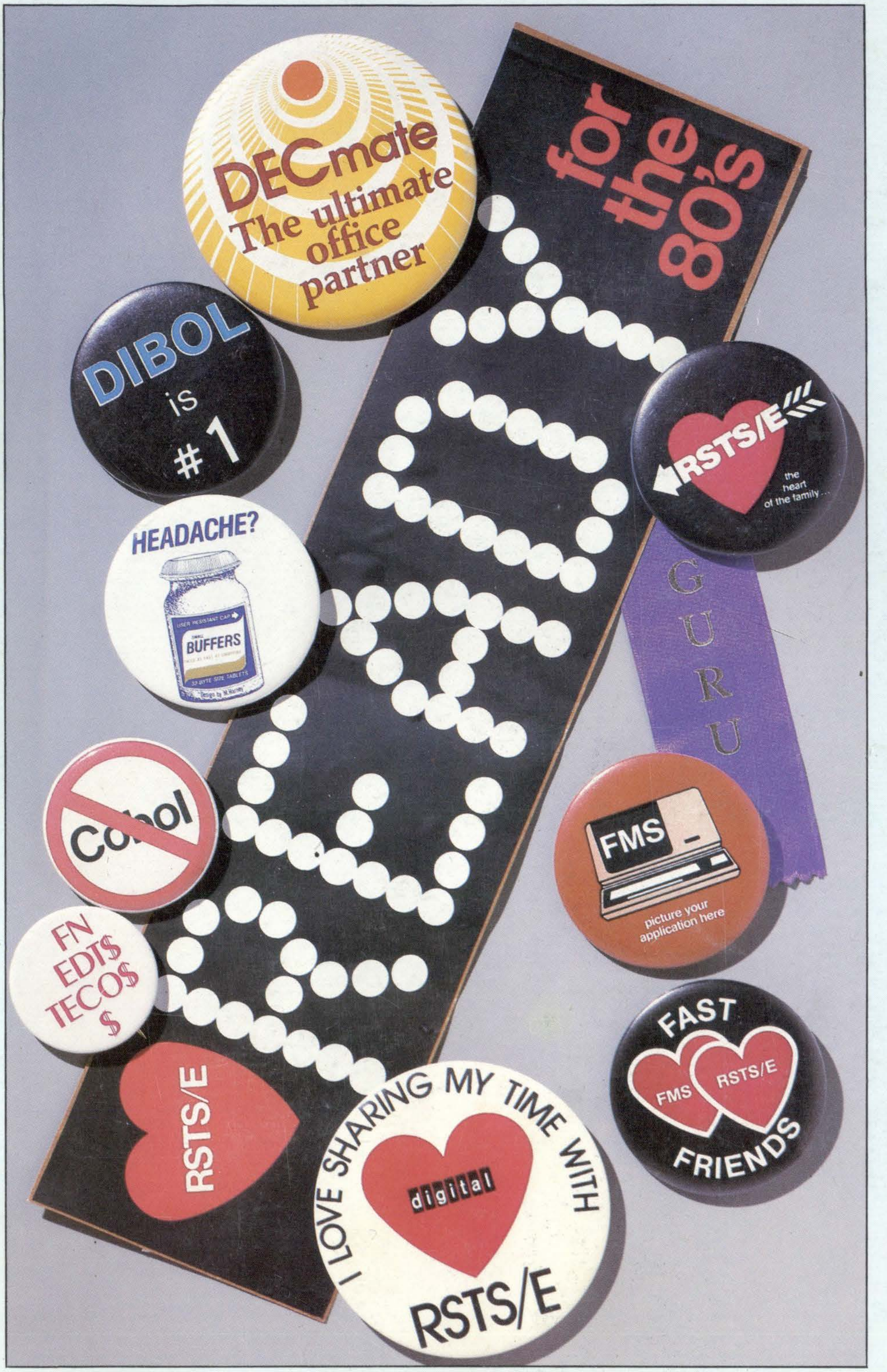
RSTS PROFESSIONAL

Volume 4, Number 1

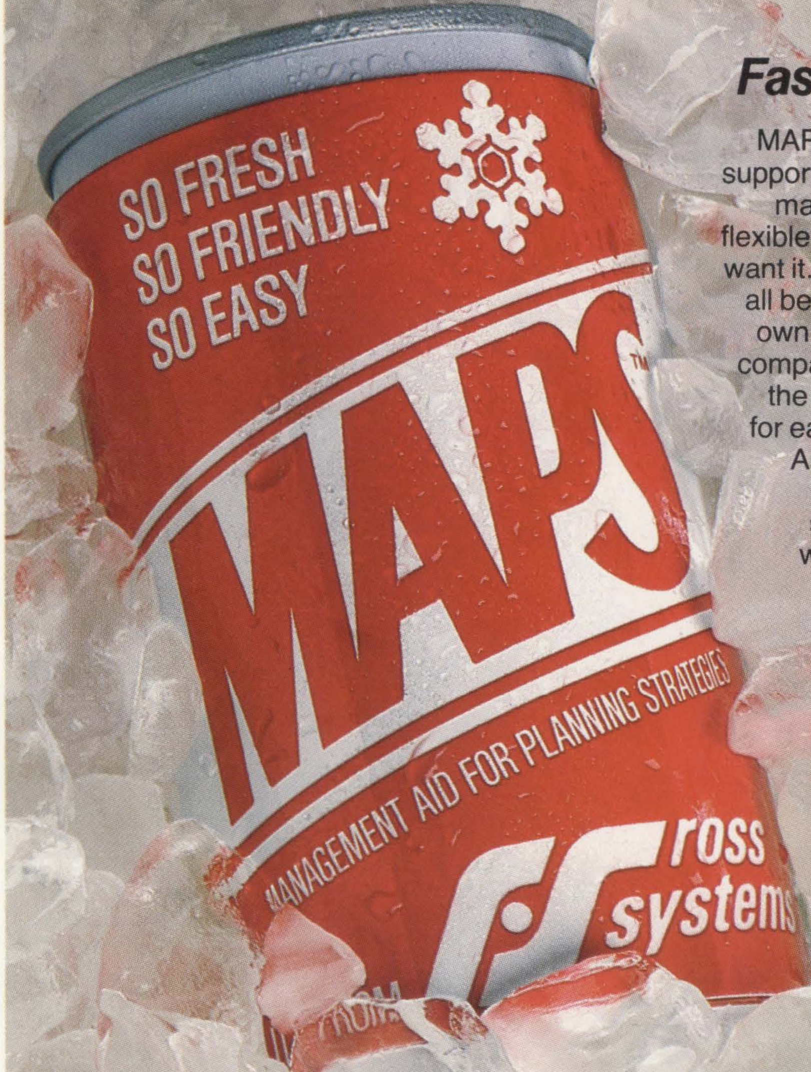
February 1982
10^{00}/issue, 35^{00}/year



INSIDE:

- □ Telephone Costs Rival Data Processing Costs?
- □ Disk I/O From MACRO
- □ Getting the Most Out of Your DEC Field Service
- □ FILMAP.BAS
- □ QUERY.TEC
- □ LINK-TIME Initialization of BP2 MAPS
- □ RSTS/E's Small Buffers
- □ The Case for NFF
- □ Tips & Techniques
- □ How Do You Read A RSTS/E Disk Structure?
- □ The VAX-SCENE
  Writing Structured Program in VAX-11 BASIC
- □ BIO.BAS
