPHABETICAL ORDER.
280 THE PROGRAM ASSUMES A CLUSTER SIZE OF 4.

**INVERT.BAS**

OPEN "C:", 3;"UFDFIL.SYS" FOR OUTPUT AS FILE "UFDFIL.SYS", MODE 16384

A typical start up command file for use with ATPK is as follows. You can start it and leave it. The system will start up from the new disk when it is finished. The time taken will
depend on the complexity of your accounts. For a system with 120 accounts and 120,000 blocks in 7000 files the time was about two hours on an RP04. (considerably longer the first time from the rubblishy old disk).

```
RUN (1,2)UTILTY
SEND KB8: Building Directories
ADD RTI:
MOUNT DBI: RSTS-PRIVATE
UNLOCK DBI:
EXIT
RUN (1,2)BRIGHT
RUN (1,2)UTILTY
SEND KB8: Transferring files.
EXIT
RUN (1,2)IP,SAV
DBB(0,1,2)0.P0:000000<0.x,1,2>PE:12<0>MOD:12<0>MOD:12<0>
INSTALL FILES MAY BE PLACED BY INSERTING
1 COMMANDS HERE
+Z
1 THIS OPTITIONAL FILE HOLDS SPACE NEAR THE CENTRE FOR NEW FILES
AND WORK FILES.
OPEN 'FILLER.TEMP:PA:000000' AS FILE 0X,FILESIZE 16384x
CLOSE 0X
RUN (1,2)UTILTY
SEND KB8: File transfer complete.
REMOVE RTI:
LOCK DBI:
DISMOUNT DBI:
EXIT
HELLO 1,1.SYSTEM
RUN 1,1 файl
HELLO 1,1,FİLES
WK!1,2,RTS.CMD
WK1,2,POOL.CMD
WK1,2,CLK.CMD
WK1,2,TTY.CMD
WK1,2,PLT.CMD
RUN (1,2)UTILTY
```

**RSTS PROFESSIONAL**

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

- Please enter my subscription for one year (4 issues).
  - All other countries: enclose $30, payable in U.S. dollars.

- Please enter my subscription for one year (4 issues), and BILL ME for $20 ($30 foreign, payable in U.S. dollars).

Please send Back Issues circled:

- $10 per issue encosed.

- Bill me for $12.50 per issue.

Vol. 1, #1 | Vol. 2, #2 | Vol. 2, #3 | Vol. 3, #1

Name ____________________________
Address ____________________________
City ___________ State ______ Zip ________

**FREE CLASSIFIED AD WITH SUBSCRIPTION!!**

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

```
RENEW LOGICAL LB
ADD LOGICAL SYM, 1,1,0,0
ENABLE CACHE
LOGING
SEND ALL RSTS/E V07.0 NOW UP AND RUNNING
SEND ALL Please say HELLO
EXIT
DIR BRIGHT,DIR*<0,*,0,S
```

The procedure to initialize the disk, and to position the swap files, and run time systems should be similar to the following description.

**Instructions for the Weekly Disk Copy.**

1. Run shutdown
2. Mount a scratch disk on DB1:
3. DSINT DB1 with the following dialogue

   **Option? DS**
   **DS-YYYY-M? Date or LF**
   **BB-00 ? Time or LF**
   **Disk? DB1**

   A description of the disk may be output.

   Pack 1ST RSTS?
   Pack cluster size? 4
   SATT,SYS base? 00000
   "S" password? SYSTEM
   "S" cluster size? 16

   Pre-extend directories on? Y
   PUB, PRI, or SYS SYST
   Library password? FILES
   Library UFD cluster size? 0
   Library account base? 00000
   Date last modified? YES?
   New files first (no)? YES?
   Use previous bad block info (yes)? YES
   (Not asked if not a RSTS disk)
   Format (no)? YES
   (Unless a new disk)
   Patterns? 4 (but use 4 at least if not a RSTS disk)
   Proceed? Y (If happy with dialogue)

4. Continue as follows when the disk is initialized.

   **Option? COPY**
   **DS-YYYY-M?**
   **HFT?**

5. To which disk? DB1

   Enabling only consoles, disks, and tapes.

   **RSTS V7.0 (DB1)**

   **Option? HARDR
   HARDR suboption? Hertz 50**
   **HARDR suboption? CSR**
   **Controller with non-standard address? DZB**
   **New controller address? 16888**
   **HARDR suboption? VEC**
   **Controller with non-standard vector? DZB**
   **New vector address? 318**
   **HARDR suboption? CSR**
   **Controller with non-standard address? DZ1**
   **New controller address? 16808**
   **HARDR suboption? VEC**
   **Controller with non-standard vector? DZ1**
   **New vector address? 328**

   **HARDR suboption? LIST**
   **(CHECK THAT THE HARWARE LISTING IS CORRECT)**
   **HARDR suboption? LF**
   3 changes being made

   Rebooting . . .

   **RSTS V7.0 (DB1)**

   **Option? INSTALL**
   5ii7 RSTS/V

   **Option? BOOT DBB**

   **RSTS V7.0-07 Marine Lab (DBB)**

   **Option? ST**

   Command File Name? PROPER.CTL (This copies files required to run BRIGHT)
   Command File Name?

   When system shuts down put old disk on DB1: and new disk on DBB:
   Then start with command file BRIGHT.CTL
   (note that DEFAULTS must first be set with BASIC as the RSTS)
   The system will come up without DB1 when the copy disk is complete.

   A similar process may be used with a disk used as a non system disk but of course automatic start up would not be possible unless you have more than two drives, in which case the process could be done on line. If you have only two the non system disk would ahve to have a system on it in order to run BRIGHT and do the copying.
CLEN CLEN
CLEN CLEN
CLEN CLEN
LENUPCLE PCLE CL UP EN PC
ENUPCL UP EN CL UP ENUPCL
NUPC UP EN CL UP EN
NUPC UP EN CL UP EN
NUPC PCLE LENU EN

1!THIS SHORT PROGRAM DEMONSTRATES CLENUP ON A SMALL BUBBLE SORT
2DIML(20%):L(I%)=RNDFORI%=1%TO20%:BUILD UP SORT LOOP
3&'STARTING SORT AT';TIMES(0%)
4Z1%=0%:Z%=20%:WHILEZ%:FORJX=1XTOZX-1X:IFL(JX)<=L(JX+1X)THEN5SELSEL=L(JX):
L(JX)=L(JX+1X):L(JX+1X)=Z%=JX:WE NEEDED TO FLIP HERE
5NEXTJX:Z%=Z1%:Z1%=0%\NEXT:&'SORTING ENDED AT';TIMES(0%)
6STOP
PCLE EN PC
UP EN ENUPCL
UP EN EN
UP EN EN
PCLE EN

00001 !THIS SHORT PROGRAM DEMONSTRATES CLENUP ON A SMALL BUBBLE SORT
00002 DIM L(20%)
     L(I%) = RND
     FOR I% = 1% TO 20%
00003 !BUILD UP SORT LOOP
00004 'STARTING SORT AT'; TIMES(0%)
00005 Z1% = 0%
     Z% = 20%
     WHILE Z%
00006 NEXT JX;
     Z% = Z1%
     Z1% = 0%
     NEXT
     PRINT 'SORTING ENDED AT'; TIMES(0%)
00007 STOP

CLENUP is a service that saves you money and time. With the cost of program maintenance ever increasing, CLENUP will make your BASIC-PLUS programs easier to read so your programmers will spend less time and money on the program changes needed. CLENUP does not change the logic or efficiency of your programs - CLENUP reformats the source code to a pleasing, visually logical format. CLENUP is not expensive, and in most cases, your programs will be back within one week.

For additional information, please call: Gary Oppenheimer (212)787-2416
Arnold Singer (212)279-3995

or write: Aim Up
   Attn: CLENUP office
   370 Seventh Ave. Suite #434
   New York, N.Y. 10001

Note: BASIC PLUS is a trademark of DEC.
TECHNICAL NOTES
By The RSTS "Pro" Staff

An understanding of what your computer is doing at the basic level is important in evaluating performance (or lack of it) and deciding what can be done to improve it.

This will be the first in a continuing series to help you understand the physiology of a PDP 11. I am not a hardware expert; lots of people think I am not a software expert, but I am willing to stick my neck out and let you have the ideas as I understand them. As an editor, I will start and keep the ball rolling. I will, when offered, submit other comments and articles to the newsletter. I ask that you send them to me so that I may exercise my editor's perogative and put them in some order.

This first article will deal with input and output. All PDP 11 devices have associated with them a set of registers which may be accessed as if these registers were in the main memory. Each device has associated with it at least two registers: a) a control and status register and, b) a data buffer register. Some devices have more than two registers. For example, one of the simplest devices attached to the PDP 11 may be the console terminal. It has four addresses or registers associated with it. They are: 1) printer data, 2) printer status, 3) keyboard data, and 4) keyboard status. You will have by now noticed that the console is really two devices: a reader and a printer. To print a character on the console printer, all that is necessary is that you deposit via a control and status register for the keyboard. After 'reading,' a bit in the status register is set to indicate that no data is currently available in the data register. This means, of course, that you now have read the data that was in there. Note that reading a character from the keyboard does not echo it on the terminal printer. This must be done by putting the data that has been read into the printer data register thus echoing the character read.

Notice that there are no true I/O instructions: only the transfer from memory location to memory location is necessary in order to do input or output.