- □ The Low Speed Spooling Package
- □ Proposed Standard EDT 2.0 Initializer
- □ SHOW
- □ "Input Loop" Programming Technique
- □ Program Notes
- □ More . . .

Financial Decision Support Canned For Easy Use!



Fast Flexible Information

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

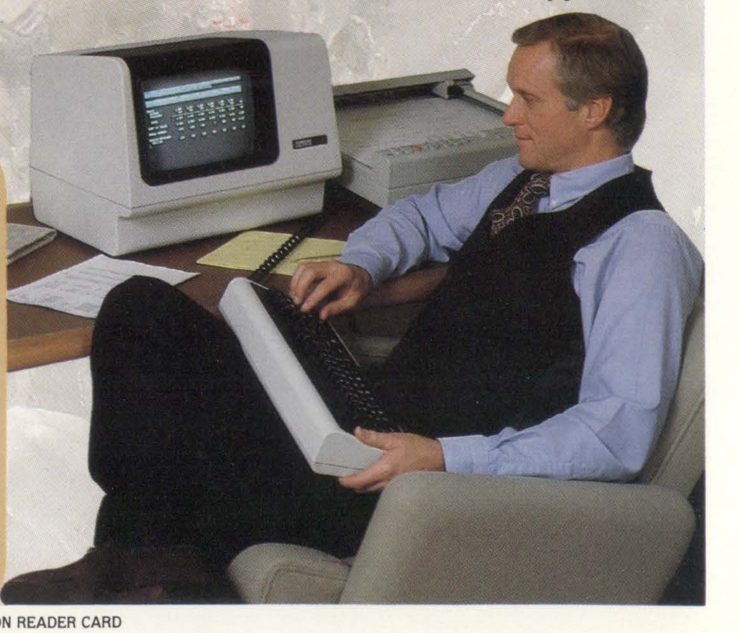
*DEC is a trademark of Digital Equipment Corporation.

ross systems

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

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

"Now that's fresh decision support."