Although this type of input/output programming is simple and straightforward, it would be inefficient in the case of trying to control a multi-terminal system. For instance, a system that had 16 keyboards attached to it would necessitate that the controlling program periodically scan the data or control registers of the keyboard to see if anything had been input. If nothing had been input on that keyboard, it could continue to the next keyboard to see if there was any data there to be input. This technique is known as polling. That is, the controlling program polls or asks each terminal if there is data that it has to transmit. If not, the CPU continues around checking each terminal or device to see if there is any action to be taken. Thus, it becomes a totally CPU-bound job to keep checking your devices to see if there is any input to be done. A second and, as you can see, more efficient class of I/O programming is the so-called interrupt directed input and output. In the control and status register of our console terminal, there is a bit that enables an interrupt. That is, when data becomes available in the data register in the case of input or output, it causes the program to interrupt. What happens when the interrupt occurs from the console terminal? At every CPU cycle the PDP 11, via hardware, is checking to see if an interrupt has occurred. The PDP 11 utilizes a priority vectored interrupt scheme. Let's take them one at a time.

First, there is a priority interrupt. The CPU must deposit or move from one real memory location to the data buffer or data register of the particular device each character separately for the transfer. This is all right for a slow device such as the DEC writer, but can cause problems for devices that go quite a bit more rapidly. When you consider that an LA 36 transmits 30 characters a second while a TU 10 magnetic tape transmits 36,000 characters a second, the overhead involved with transmitting characters through a magnetic tape would greatly exceed that of an LA 36. Therefore, the CPU must not do data transfers simultaneously available within the PDP 11 hardware. This is the so-called NPR transfer. The PDP 8 refers to a data break facility and a generalized computer term for this is devices that work on a cycle stealing basis. (Non Processor Request) Devices that operate on a cycle stealing basis steal an occasional cycle from the CPU in order to do their input and output. This means that the devices themselves have the ability to do data transfers from memory to the device or from the device to memory without processor intervention. These 'smart' devices need only be told by the processor what location in memory to begin transfers from, how many locations to transfer, where on the peripheral device to begin putting the data and then these devices begin doing it by themselves. When they are finished, they automatically interrupt the CPU. Take the situation of transferring a file from disk to memory. The device, in this case the disk, needs to be told at what memory locations to begin putting the data, how many words (Bytes) to transfer, and where on the disk to begin getting the data.
What Can One Coupon Get For You?
A Free Resume Kit...
From the Only Employment Service with Twin Inhouse PDP 11/70 RSTS systems.
An Invaluable Aid In Your Job Changing Process

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

Name
Applications

Degree
Yr. Grad.
No. Yrs. Experience

Current Employer

Job Title

Languages

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

Scientific Placement, Inc. Employment Service
P.O. Box 19949 Houston, Texas 77024 (713) 496-6100

Once these registers are loaded, the I/O begins to take place in the same fashion as a character was printed on the console terminal when the printer register was loaded with the data to be printed. When the number of words that we specified to be transferred has been transferred, the device automatically interrupts the CPU for more information. This is the so-called stealing of a cycle from the CPU. It would be possible to transfer many thousands of Bytes, or words, of information between peripheral devices only utilizing a very few cycles of the CPU. While this transfer is going on, the CPU is free for other instructions.

In the next article, we will deal with more specific high-speed, large data capacity devices, particularly RPO 2, 3, 4, 5 and 6 disk drives. We will also attempt to talk about the Unibus structure, how data is transferred over the Unibus and the Unibus extensions: the Fastbus and Massbus.

For more information about this article and future articles, I suggest the following reading. In 1968 or 1969, DEC published a book called, Introduction to Programming. Although this book is based on the PDP8 series of computers, many of the concepts have carried over to the PDP 11. Next, the PDP 11 Peripherals Handbook contains in its introduction and beginning chapters useful information on programming peripheral devices. The addresses and use of different control and status registers for different devices can be found in this particular book. I have a 1973 version of the Peripherals Handbook as well as the 1976 version and they are somewhat different. If you can get both of them, you are that much better off. In addition to the aforementioned, it is also useful to have the Processor Handbook for the particular PDP 11 that you happen to have. It doesn't hurt to have processor handbooks for PDP 11's that you don't have, also. There are differences you will find between the LSI 11 and the PDP 11/70. If you take the time to read and look through some of these books, you will find all of the information that we have talked about here contained in them as well as many examples which space precludes us from putting in the magazine.

I hope this first article has been interesting to all, useful to some and boring only to a few. I apologize for its inaccuracies and look forward to your comments.
BLDCTL: Control File Processor
By David Spencer

The concept of job control files is far from new. On RSTS/E we have as control file processors both BATCH and ATPK [and now QUE.11, ed.]. These programs accept stream input files with lines similar to typing at the terminal. All this is well and good, provided that every control stream only needs to do one thing, and do it the same way every time.

BLDCTL is a TECO program written to pre-process job control files. The most unique feature of BLDCTL is its ability to ask questions during the process phase, and in doing so, mold the output stream according to the responses.

General Concepts

BLDCTL was written in TECO mainly because TECO has a rich set of text processing commands already defined within it. However, any other language would have done the job just as well.

The commands to BLDCTL are found within the source control file itself. Text processing is triggered by a set of special characters found at the beginning of the text line.

; Internal comment, will not appear in the output stream.
& Displays line to TTY, becomes a "!" in output.
?

Full format: ? [input name][help text][default min max 'code
Where:
"?" Specifies accept user input.
"↑" Indicates start of name for user input.
"name" Is replaced with user input.
"I" Ends input name, starts help text.
"'help text" Displayed when user types "?".
"J" Ends help text, starts default text.
"default" Default input when flagged.
"=" End of default value.
"min" Minimum length of user input.
"max" Maximum length of user input.
"code" BLDCTL user input edit code.

To apply default, make "min" a zero. If the user enters a "?", then the help text will be displayed.

Conditional command
.IF text = constant [or] .IF text <> constant:
.END
When found, the conditional in the " .IF " statement will be tested. If the test is TRUE, then the lines between the " .IF " and the " .END " will be included in the output stream. If the conditional is NOT TRUE, then the lines between the " .IF " and the " .END " will be excluded from the output stream.

";" operator

The semi-colon character defines an internal comment. This allows lines that are useful to see in the source (such as modification histories) to be omitted from the output. I have seen BATCH chew up many minutes processing simple comments.

"&" operator

Use of a line beginning with an ampersand will print the remaining text to the terminal. This is useful for identification of streams, warning messages, etc.

"?" operator

This command is the heart of BLDCTL. It allows questions to be asked of the user at file build time. After the input is accepted, everywhere in the control file where the prompt name is found bracketed by up-arrows is replaced by the user response. This allows dates, numbers, Y or N answers, and all sorts of input to be inserted into the control stream. In addition, there is an unimplemented parameter called code. This number can be used in modified version of BLDCTL to edit user input.

".IF, .END"

The " .IF " conditional allows blocks of code to be inserted into or deleted from the output stream. There are two forms of the conditional. The first form is a compare by a " = ", the second with a " < > ". These tests act just as they look in BASIC. If the condition is false (Y = N), then the code from that point to the closing " .END " will be deleted from the output stream. If the conditional is true, then the text remains. Please note
FINALLY, A TERMINAL FOR DEC PROGRAMMERS

THE PROTERM 80 REDUCES:

- Program development time............80%
- Disc accesses..........................90% *
- CPU time..............................66%

PUT THE TEXT EDITOR IN YOUR TERMINAL...
NOT IN YOUR CPU!

PROGRAMMING MODE:
Escape key emits $ in programming mode. These keys are lower case: % $ | & () + [ ]
80 variable length functions that can type to host, display, or type to host and display.
Samples of single key functions:
- LISTNH == OPEN ~ FOR INPUT AS FILE ¥ == CLUSTERSIZE
- PRINT ASCIAMCDS(AS, 1X, 1X) FOR I%=1% TO LENCAS) == GET $%, BLOCK $% FOR I%=1% to %=1%
PROTERM 80 enters fill in the blank mode at the ~ character. Control codes may be embedded.
Save/Send favorite phrase. Ex: RUN DB7:[213, 159]VTSDPY "U10" 'T ' Sn 'D' (' <CR> code)
Editing features include insert/delete line/character, transmit line/page, copy line.
?Fatal and %Warning messages are trapped and displayed on the bottom line of the display.
The PROTERM 80 marks the line (not line $%) that causes BASIC to generate a ?Fatal error.

If you don’t like these load your own favorites. The PROTERM 80 is a PERSONAL terminal.
The PROTERM 80 has been used with DEC BASIC’s, ROSS/V, DATA BOSS / 2, WS-11 and FINAR.

ECHO CONTROL MODE:
All the features DEC forgot. Field attributes include new/old, alpha, numeric, must fill.
Up to 64 downloadable functions. Type ahead with data compression. Operates on channel 0.
Local editing keys ‘U’, ‘rubout’, cursor left/right, delete/insert character. “Peek a boo”.
If you can use INPUT LINE and remote cursor position ALL the rest is done by the PROTERM 80.
Provides a RSTS terminal environment on ANY host, <LF>, <FF>, <ESC>, etc. can be delimiters.

ANSI MODE:
If you want just a VT-100 type terminal, push a key and operate in ANSI mode, VT-52 mode,
OR use 1-150 column widths with full editing, format protect, block mode and other features.

REDUCTIONS may be less for those that write perfect code the first time.

FOR MORE INFORMATION:
telex #
924-325
call
[617] 359-4188
or write
45 WEST ST.
MEDFIELD MASS.
02052
that spaces in the conditional text are meaningful.

Another item worth noting is that "IF" blocks can be nested within other "IF" blocks. Of course, any inner "IF" blocks will be deleted if the outer block is false.

Running BLDCTL

BLDCTL is designed to be entered by either normal run or CCL entry. The default extension on input is ".PCF" (Pseudo Control File). The output default is the common "..CTL".

On run entry, both the input file name and output file name are accepted from the keyboard. There is no default for the input file. The output file will, however, default to the input file name with the proper default extension.

For CCL entry, any CCL name will do. If only the CCL is given, then BLDCTL continues as if it were entered through RUN entry. If a filename is given, it is assumed to be the input file name and generates a default output file. Input and output file can be specified by an equals character, in the form of "=in".

Final Notes

Although BLDCTL is a complete, functioning program the current version is not intended to be the last. I openly encourage modification to the primary program. Anyone with a reasonable knowledge of TECO should be able to make changes to the code.

The examples should reasonably demonstrate in short form the capabilities of BLDCTL. About the source listing: the only control characters in the source are the two escapes at the very end of the program. These have been underlined to indicate them as such. Please don't be fooled by the alignment of the comments and don't use tabs to move out to the margin.

Any correspondence concerning BLDCTL can be sent to:

David Spencer
2901 South Sepulveda Suite 305
Los Angeles, Calif. 90064

Special offer —
A 9 track 800 bpi distribution
—send $25 check or money order to:
Dave Mallery
70 James Way
Southampton, PA 18966
A concentrated course at Saint Mary's College July 7-17, 1981.

For full details, write:
Institute for Management and Information Systems
Box 56
Saint Mary's College
Winona, Minnesota 55987

Call MACRO MAN
for superior performance
RSTS internals
custom Macro programming
RSTS or 11/M

SPECIAL OFFER
MACRO SYSTAT-V7.0
super speed, 'no' load, great features
send $40 and installation name
MACRO UTILITY LIBRARY
build your own! — call for information

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

Your intrepid Editor winged his way North again this year to visit Montreal, P.Q. the site of this year’s Canadian Decus Symposium. I met many of the friends I had seen for the first time last year, and wondered privately why I only see them at the Canadian Symposium; but then I don’t see a lot of Americans at these meetings either. Decus is Decus whether its north or south of the border and I must confess the northern branch is much friendlier than its southern neighbor.

Montreal is a delightful city filled with big city noises and smells, but small town friendliness. Bonjour is the greeting of the day and I admit to feeling somewhat inadequate if bi-lingual means I speak COBOL, BASIC, DBOL, RPG, and FORTRAN but no French. Customs is no problem (What are these magazines?????) and the proximity to the States argues for some more attendance from the south. The restaurants are... but on to the technical end of this meeting.

Most sessions were translated from English to French but not the other way around, translators were stationed in little booths in each of the rooms of the Queen Elizabeth (LE REINE ELIZABETH) Hotel. The sessions were held from Wednesday through Friday with Tuesday saved for seminars (tutorials etc). I can’t get to everything I want to hear, there is just too much. When are we going to get a timely, fully prepared proceedings and/or the availability of cassette tapes of the sessions? Here then is a distillation of my highlights of what I could hear.

A rehash of the intricate Terminal handler for RSTS. Isn’t this going to be rewritten soon anyhow? It is enlightening to see how complicated it can be just handling people typing away at keyboards, sort of reminds me of my first PDP-8 program: Accept a character and make it echo on the keyboard. Took me a whole day! Now do it using interrupts. Another day! Echo control?? Forget it.

How about VAX performance. An 11/750 peaks out at about 80 terminals due to memory limitations while an 11/780 (4MB) kept going to at least 128. If you are not CPU bound, an 11/70 did almost as well, although I don’t know where it got enough small buffers. The summation with lots of caveats was 30-50 users for the 11/750 more for the 11/780 while a RSTS system (11/70) was good if you were not CPU bound. I asked when a second Unibus would be a good idea and when a massbus would be a good idea and when a second massbus would be a good idea. They didn’t know. Didn’t make much difference they thought. I don’t believe it. A massbus has a bandwidth much larger than the unibus, and I know that my disks could saturate the unibus and I have an 11/70, the VAX must be able to swamp the unibus in a COMMERCIAL environment. I’d like to bet that there is a huge difference between a unibus VAX and a massbus one, and that by judiciously choosing when to add more you can fully utilize the CPU and the disks. Channels for disks is an IBM type word but it starts to mean something for VAX.

There were many more sessions dealing with specific RSTS applications and how professional people were supporting their customers. We ought to know by now that in order to maintain software you must have standards, and lots of other conventions that you live by. This is where you can see and hear what people have done that have made it past a few years. It is always great to be able to hear from these COMMERCIAL users. The atmosphere in Canada seems to support them better than we do in the Lower 48.

I came, I learned and I went home knowing more and feeling that the trip to Canada and Decus was one that I hope I won’t miss in the future. Good Job Decus people. Keep up the good work.
Access Control and Utilization Monitor
By C.M. Battistel and Ed Giovanella
Open Learning Institute, 7671 Alderbridge Way, Richmond, BC, Canada, V6X 1Z9

Many installations find that the access control and protection provisions provided by RSTS are often inadequate or are too inflexible to meet their needs. For example, consider the situation where we have ten users, each of whom must have access to a different combination of five programs. Using RSTS protection codes and account placement of programs and files, it is impossible to implement controlled access to these programs. Further, if the combinations of programs to be accessed by an individual were to change, the whole account, protection code structure would have to be changed. Clearly, although RSTS may provide a reasonable level of access control where there is little overlap of program access requirements, it is not adequate in most commercial environments.

Another inadequacy of RSTS exists in terms of utilization monitoring. The information provided by MONEY while useful, does not provide enough detail for many installations. It is, for example, useful to know not only who is using the system, but what programs account for the biggest load. This is especially true in an environment where charge-back and priority setting is important.

There exist instances when a program should not be run concurrently with specific other programs. An example of this is running a General Ledger Close program while another user is running a G/L data entry program. Again, we have found that RSTS and RMS do not handle this situation satisfactorily. The typical RSTS solution would be to make one program open the file ALLOW READ. However, if this is done when a program is run against the same file that opens the file ACCESS READ and ALLOW MODIFY, an error is generated. Thus we are left with having to do complex error trapping to try and control this situation.

To resolve these problems we have developed a system which not only controls access, but also monitors utilization. ACUM is an acronym for Access Control and Utilization Monitor. ACUM consists of a series of programs which restrict access to all applications programs to authorized users and monitor application program utilization rates. ACUM also includes a number of programs which provide information useful for system management. All applications programs are accessed through and return to ACUM. All applications programs also maintain utilization data during execution.

ACUM Data Base

Four files are required to accomplish the control and utilization monitoring functions of ACUM, the Access Control File, The Utilization Monitoring File, the Active Users File, and the Program Conflict File.

The Access Control File contains all of the data needed to ensure that only authorized users can gain access to the application programs and to restrict users to a specific set of programs within the entire application package.

There are 3 record types within the Access Control File. User Records (containing user I.D. number, user name and access code) define authorized users to the system. Each valid program within the applications package is defined in a Program Record (containing program code, program name and the name of the program's .TSK file). The third record type within the file is the User Clearance Record. Each of these records "link" a user to a program, thereby clearing a user for that specific program. Users may be cleared for "ALL" programs or may have several Clearance Records to authorize them for several programs.

The Utilization Monitoring File contains one record for each user-program and user-file combination. Each user-program record contains User Number, program name, CPU time, connect time, number of pages printed, and number of copies produced. The user-file records contain User Number, file name, number of records created, changed, deleted, output, screened for changes, and screened for output.

The Active Users File contains one record for each active applications job. Each record contains the User Number, the program being run, the start CPU and connect time, the keyboard number, and the RSTS job number.

The Program Conflict File contains one record for each applications program which should not be run while another specific program is active. Each record contains the name of the program which cannot be run and the name of the program it cannot be run with.

Program Access Through "ACUM"

Each user of the system is assigned a unique user I.D. number and an access code or password. This data is maintained in the Access Control File. In addition, the Access Control File also specifies the programs which a user is authorized to access. In this way, an individual user can be authorized to use only selected programs from the total applications package.

Application program access through ACUM is accomplished in the following manner. The system will automatically run the Access Control program when a non-privileged user logs onto the system. This program prompts the user to enter
HAD ENOUGH?

Outrageous Prices?
Impossible Lead Times?
Little Or No After Sale Support?

Compumation, the name that is quickly becoming synonymous with reliability & quality, now offers a complete line of D.E.C. compatible system enhancements.

The newest members of our line are our new Comp-DK & Comp-DF Backplanes, which are DD11-DK and DD11-DF equivalents, and our Comp-CK & Comp-CF which are DD11-CK and DD11-CF equivalents. Our lead time is typically 30-45 days, ARO.

SUPPORT? We offer the finest, over the phone or on site.

COMPUMATION

Call us. When you've had enough.

2326 Walsh Ave. Santa Clara CA 95051
(408) 988-8600

D.E.C. is a registered trademark of Digital Equipment Corp.
their I.D. number and access code (password). The Access Control File is checked to verify that the user has entered a valid combination. An error message is displayed if the combination is invalid and the user is re-prompted. If the user fails to enter a valid combination after three attempts, the user is logged off the system and a warning message is displayed on the system console.