MAPS meets your daily requirements:

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

CIRCLE 1 ON READER CARD

SSi

SOFTWARE

SUBJECT: CALOUT

NEED TO GET FILES ONTO SOME OTHER SYSTEM?

OR BACK ONTO YOURS?

BY TELEPHONE?

FAST?

SOME THINGS TO CONSIDER. Of the few utility packages available for this requirement there are some important differences.

CALL ANY RSTS SYSTEM YOU LIKE. Some designs require that a copy of the communication package reside on each correspondent system. This means you pay the license cost again and again for each system in a defined network. Many users wish to link up with any system. The CALOUT package is designed to support this requirement. Only your system requires the license.

MOVE ANY KIND OF FILE AND THE ATTRIBUTES. RSTS files are always associated with special file attributes that are not part of the data itself. This information is often essential to the use of the file. Some communication packages do not deal with it all. The CALOUT package automatically sets up the correct file attributes on the correspondent RSTS system. Virtually any file type can be transferred.

ABOUT ERROR CHECKING AND CORRECTION. Standard telephone lines can insert bad data into a link. Some communication packages have no way of overcoming this problem. The CALOUT package insures that each file is transferred correctly. This is done by automatic error detection and correction.

CONCERNING FILE TRANSFER SPEED. The speed for moving files is determined by three primary factors. Dominant is the baud rate of the communication link. Normally, a communication package will transfer files between systems as fast as the hardware link will permit. Synchronous or asynchronous links may be used with the CALOUT package. A wide range of standard communication hardware from a variety of vendors is found to be suitable. Transfer speed will be lost if the link is bad and a lot of error correction is required. Also, speed is lost if either system is heavily loaded and responding poorly. The CALOUT package is well optimized and uses very little system resource.

ON BEING EASY AND CONVENIENT TO USE. The user interface can be quite awkward and clumsy to work through. This is true to varying degrees among the packages offered. The CALOUT package is designed to support even the casual user without reference to the user manual. This of course is when everything is going well. When problems occur with the hardware link, CALOUT provides extensive diagnostic information to guide hardware maintenance activities. Protocals for logging and using a correspondent system can be invoked automatically from a table you define.

TRANSFERRING SETS OF FILES. The CALOUT package provides a wild card transfer request. With this feature many files may be automatically moved with a single request. It works much like PIP.

ALSO, ON BEING A TIME SHARED USER. You may often wish to run a few tasks on the correspondent system. The CALOUT package will let you run tasks or move files as you wish. You may pop back and forth between several powerful modes with simple control commands.

ON NEEDING SOMEONE AT THE OTHER END. Watch out for this one. It can be quite inconvenient if you find that you must have someone at the correspondent system set your link up before you can transfer files. The CALOUT package requires no attention whatever at the other system site. Of course, they must have some standard dialup capability at their end.

REGARDING LOCAL SYSTEMS. For systems that are physically near each other, the CALOUT package can be implemented without a telephone line or any special hardware beyond a standard cable linking the two or more systems.

CALLING A NON-RSTS SYSTEM. The CALOUT package will fully support links with RT11 and VAX systems. Other non-RSTS and non-DEC systems are supported for text file transfers.

ABOUT THE TWX LINKS. The CALOUT package supports alternate dialup and direct TWX connection for fully automated use of the Western Union network.

DIGITAL SYSTEMS, INC. Clyde

HOW TO GET MORE INFORMATION. Call Janet at (617) 275-6642, or write: **Clyde Digital Systems, Inc., P.O. Box 348, Bedford, MA 01730.**

Contents

Coming . . .

- ■ VAX MACRO
- ■ Relational Data Bases
- ■ A Free Advanced Video Board
- ■ Push/Pull MACRO's
- ■ More EDT Hints & Kinks
- ■ Program Notes
- ■ Login without LOGIN
- ■ MACRO MAN
- ■ VAX-Scene
- ■ V7.1 Report
- ■ Word Processing
- ■ A PDP-11/115?
- ■ How to use BUILD
- ■ More . . .

The RSTS Professional Magazine, February 1, 1982, Vol. 4, No. 1. Published bi-monthly. Single copy price $10.00, $35.00 per year. Published by M Systems, Inc., 753 Johns Lane, Ambler, Pa. 19002-0245, telephone (215) 542-7008. Send all correspondence and change of address to: Box 361, Ft. Washington, Pa. 19034-0361.

From the editors...

NO!