If the user enters a valid I.D. number and access code, he or she is at ACUM command level and may issue commands or program requests to the Access Control program when prompted to do so. Valid user commands are "LIST" which displays all programs for which the current user is authorized, and "END" which terminates the job and logs the user off the system. If the user enters a program request, the Access Control File is checked to verify that the program exists and that the current user is authorized to use it. If either edit fails, an error message is displayed and the user is re-prompted. If the user enters a valid program request, the Access Control program places the user I.D. number in core common, updates the Active User File to reflect the status of the user, checks for conflicts with currently active programs and if no conflict exists, CHAINS to the appropriate application program. When the user is finished with the application program, they are CHAINED back to the Access Control program and the Active User File is updated to reflect their current status. Users are then re-prompted for a command or program request. If command "END" is entered, the Active User File record corresponding to the user and job is deleted, and the user is logged off the system. It can be seen from the above description that during a job session, a user is always in communication with an application program and at no time can they gain access to RSTS. For the convenience of application programmers, it is possible to clear a user to run a program which allows access to RSTS.

Two other points of interest concerning ACUM must also be mentioned. The first statement of each application program is a CALL to a subroutine which immediately logs the user off the system. The Access Control program "CHAINS" to application programs at a point past the subroutine CALL. It is, therefore, impossible to access any application program directly without going through the Access Control program. The second point of interest is the passing of the user I.D. number into core common. Each application program can then obtain the user I.D. from core common and thereby maintain utilization statistics for that user.

All programs must chain to the Access Control program upon termination. This enables users to select additional application programs for execution or issue the "END" command which automatically logs the user off. In cases where a job detaches and a new job is created, LOGIN has been modified to chain to the Access Control program. The detached job upon termination kills itself. LOGIN chains to the Access Control program by running a program called START in the user's account. If there is no START program in the user's account the user is logged into RSTS command level. This feature is used to enable programming staff immediate access to RSTS without using the Access Control program.

---

**25% SAVINGS TO LARGE LESSEES OF**

**DEC TERMINALS**

<table>
<thead>
<tr>
<th>OUR PRICE</th>
<th>Typical Competition</th>
</tr>
</thead>
<tbody>
<tr>
<td>LA120</td>
<td>$94/mo.</td>
</tr>
<tr>
<td>LA34</td>
<td>$44</td>
</tr>
<tr>
<td>VT100</td>
<td>$64</td>
</tr>
</tbody>
</table>

- NEW EQUIPMENT
- INCLUDES DEC LOCAL ZONE MAINTENANCE
- PRICES BASED ON 25 UNIT SINGLE BATCH NEW INSTALLATIONS FOR ONE YEAR
- PURCHASES AVAILABLE AT SIMILAR DISCOUNT

**DACOTERM, INC.**

(313) 229-7850
11365 CULVER ROAD
BRIGHTON, MICH. 48116 453-7250
Utilization Monitoring

The Utilization Monitoring Function of ACUM is accomplished through the tracking of user system utilization within each application program. The utilization data must be maintained from the time that the Access Control program chains to an applications program, until the job kills itself. Although entry to the Access Control program or in the case of a job which detaches, until the job kills itself. Although entry to an applications program can only be made through the Access Control program, exit from applications programs can take a number of forms.

1. Use of ESC key during data entry
2. KB wait exhausted during data entry
3. 'normal' exit from mainline by chaining to the Access Control program
4. exit from mainline by chaining to QUE

The method of collecting utilization data remains constant regardless of exit mode; however, the method of updating utilization data stored in ACUM files will vary.

In order to properly maintain utilization data while programs are executing the following functions must be performed at the beginning of each applications program.

1. Declare a common area with the following data:
   a. ACUM user number (the Access Control Program places the user number in core common)

   b. number of data files
   c. current CPU time
   d. current connect time
   e. 6 character program name
   f. 6 character file name of data files used in program

2. During program execution the program must maintain in the common area the following data:
   a. number of pages of hardcopy output (handled by subroutines)
   b. number of copies of hardcopy output (handled by subroutine)
   c. number of file records created in each file
   d. number of file records changed in each file
   e. number of file records deleted in each file
   f. number of file records output from each file
   g. number of file records read and reviewed from each file in an attempt to find a specific record which is to be changed
   h. number of records read from each file in order to find a specific sub-set of the file which is then output.

The utilization data maintained by each applications program is written to the Utilization Monitoring File by a subroutine prior to program termination. The method used for update varies with the manner in which the program ended.
Program Conflict Control

Since ACUM knows at all times which application programs are running, it can control the execution of conflicting programs. To accomplish this ACUM maintains a table containing the names of all programs which conflict with each other. This table is checked by the Access Control program prior to chaining to a user selected applications program.

Prior to implementing an applications program which conflicts with other application programs, programmers create the appropriate conflict records in the Program Conflict File as follows:

<table>
<thead>
<tr>
<th>USER-SELECTED-PROGRAM</th>
<th>CONFLICTING-PROGRAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROGA</td>
<td>MPROP</td>
</tr>
<tr>
<td>PROGA</td>
<td>PROGER</td>
</tr>
<tr>
<td>PROGA</td>
<td>TSTPRG</td>
</tr>
</tbody>
</table>

If the conflict works both ways for all these programs the following additional records would be required:

| MPROP                  | PROGA                |
| TSTPRG                | PROGA               |

When a user selects a program, the Access Control program checks the Program Conflict File for a matching USER-SELECTED-PROGRAM. For each one found, the corresponding CONFLICTING-PROGRAM field is used to search the Active Users File. If a matching record is found in the latter file an error message is displayed to the user, and he or she is prevented from executing the USER-SELECTED-PROGRAM.

ACUM Management Reports

Two sets of reports are available to inform management as to the current status of application programs, users and computer utilization. One set of reports is printed from the Access Control File and lists all application programs and authorized users. The other set is produced from the Utilization Monitoring File and provides statistics on data file activity and computer resource utilization. These reports are described in detail below.

Access Control File Listings

There are three simple listings that can be produced from the Access Control File. These reports show the current status of users, user clearances, and application programs. The reports can be produced either at a keyboard or on the line printer.

a. Application Programs: lists each application program available through ACUM and shows the program code, program description, and the name of the program's .TSK file.

b. Authorized Users: lists each user cleared to access ACUM and shows user I.D. number, name and access code or password.

c. Authorized Users and Clearances: lists each user with access to ACUM and the application programs for which the user is cleared. The report shows user I.D. number, user name, password, program code and program description. Appropriate messages are printed if the user is cleared for all programs or not cleared for any programs.

Computer Resource Utilization Summaries

There are four summary reports that can be provided from the Utilization Monitoring File. These reports contain a variety of utilization statistics including file activity, CPU and connect time usage and volume of hardcopy output. They are designed to assist management in monitoring the utilization of application data files and computer resources and can be a valuable aid in detailing utilization by individual users, defining through-put "bottle-neck" areas, planning for future hardware and/or software enhancements and monitoring file activity to determine file volatility and storage requirements.

The reports can be displayed at a terminal or printed on the printer.

a. File Activity By User: This report analyzes file activity by individual user. Six activity categories (number of records created, changed, deleted, displayed, screened for change and screened for display) and a total activity count is printed for each user of the data file. The report also shows the percentage of each activity category as it relates to the total activity by each user. A total line is also printed for each file showing the total activity and related percentages for all users of the file. The program that produces the report can be instructed to print data for all files or one file.

b. Computer Utilization By User: This, and the next two reports display the computer resource utilization statistics in a variety of ways. This report organizes the data by user number and displays connect time, CPU time, number of pages and copies printed and total paper used. The program also calculates and displays, for each user, the percentage of each utilization category as it relates to the total of that category.

c. Computer Utilization By Program: This report is identical in content to the one "By User" but is presented in sequence by Program Name.

d. Computer Utilization By User and Program: This report organizes the connect and CPU time statistics by user and program. For each user, the user number and user name is printed. For each of the user's programs, the program name, description, connect time and CPU time is displayed. The percentage of each program's connect and CPU times (as it relates to the total times for the user) is calculated and printed.

CONCLUSIONS

The above description represents the results of our attempts to provide access control and utilization monitoring capabilities beyond those provided by RSTS. We have found the system extremely adaptable and easy to use and maintain.
PDP/11 Systems — UTILITIES
By M. A. Jackson, A.C. Nielsen Co. Ltd., Oxford, UK

INTEGER TO DECIMAL STRING CONVERSION SUBROUTINE

Invocation
CALL NUMASK (A$, B%)

Processing
This module right-aligns in A$ the decimal string representation of B%. Only as many characters as are necessary to hold the value are changed; other characters in A$ are unchanged. If B% equals zero, no action is taken, i.e. the routine will not right-align a single zero character into a blank string.

Checks
1. That two parameters are passed. — No action taken.
2. That B% is not negative. — No action taken.
3. That A$ is long enough to hold the string value of B%. — No action taken.

Critical considerations
Size 108 bytes
Speed <1ms

FINALLY! A publication exclusively for DEC* or DEC-related users, manufacturers, and consultants! Order your free subscription TODAY!

FREE Subscription Form

Name ____________________________
Title ____________________________
Company _________________________
Company Address __________________
City/State/Zip ____________________
Phone( ) _________________________
Signature _________________________
Date _____________________________

☐ Check here if you wish to receive promotional mail from Consultant Educator

IMPORTANT — To qualify, please answer all questions below.

Type of DEC computer at this location:
CPU ____________________________ Operating System __________________
Disk ____________________________ Tape ____________________________
Floppy ____________________________ Printer _________________________
CRT ____________________________ Other ____________________________

Primary product or service of your firm at this location:
End User ____________________________ OEM (type) __________________
System House _________________________ MFG ______________________
Consultant ____________________________ Educator __________________
Data Processing ______________________ Government __________________

HARDCOPY is published six times per year by Seldin Publishing, Inc. One subscription per DEC site at no charge.
One year subscription for other than qualified individuals may be purchased at the following rates:

United States — Surface Mail ........................................ $10.00
United States — Air Mail ........................................... $17.00
All Other Countries .............................................. $19.00

SEND TO HARDCOPY BOX 759 — BREA, CALIFORNIA 92621
.TITLE NUMASK
.IDENT /00/

; CALL NUMASK(A$,B$)

; THIS SUBROUTINE IS CALLED FROM BASIC-PLUS-2. IT RIGHT-ALIGNS THE
; DECIMAL REPRESENTATION OF B$ IN A GIVEN MASK STRING A$. IT IS A NO-OP
; IF B$ IS NEGATIVE OR ITS DECIMAL REPRESENTATION WILL NOT FIT INTO A$.

NUMASK:
CMP $2,(R5) ; 2 ARGUMENTS ?
BNE RETRN ; NO - GIVE UP
MOV @4(R5),R4 ; R4 = B$
BLT RETRN ; NEGATIVE - GIVE UP
MOV 2(R5),R0 ; R0 = ADDR OF A$ HEADER
MOV 2(R0),R1 ; R1 = LENGTH OF A$
BEQ RETRN ; A$ NULL - GIVE UP
MOV #POWERS,R2 ; START R2 AT 10000
CLR: CLR R3 ; INIT COUNTER
SUBT: SUB (R2),R4 ; SUBTRACT POWER OF 10
BLT NEG ; OVERDONE IT
INC R3 ; INCREMENT CNTR
BR SUBT ; GO SUBTRACT AGAIN
NEG: ADD (R2),R4 ; ADD IT BACK IN
CMP 2(R0),R1 ; FIRST CHAR ?
BNE PUSH ; NO - NEED IT WHATEVER
TST R3 ; IS IT ZERO ?
BEQ NEXT ; YES - BYPASS PUSH
PUSH: MOV R3,-(SP) ; PUSH DIGIT ONTO STACK
DEC R1 ; COUNT DOWN NUMBER OF CHAR
BLT OVERFL ; TOO MANY CHAR
NEXT: TST (R2)+ ; POINT R2 AT NEXT (LOWER) POWER
TST (R2) ; CHECK END OF POWERS
BNE CLR ; MORE TO SUBTRACT
MOV (R0),R2 ; R2 = ADDR OF A$
ADD 2(R0),R2 ; R2 = ADDR OF END OF A$ + 1
FILL: CMP 2(R0),R1 ; ALL DIGITS POPPED ?
BEQ RETRN ; YES - GO HOME
BIS $000060,(SP) ; CUT TO ASCII DECIMAL
MOVB (SP)+,-(R2) ; MOVE DIGIT TO STRING
INC R1 ; COUNT CHAR POPPED
BR FILL ; GO CHECK FINISHED
OVERFL: CMP 2(R0),R1 ; ALL DIGITS POPPED ?
BEQ RETRN ; YES - GO HOME
TST (SP)+ ; 'POP A DIGIT' (TO NOWHERE)
INC R1 ; COUNT CHAR POPPED
BR OVERFL ; GO CHECK FINISHED
RETN: RETURN
POWERS: .WORD 10000,1000,100,10,1}

.END
VERIFY SUBROUTINE

Invocation
CALL VERIFY (A$, B$, C%)

Processing
This module returns in C% the position of the first character in B$ which is not in A$, or zero if all characters in B$ are in A$. It also returns zero if B$ is null, even if A$ is also null. This is logical (there are no characters in B$ which are not in A$) and consistent with Basic-plus-2 functions.

Checks
1. That three parameters are passed. — No action taken.

Critical Considerations
Size 70 bytes
Speed Typically <1ms; dependent on the string lengths.

A$ is searched sequentially from the left; characters expected to be most frequently present in B$ should therefore be at the left of A$.

```
.TITLE VERIFY
.IDENT /00/

; CALL VERIFY(A$,B$,C%)
; SUBROUTINE TO RETURN THE POSITION OF THE FIRST CHARACTER IN B$
; WHICH IS NOT IN A$, RETURNS 0% IF ALL BYTES ARE PRESENT.

VERIFY:
    CMPB  #3,(R5)            ; 3 ARGUMENTS ?
    BNE   RETRN             ; NO - GIVE UP
    CLR   @6(R5)            ; INIT C% = 0%
    MOV   4(R5),R0          ; R0 = ADDR OF B$ HEADER
    MOV   2(R0),R1          ; R1 = LENGTH OF B$
    BEQ   RETRN             ; B$ NULL - RETURN 0%

TESTA:
    MOV   2(R5),R2          ; R2 = ADDR OF A$ HEADER
    MOV   2(R2),R3          ; R3 = LENGTH OF A$
    BNE   START             ; NOT ZERO LENGTH
    INC   @6(R5)            ; C% = 1%
    BR    RETRN             ; A$ NULL - MUST RETURN 1%

START:
    MOV   (R0),R0           ; R0 = ADDR OF B$
    ADD   R0,R1             ; R1 = ADDR OF END OF B$ + 1
    ADD   (R2),R3           ; R3 = DITTO A$

NEXTA:
    MOV   (R2),R4           ; ADDR OF A$
    INC   @6(R5)            ; INCREMENT C%

TESTB:
    CMPB  (R0),(R4)+        ; COMPARE BYTE OF B$ WITH BYTE OF A$
    BEQ   NEXTB             ; IT'S THERE; GO GET NEXT OF B$
    CMP   R4,R3             ; FAST ALL OF A$ ?
    BNE   TESTB             ; NO - TEST NEXT
    BR    RETRN             ; YES - C% ALREADY SET - RETURN

NEXTB:
    INC   R0                ; POINT TO NEXT OF B$
    CMP   R0,R1             ; FAST ALL OF B$ ?
    BNE   NEXTA             ; NO - TEST NEXT OF B$
    CLR   @6(R5)            ; YES - SET C% = 0%

RETN:
    RETURN
.END
```
INTEGER TO OCTAL STRING CONVERSION SUBROUTINE

Invocation
CALL OCTAL (A$, B%)

Processing
This module returns in the six-byte string A$ the octal string representation of B%.

Checks
1. That two parameters are passed. — No action taken.
2. That A$ is six bytes long. — No action taken.

Critical considerations
Size 62 bytes
Speed <1 ms

```
TITL E OCTAL
IDENT /01/

; CALL OCTAL(A$,B%)
;
; THIS SUBROUTINE PUTS THE SIX CHARACTER OCTAL REPRESENTATION OF B%
; INTO THE (SIX-BYTE) STRING A$
;
OCTAL:

CMPB (R5),$2  ; 2 ARGUMENTS ?
BNE ERR      ; NO = ERROR
MOV 2(R5),R0  ; R0 = ADDR OF A$ HEADER
CMP 2(R0),$6  ; A$ SIX BYTES ?
BNE ERR      ; NO = ERROR
MOV @4(R5),R1  ; R1 = B%
MOV (R0),R2  ; R2 = ADDR OF A$
ADD $6,R2  ; R2 = ADDR OF END OF A$ + 1
MOV $5,R3  ; LOOP COUNT
LOOP:

MOV R1,+(SP)  ; WORD ONTO STACK
BIC $1777770,(SP)  ; ONE OCTAL VALUE'
ADD $'0,(SP)  ; CONVERT TO ASCII
MOVB (SP)+,-(R2)  ; STORE IN A$
ASH $-3,R1  ; SHIFT RIGHT THREE
SUB R3,LOOP  ; TEST FOR FIVE DONE
BIC $1777776,R1  ; GET LAST BIT
ADD $'0,R1  ; CONVERT TO ASCII
MOVB R1,+(R2)  ; STORE IN A$

ERR: RETURN
.END
```
SUBSTRING PSEUDOVARIELABLE EMULATION SUBROUTINE

Invocation

CALL INSERT (A$, B$, C%)

Processing

This module overwrites string B$ into string A$ starting at position C%.

Checks

1. That three parameters are passed. — No action taken.
2. That C% is greater than zero. — No action taken.
3. That B$ will fit into A$ from that position. — No action taken.

Critical consideration

Size 50 bytes
Speed <1ms

```assembly
.TITLE INSERT
.IDENT /00/

; CALL INSERT(A$,B$,C%)

; SUBROUTINE TO OVERWRITE B$ INTO A$ STARTING AT POSITION C%

INSERT:

CMPB $3, (R5) ; 3 ARGUMENTS ?
BNE RETRN ; NO - GIVE UP
MOV $6, (R5), R2 ; R2 = C%
BLE RETRN ; C% <= 0% - GIVE UP
MOV 2(R5), R0 ; R0 = ADDR OF A$ HEADER
MOV 4(R5), R1 ; R1 = ADDR OF B$ HEADER
MOV 2(R1), R4 ; R4 = LENGTH OF B$
BEQ RETRN ; B$ NULL - NOTHING TO DO
DEC R2 ; R2 = C% - 1%
MOV R2, R3 ; R3 = C% - 1%
ADD R4, R3 ; R3 = C% - 1% + LEN(B$)
CMP R3, 2(R0) ; GREATER THAN LEN(A$) ?
BGT RETRN ; YES - B$ WON'T FIT - GIVE UP
MOV (R1), R1 ; R1 = ADDR OF B$
MOV (R0), R0 ; R0 = ADDR OF A$
ADD R2, R0 ; R0 = ADDR OF 1ST CHAR TO OVERWRITE
MOVONE: MOVB (R1)+, (R0)+ ; MOVE A CHAR FROM B$ INTO A$
SDB R4, MOVONE ; GO MOVE ANOTHER IF NOT FINISHED

RETN: RETURN
.END
```
STRING INSERTION SUBROUTINE

Invocation

CALL PUSHIN (A$, B$, C%)

Processing

The module writes string B$ into string A$ starting at position C%. Characters in A$ from position C% onwards will be moved up to follow the inserted characters.

Checks

1. That three parameters are passed.
2. That C% is greater than zero
3. That B$ will fit into A$ from that position.
4. That sufficient blanks or nulls are present at the end of A$ to be "pushed off the end". — No action taken if any fails.

Critical Considerations

Size 80 bytes

Speed <1ms

```
; TITLE PUSHIN
; IDENT /00/
; ; CALL PUSHIN(A$,B$,C%)
; ; THIS SUBROUTINE INSERTS B$ INTO A$ STARTING AT POSITION C%,
; ; PUSHING THE REMAINDER OF A$ TO APPEAR AFTER B$.
; ; A$ MUST HAVE SUFFICIENT BLANKS OR NULLS AT THE END TO ACCOMMODATE B$.
; ; PUSHIN:
; CMPB (R5),#3 ; 3 PARMS?
; BNE RETRN ; NO - GIVE UP
; MOV @6(R5),R0 ; R0 = C%
; BLE RETRN ; C% <= 0% - GIVE UP
; MOV 4(R5),R1 ; R1 = ADDR OF B$ HEADER
; MOV 2(R1),R2 ; R2 = LEN(B$)
; BEO RETRN ; B$ NULL - NOTHING TO DO
; MOV 2(R5),R3 ; R3 = ADDR OF A$ HEADER
; MOV 2(R3),R4 ; R4 = LEN(A$)
; ADD R2,R0 ; POSN PAST END OF INSERT
; DEC R0 ; POSN OF LAST INSERTED CHAR
; CMP R0,R4 ; IS THIS PAST END OF A$?
; BGT RETRN ; YES - GIVE UP
; ADD (R3),R4 ; R4 = ADDR OF BYTE PAST A$
; MOV R4+R0 ; STORE DITTO IN R0
; TEST: BITB -(R4),#137 ; IS IT BLANK OR NULL?
; BNE RETRN ; NO - GIVE UP
; SOB R2,TEST ; TEST AS MANY BYTES AS IN B$
; MOV 2(R3),R2 ; R2 = LEN(A$)
; SUB @6(R5),R2 ; R2 = LEN(A$) - C%
; INC R2 ; NO. OF BYTES TO BE PUSHED UP
; PUSH: MOVB -(R4),-(R0) ; MOVE BYTES OF A$ AFTER INSERT
; SOB R2,PUSH ;
; MOV 2(R1),R2 ; R2 = LEN(B$)
; MOV (R1),R4 ; R4 = ADDR OF B$
; ADD R2,R4 ; R4 = ADDR OF BYTE PAST B$
; ADD R2,R0 ; R0 = ADDR PAST LAST INSERT
; INSERT: MOVB -(R4),-(R0) ; MOVE BYTES OF B$
; SOB R2,INSERT ;
; RETRN: RETURN
; END
```
BACmac can do it all!

BAC into RTS / BAC into MAC / BAC into BAS

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

- translation from Basic-Plus "compiled" back to Basic-Plus source code (only the comments will be missing)
- translation from Basic-Plus into Macro source code, which compiled under RSTS runs faster than Basic-Plus
- translation from Basic-Plus into Macro source code which may be compiled under RSTS for execution under RT11 — a migration facility
- translation from Basic-Plus into a RUN-TIME-SYSTEM. Now you can write an RTS in Basic-Plus. The ideal solution to memory thrashing due to “multi-copy” applications programs.

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

Please write for more information

STRING INSERTION SUBROUTINE

Invocation

CALL PUSHON (A$, B$, C%)

Processing

The module writes string B$ into string A$ starting at position C%. The character in A$ at position C% is overwritten; characters from position C% + 1% onwards will be moved up to follow the inserted characters.

Checks

1. That three parameters are passed.
2. That C% is greater than zero.
3. That B$ is not null.
4. That B$ will fit into A$ from that position.
5. That sufficient blanks or nulls are present at the end of A$ to be "pushed off the end".

No action taken if any fails.

Notes

1. Check 3 implies that a character cannot simply be "lost" from A$ by specifying replacement of the character at position C% with a null string.
2. That routine may be used simply to overwrite a single character in A$ (similar to INSERT subroutine) in which case check 5 obviously does not apply.

Critical Considerations

Size 82 bytes

Speed <1ms.
.TITLE FUSION
.IDENT "ROGY\n
# CALL FUSION(A$,B$,C$)

# THIS SUBROUTINE INSERTS B$ INTO A$ STARTING AT POSITION C$,
# OVERWRITING THE CHARACTER AT POSITION C$ AND
# PUSHING THE REMAINDER OF A$ TO APPEAR AFTER B$.
# A$ MUST HAVE SUFFICIENT BLANKS OR NULLS AT THE END TO ACCOMMODATE B$.

FUSION:

CMP B$ (R5)$+ ; 3 PARRS ?
BLT RETRN ; NO - GIVE UP
TEST (R5)$+ ; R5 = ADDR OF 1ST PARM
MOV (R5)$+ ; R3 = ADDR OF A$ HEADER
MOV (R5)$+ ; R1 = ADDR OF B$ HEADER
MOV 2(R1)$+R2 ; R2 = LEN(B$)
BEQ RETRN ; B$ NULL - NOTHING TO DO
MOV 2(R5)$+R4 ; R4 = LEN(A$)
MOV (R5)$+R5 ; R5 = ADDR OF C$
MOV (R5)$+R6 ; R6 = C$
BLE RETRN ; C$ <= 0% - GIVE UP
ADD R2$+R0 ; POSN PAST END OF INSERT
DEC R0 ; POSN OF LAST INSERTED CHAR
CMP R0$+R4 ; IS THIS PAST END OF A$ ?
BEQ RETRN ; YES - GIVE UP
BGT RETRN ; STORE LEN(B$) IN R0
ADD (R3)$+R0 ; R4 = ADDR OF BYTE PAST A$
DEC R2 ; ALLOW ONE CHAR OVERWRITE
BEQ NOTEST ; ONLY ONE BYTE - NEED NO SPACE
TEST: BITB -(R4)$+137 ; IS IT BLANK OR NULL ?
BNE RETRN ; NO - GIVE UP
SBE R2$+TEST ; TEST AS MANY BYTES AS IN B$ MINUS ONE
NOTEST: MOV R0$+R2 ; R2 = LEN(A$)
ADD (R3)$+R0 ; R0 = ADDR OF BYTE PAST A$
MOV 2(R1)$+R3 ; R3 = LEN(B$)
SUB (R5)$+R2 ; R2 = LEN(A$) - C$
SUB R3$+R2 ; R2 = LEN(A$) - C$ - LEN(B$)
INC R2 ; NO. OF BYTES TO BE PUSHED UP
BEQ NOPUSH ; NONE TO MOVE
PUSH: MOV B$ -(R4)$-(R0) ; MOVE BYTES OR A$ AFTER INSERT
SBE R2$+PUSH ;
NOPUSH: MOV (R1)$+R4 ; R4 = ADDR OF B$
ADD R3$+R4 ; R4 = ADDR OF BYTE PAST B$
INSERT: MOV B$ -(R4)$-(R0) ; MOVE BYTES OF B$
SBE R3$+INSERT ;
RETN: RETUR
AMERICAN USED COMPUTER
P.O. Box 68, Kenmore Station, Boston, Massachusetts 02215

Leaders in Used DEC Hardware Since 1968.

Chapter 5

Programming Examples.

This section contains a runthrough of routines to read and write to a device. The development is historical: one day I read an article by Bob Meyer in the May/June issue (1980) of the RSTS Professional: the results are before you. At the end is an example of a subroutine which creates a file, to show how the important directive CALFIP is used.

5.1 PRINT Routine — Version 1.

The original program in the article I read looked like this:

```
.TITLE PRINT
.END

.ASCII /These are the voyages ...
.DIV

MOVL = . . . MOPL

; message length = this location -
; start of buffer location

LOOP : MOV XOFF, 10
       MOV XOFF, (XOFF)+
       MOV XOFF, (XOFF)+
       MOV XOFF, (XOFF)+
       MOV XOFF, (XOFF)+
       MOV XOFF, (XOFF)+
       .WRITE
       .EXIT

; global follow

PRINT: MOV MOPL, 5
       MOV MOPL, (MOPL)+
       MOV MOPL, (MOPL)+
       MOV MOPL, (MOPL)+
       MOV MOPL, (MOPL)+
       MOV MOPL, (MOPL)+
       .WRITE
       .EXIT

; global follow

MOVL = 080<52
.END = 104064
.EXIT = 104046

.DATA SPACE

; this program loads up the XRB with
; the information it needs, and goes and does its .WRITE
; and then goes home.

As you can see, this program loads up the XRB with the information it needs, and goes and does its .WRITE, and then goes home.

5.2 PRINT Routine — Version 2.

What might we reasonably want this PRINT routine to do? First, we should want to re-use it, so it has to go into subroutine form. Second, everything in version 1 is specified too exactly — we have to operate on channel 0, and print block 0, and make it whatever is in OBUFF, etc. So the second program should access memory locations which contain all this information. We could, of course, have it access registers, but if we wrote all our subroutines with such gay abandon, we would never be satisfied with a paltry 8 registers at all. A re-write of version 1 might well look like this:

```
.TITLE PRINT
ENDIF /MOPL/;

; PRINT subroutine, substitutes MSGCS
; PRINT statement.

parameters:

FNNLEN = containing length of buffer to move.
CHLCH = channel + 2 to print to.
BCHLCH = block to print to.
MOPL = containing start address
       of output buffer.

; routine code follows

PRINT: MOV MOPL, MSGCS
       MOV MOPL, (MOPL)+
       MOV MOPL, (MOPL)+
       MOV MOPL, (MOPL)+
       MOV MOPL, (MOPL)+
       MOV MOPL, (MOPL)+
       .WRITE
       .EXIT

; routine code follows

MOVL = 080<52
.END = 104064
.EXIT = 104046

.DATA SPACE

; this program runs the PRINT
; subroutine once.

.ASCII /These are the voyages ...
.DIV

MOPL = . . . MOPL

; message length = this location -
; start of buffer location

LOOP : MOV XOFF, 10
       MOV XOFF, (XOFF)+
       MOV XOFF, (XOFF)+
       MOV XOFF, (XOFF)+
       MOV XOFF, (XOFF)+
       MOV XOFF, (XOFF)+
       .WRITE
       .EXIT

; global follow

MOVL = ASCII /These are the voyages ...</cr></lf>

; data follows.

BLOC1: ASCII (nl)</nl>

; reserve 2 bytes of 0

.TEST: MOV XOFF, 10
       LOOP: TYPH (XOFF)+
            BNE LOOP
            MOV XOFF, (XOFF)+
            MOV XOFF, (XOFF)+
            MOV XOFF, (XOFF)+
            MOV XOFF, (XOFF)+
            MOV XOFF, (XOFF)+
            .WRITE
            .EXIT

; global follow

MOVL = ASCII (nl)</nl>

; global follow

MOVL = ASCII (nl)</nl>

; this program runs the PRINT
; subroutine once.

.ASCII /These are the voyages ...
.DIV

MOPL = . . . MOPL

; message length = this location -
; start of buffer location

LOOP : MOV XOFF, 10
       MOV XOFF, (XOFF)+
       MOV XOFF, (XOFF)+
       MOV XOFF, (XOFF)+
       MOV XOFF, (XOFF)+
       MOV XOFF, (XOFF)+
       .WRITE
       .EXIT

; global follow

MOVL = ASCII (nl)</nl>

; data follows.

BLOC1: ASCII (nl)</nl>

; reserve 2 bytes of 0

.TEST: MOV XOFF, 10
       LOOP: TYPH (XOFF)+
            BNE LOOP
            MOV XOFF, (XOFF)+
            MOV XOFF, (XOFF)+
            MOV XOFF, (XOFF)+
            MOV XOFF, (XOFF)+
            MOV XOFF, (XOFF)+
            .WRITE
            .EXIT

To assemble and run this you would command

macro print, print = print
mac test, test = common, test
trk test, test = test, print
run test

[Note: COMMON.MAC can be found on the SYSGEN tape.]

5.4 Creating a File.

This last example shows a subroutine which creates a file. There are a few new ideas and instructions used, but everything is pretty straightforward:

1. Macro procedures — it becomes repetitive to write a similar piece of code several times, so macro procedures are employed. When the assembler finds a statement calling a macro procedure, it inserts the procedure code in place of the call, and replaces the dummy variables in
DBL for RSTS

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

Some of the features offered with DBL under RSTS:

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

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

<table>
<thead>
<tr>
<th>Quantity</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$4,200.00</td>
</tr>
<tr>
<td>2 - 10</td>
<td>$2,940.00</td>
</tr>
<tr>
<td>11 - 25</td>
<td>$2,520.00</td>
</tr>
<tr>
<td>26 +</td>
<td>$2,100.00</td>
</tr>
</tbody>
</table>

Prices effective February 1, 1981

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

DISC

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

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

DBL and DISC are registered trademarks of Digital Information Systems Corporation.
the definition with the passed parameters in the call. Dummy variables preceded by a question
mark have special meaning: if the parameter is not supplied, they become a unique local symbol
between 64$ and 127$.

2. .PSECT — this instruction controls linking. In this case it is used to tell TKB that this piece of
code is pure instructions.

3. The stack — to use registers freely inside a

4. Local symbols — these are of the form
<number>$. They are local to the .PSECT of
code they appear in, and insure that separately
assembled pieces of code do not conflict.

To open a file for output, you have to load up the
firqb with the file name in RADSO format, plus info like
the disk and ppn, mode, file size, etc. This is done by
making the file string scan call. It automatically sets up
most of the information to do the open. The code looks
like this:

```
.title create a file (if possible)
.objct /BIG216/
.

This module is a subroutine to create a file. It is called by

JBR pe, create
.

It expects to see the address of the file name in r6 and the channel
to open in r1. It returns errors in location FIRQB.
.

.upper clear

.ascii clear loc, len, req1, req2, plo, clear len bytes of memory

.4ops include, req2

.ofs starting at loc.

.clr (req1),*

.add req5, a

.endm clear

.ssect create, 1
.

create:

.nov r7, -160

.nov r1, -160

.nov r2, -160

.nov r3, -160

.clr firqb, 4p/3, r7, r3

.clr firc, 4p/3, r7, r3

.nov r6, xre=xrloc

.clr r7, r3

.label fircb, (rc)

.bcc dc

.ind r5

.br 10

.nov r7, xre=xrloc

.nov r5, xr=xc

.fas

.test fircb

.bcc 39

.clear xru, 20/7, r2, r3

.nov (ercf, fircb+frcf)

.add r1, +1

.nov r1, fircb=0xf11

.callfip

.try to create a file

.nov (up+), r3

.nov (up+), r2

.nov (up+), r1

.nov (up+), r0

.rts pc

.end create

.end of test
```

What is going on is really pretty simple. We save the
registers, find the length of the file name string, set up
the xrb, and see if the name is legal. If it is, we set up the
firqb and open the file. The only thing which should
seem strange is the ASH instruction. It stands for
Arithmetic SHift. This one means shift all the bits in r1
left by 1. Bit 15 goes into the C bit. As the procedure
dies, it restores the values of the registers it changed to
their original contents. The calling program decides if
the file was actually created by testing location FIRQB.

5.5 Last comments. People tend to show off what they
know (after all, why else write articles?). In the case of
assembly language programming, DON'T DO IT! There
are only a few times you will really need to program
something in MACRO-11: some i/o, or a piece of code
that has to be really fast or small. Remember that the
cost for this is that development time will skyrocket,
and the resulting code will be difficult to maintain—so
don't plan on writing an accounts receivable package in
assembly language, for you will still be at it when the
company folds.

A final note on tools: there is a very good debugging
tool called ODT. It allows you to establish breakpoints in
your code, look at memory locations, change them and
continue onward. Many people also do the same sorts of
things from a higher level language, like Pascal or BASIC
Plus 2. I think which approach you use is mostly a mat­
ter of temperament, but that before too long, you'll find
you will need some tool to work with.

**Appendix A**

**COMMON.MAC**
We need space and time! Need 1170 RSTS time and office space in Southern California evening and weekend use! Will pay ¾ of your lease costs. Call Allen (213) 394-3857.

RSTS/E BASIC +, DIBOL Programmers call or send resume to Scientific Placement, Inc., P.O. Box 19949, Houston, TX 77024; (713) 496-6100.

 MEDIA TRANSFER SERVICES - We can transfer your data between any of the following media: RK05, RK06, RK07, RX01, and 800 BPI 9 Track MT. We handle most DIGITAL directory structures on a fast turn around basis and also offer programming services for any special transfer problems you may have. Call (201) 584-4026 Advanced Digital Data, 354 Route 206, Flanders, New Jersey 07836. Authorized DIGITAL DISTRIBUTOR.

FORT/RTRAN Software Design and implementation at reasonable cost. RSTS, RT-11, and RSX are our specialty. Golden Triangle Computers, Box 1866, Orange, TX 77630, (713) 883-4588.

Now paying $5050 for used DEC RK07 drives. Call Alvin at (714) 732-0782.

Australian DEC specialists. Call (02)4766820.

Gary Oppenheimer-Consulting (9 years experience with RSTS & RSTS/E). N.Y.C. area (212) 787-2416.


For Sale: Systems Industries Model 9500 SMD Controller complete with cables, manual $3,500 30 day warranty. Consolidated Funding Corporation, 2361 W. Devon, Elk Grove, IL 60007 (312) 766-1400.


RSTS/E freelance work in San Diego, call Gary (714) 263-7321.

How-to COPYRIGHT SOFTWARE. $8.50 to Sofprotex, Box 271-P, Belmont, CA 94002.

Protect your PDP11 from damage with Nassau Systems over temperature protection unit, Box 19329, Cincinnati, OH 45219.

$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

H&R for electronics. 401 E. Erie Avenue, Philadelphia, PA 19134, request catalog.

Connect your office to the future with cableshare’s inter-office communication system. Electronic Mail, Word Processing, Electronic Filing, Message Switching, and more for RSTS users.

DEC systems and components - buy and sell, Western Data Sales, Inc., 23520 Telo St, #12, Torrance, CA 90503, (213) 539-8917.

Systems Analysis, Design and Programming, RSTS and many other systems, Phone OIC, (213) 506-8700.

RSTS/E Medical Billing Programs, Pete Johnson (505) 255-3777.

We salvage all kinds of disasters:
• unreadable disks
• ruined UFDs and MFDs repaired
• immediate response
• telephone DIAL-UP
• on-site
• software tools
• custom recovery

Brought to you by On Track Systems, Inc. and a well known (and read) Disk Directory expert.

Page 92 March 1981
GENERAL ACCOUNTING

MCBA applications A/P, G/L, P/R, A/P, INV including source. 750 each

David Talbot
Interstate Data Systems
214 West Main
Urbana, IL 61801
(217) 337-1144

FOR SALE: DH11-AE, DH11-AD, DEC
Maintained. Terms: List less 5%. FOB

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 $140 down,
cash check or m/o. Bob Meyer, 9 Lockwood Ave., Fieldsboro, NJ 08050.

SOFTWARE & HARDWARE DESIGN
Mini — Micro
RSTS, A Specialty

Programmer wants job BASIC-PLUS, Boston, 20K, 6 MO/YR, Tom Pears (167) 698-7800.

DELWARE VALLEY

TIMESHARING

Eastern Pennsylvania
Philadelphia South Jersey

DEC 11/70 RSTS/E

24 hours - 7 days
Most commercial packages
Custom work, Raw time

NATIONWIDE DATA DIALOG

70 James Way
Southampton, PA 18966
215-364-2800

The producers of the RSTS Professional are now using "telecommunicated" - telephone
communicated phototypesetting.
This method is the latest in typesetting technology. We are now able to receive data
directly from your computer or word processor to our phototypeset. Once these keystrokes
have been "captured", we will mold your copy to your specifications, or to our design if you
wish, and send to you camera ready mechanica l
or final printed material.
The QWORD software is 'menu driven' and therefore very versatile. Users can add, delete, transpose or duplicate text. Words, sentences, paragraphs, pages or even entire documents can be manipulated. All color-coded keys exist to facilitate such tasks as setting margins and tabs, centering, and deleting or transposing parts of documents. QWORD is functionally similar to WPS/8 word processing system for the PDP-8.

In addition to color-coded keys, the system prompts the operator to provide next-step choices as well as error messages. The system can also store commonly used abbreviations and paragraphs. Stored text saves time and reduces error by allowing insertion of commonly used phrases, company or product names, paragraphs or other information by simply using a few key strokes.

Instant Reports

Reporting is a strong suit. Two variations of the report writer exist. One is called PQO, which permits non-technical staff to generate instant reports under computer guidance. A more sophisticated variation of the report writer can handle virtually any format desired. Control breaks, extensive calculations, subtotals and many other features are standard. Sort and selection options are available. More than a million records may be sorted, and over 200 levels of selection criteria exist.

The PDP-11 is the world's most popular microcomputer family. RSTS/E is recognized as the easiest to use and most popular interactive timesharing system available on a mini-computer. It is versatile, sophisticated, and proven operating software, boasting some 20,000 installations. RSTS/E supports concurrent, interactive and batch processing. BASIC-plus, BASIC-plus II, COBOL, FORTRAN, PASCAL, APL, RPGII, PL/I, assembler and more programming languages. This same RSTS/E can run on 11/44, 11/60 and 11/70 computers with as many as 63 independent user jobs and connections to 127 terminals. QUODATA, not DEC, supports RSTS/E on the 11/23.

At the low end, word processing software can start with a system which costs as little as $39,900. The QW620 consists of DEC's new, popular PDP-11/23 operating system available on low cost computers. RSTS/E. Then we add the QWORD word processing software to the already proven QDMS data management system.
LETTERS TO THE RSTS PRO...

LETTERS TO THE RSTS PRO...
continued from page 6

Here's what our new photo contests inspired. (RSTS Professional, Vol. 2, #4, p. 83, Dave Mallery, "Best Caption": p. 91, William Shakespeare at Stratford upon Avon. The question concerning Stonehenge, (Salisbury Plain, Wiltshire, England), p. 19, was not intended as a contest, but simply a rhetorical statement. However, because we received a few comments we decided to include the photo as part of this month's feature.)

Dear Dave and Carl,
I'd like to enter the following caption in your contest.
"You mean I can't play DUNGEON on a VAX!!!!"

I enjoy your magazine very much and look forward to the next issue.
Sincerely, Ronald Charbonneau
Pittsfield School District
Pittsfield, MA

Dear RSTS Professional:
It's anyone's guess as to what Dave Mallery was thinking or about to say in the picture, but he certainly looks surprised to me.

After San Diego, I would offer a caption of either:
"You're running 63 jobs with 96 ports on an 11.70 under Version 7.0 and NO SMALL BUFFER problems!! Come on!!!"
OR
"Did you just say to me how to solve the SMALL BUFFER problem once and for all??!

I really enjoy reading your magazine and anxiously await its arrival. I think that many of the articles contain information that is useful to many people at all levels. I wish that we could see the RSTS Professional distributed more often.

Very truly yours,
Arnold V. Fish, President
Integrated Computer Systems, Inc.

"Oh-h-h-h No-o-o-o! I don't speak his language."

— Ed Hamaker
Amon Press Inc., Sidney, OH

Carl & Dave

Concerning your photo caption contest, how about—
"You mean someone has solved the small buffer problem!!!!!!"

I'm still working on the last issue's TECO problem. A few months ago I was driving East on I-70 just out of Columbus with my wife and saw a similar truck. I said "Hey, TECO!" and my wife just stared at me and said "Huh?". Some people just don't understand.

I really enjoyed your show at San Diego, but I thought you had said that your long awaited "How to build a well structured disk" article was to appear in the December issue. I guess things get a bit blurry after midnight.

Bob Ainsley
National Revenue Corp.
Columbus, OH

Buried and Buried, Bob. See "Disk Structure Notes", this issue.

Dear RSTS Professional:
Below is a caption I feel suitable:
"Small Buffers?? — Oh, I've got 2 of those."

Sincerely, Kim Branch
Daniel International Corp.

Gentlemen:
Your current photo (p. 19) is of Stonehenge.
Many believe that this is indeed a computer, used to calculate astronomical data such as the date of the equinox which was of great importance to prehistoric farmers and hunters. I expect it was reliable, or at least minimal downtime!

Everyone involved with timesharing at our installation really enjoys your magazine; keep up the good work.

Donald E. East
First International Services Corp.

Dear RSTS Pro:
In reference to the photo contest (p. 19), some people think that it was a start. Actually the object in the photo is called a dolmen. A group of these places in a circle is called a cromlech. The most famous cromlech is Stonehenge. It is believed to have been built around 2000 B.C. and seems to have been a kind of astronomical observatory.

How's that?!?!??

Sincerely, Joseph M. Nyikos
Union-Tribune Publishing Co.

THAT was just great!!!

Here's someone who has included a little something for everyone.

Gentlemen

Page 19 — Yes, Stonehenge.
Page 83 — Hey you guys... give me a break will you.

Page 91 — Bill Shakespeare at Stratford upon Avon.

Please remind Users that during a single user Systex one should allocate all except 2 job slots to XBUF. (Free available memory that was.)

How does one reset the statistics without the switch register - i.e., after RDC has nickel'd it?

Bye Y for now,
Peter Dick
Silver Programs, London

OK, Peter, you get a T-shirt for at least one of the above.

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

LOOK

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

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

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

RSTS Professional Rate Increase
Effective May 1, 1981

U.S. (3rd Class) .......... $25
Canada & U.S. (1st Class) . $35
Foreign (Air Mail) ........ $40
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         | DRP06 200MB Slave         |
| DRM02 80MB + Controller   | DRWP06 200MB + Controller |
| DRWM03 80MB + Controller  | DTU45                    |
| DRM04-5 300MB Slave       | DTU77                    |
| DRM04 300MB + Controller  | DTU125                   |
| DRWM05 300MB + Controller | Line Printers             |
| RJM07 600MB + Controller  | Memories                 |
| RW/WM07 600MB + Controller| 11/34 → 11/70, VAX        |
| RM07 600MB Slave          | Systems                  |

DATA SYSTEMS SERVICES

(714) 770-8024
23901 Remme Ridge
El Toro, CA 92630
New, from the makers of DMAX/16™

Super Max

The single-board, 16-line ABLE DH/DM™ that enhances any UNIBUS system with DH-performance at DZ-prices.

Two years ago, we broke new ground with our DMAX/16™, the original alternative to the DEC DH11. DMAX cut the space requirements from nine slots to three and became an immediate worldwide success. Now we’ve come up with something even better. This time it’s ABLE DH/DM™, an alternative that achieves the optimum cluster size – 16 lines with modem control on a single board. You can compare price, or you can compare throughput. Either way, ABLE DH/DM™ beats everything in its class. No one else comes close.

The ABLE DH/DM™ is today’s answer to VAX system needs for DMA communications multiplexing and serves all standard UNIBUS systems equally well. Each 16-line ABLE DH/DM™ installs in any standard hex-width slot at only one unit bus load and is DH11 compatible to the diagnostic level. Just plug it in and see it run – up to 19.2K baud using only half the UNIBUS bandwidth of a DEC DH11.

Key ABLE DH/DM™ features include on-board diagnostics with LED display, modem control on all lines, improved on-board silo depth and variable PROM set for proprietary OEM applications.

Keep up with ABLE and optimize your VAX, PDP-11 or System 20. Write or call today for details on our full line of UNIBUS-compatible special-memory, general-purpose and data-communications products.

ABLE the computer experts
ABLE COMPUTER, 1751 Langley Avenue, Irvine, California 92714, (714) 979-7030. TWX 910-595-1729 ACT IRIN.
ABLE COMPUTER-EUROPE, 74/76 Northbrook Street, Newbury, Berkshire, England RG13 1AE, (0635) 32125. TELEX 845907 HJULPHG.
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-II, the only system with all the sophisticated features you could want.


Data Processing Design, Inc.

Overseas Distributors:
Management Information Services Pty. Ltd.
Melbourne, Australia
Jencon Ltd.
Bristol, England
Système Ltd.
Leeds, England
Network Computer Services Pty. Ltd.
Sydney, Australia