Carl Marbach

NO! Only two letters long, but it sure packs a lot of negativity into a small space. The Word at the December Los Angeles DECUS meetings was: NO! DECUS is becoming one big NO and its catching on. Clerks and secretaries won't say no if they think the boss will say yes; the NO-ism comes from the top.

What NO's? Some examples: Joyce Hayes, Steve Stepanik and Boyce Cargill gave a three part TECO tutorial (attended by over 125 people each session) that is the best, most professional tutorial I have seen presented at a DECUS meeting. NO handouts. When I called to plead the case some time before the December meeting I was told that since the session was not new (it had been presented before, and was back by popular demand) it did not meet the "new" requirements for having DECUS produce handouts, besides it would be in the proceedings. More: Pretty stiff airfares to attend DECUS for us Easterners, how about the possibility of charter flights to Atlanta in the spring from the West and to L.A. next fall from New York or Boston? NO. Might be some liability for DECUS. I belong to smaller organizations that offer charter flights for their members, but not DECUS. NO. Dan Esbensen from North County Computer Services has given sessions at several DECUS meetings and this was no exception with a presentation on optimizing strings in Basic. I happened to be in the DECUS office when Dan came in at 8:45 one morning looking for a marking pen to use on the overhead projector during his presentation scheduled for 9:00 A.M. "Can't give you one (NO!) ", said one of the DECUS workers, "I give them out and I don't get them back". "I'll leave my briefcase", replied Dan, trying to get the needed marker. "NO". Lucky for us Dan persisted and finally got someone to give him the 98 cent marker. At least one request for a "birds-of-a-feather" session wasn't even given the courtesy of a reply; A silent (pocket-veto) NO. You might also have noticed that there were NO RSTS handouts, There is NO new TECO card, and the message board was cleverly stashed inside the exhibit hall where it was accessible only from 9-5 (NO, we can't move it out).

I am a strong supporter of DECUS; Even now I am the TECO sub-SIG chairman and will give no less than three sessions at the Spring meetings (New user, TECO Macros, and How to get the most out of Version 7.1 or The Carl and Dave show revisited). The leadership of DECUS has got to change its perspective to a more positive one. The members of DECUS are important, the most important thing DECUS has. This is a USER organization, decUs, to serve the USERS not to support itself.

I would like to hear more YES's from DECUS. New members would feel more at home and old members might give a little more if they felt some positive feedback from the DECUS leadership. Why don't we find more ways to say "YES". Stop running scared with all the NO's. Grace Hopper once told a DECUS audience that she would come back to haunt anyone who said, "but, we have always done it that way". I think Ms. Hopper would visit some of you who only know how to say NO, and can't figure a way to say "YES".

DECUS might learn something from a story I like to tell of a time I was traveling in England after a United Kingdom DECUS Seminar. We spent the day driving and sightseeing and finally arrived at the Inn we were to spend the night at 5:30 P.M., past the usual hour for Tea. We were tired and cold and hopefully asked, "are we too late for Tea", "you are never too late for Tea", they replied. "Where can we have Tea", we asked. "Anywhere you like", was the answer. I told my wife, "we're going to like it here". I was right. And we're going back. ♥

Dave Mallery will return in the next issue.

We dedicate this issue to our friend, SIMON SZETO.

Thanks for listening to us— over and over again.

Good luck, Simon, whatever you are working on . . .

Carl & Dave

RSTS PROFESSIONAL

Editors
R.D. Mallery
Carl B. Marbach

Assistant Editor/Advertising
Helen B. Marbach

Controller
Peg Leiby

Administrative Assistant
Hope Makransky

Subscription Fulfillment
Kathi B. Campione

United Kingdom Representative
Pauline Noakes
RTZ Computer Services Ltd.
P.O. Box 19, 1 Redcliff Street
Bristol, BS99-7JS
Phone: Bristol 24181

Contributors
Philip G. Anthony
Al Cini
Mark H. Deibert
Mark J. Diaz
Steven L. Edwards
Peter Ehrenstrom
Lawrence Fisher
Rob Frazer
Timothy P. Hart
Michael H. Koplitz
Bob Meyer
James F. Shaughnessy, Jr.
David Spencer
Jim Swanson
Traudi Tissler

Cartoons
Douglas Benoit

Photographic Consultant
Bill Marbach

Design & Production
Grossman Graphics

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

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

LOW COST RSTS/E NETWORKING.

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

REDUCES COSTS

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

EASY TO USE

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

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

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

DMG/NET

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

Digital Management Group Ltd.
4800 Yonge Street, Willowdale, Ontario, Canada M2N 6G5
Telephone: (416) 225-7788

CIRCLE 88 ON READER CARD

LETTERS to the RSTS Pro...

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

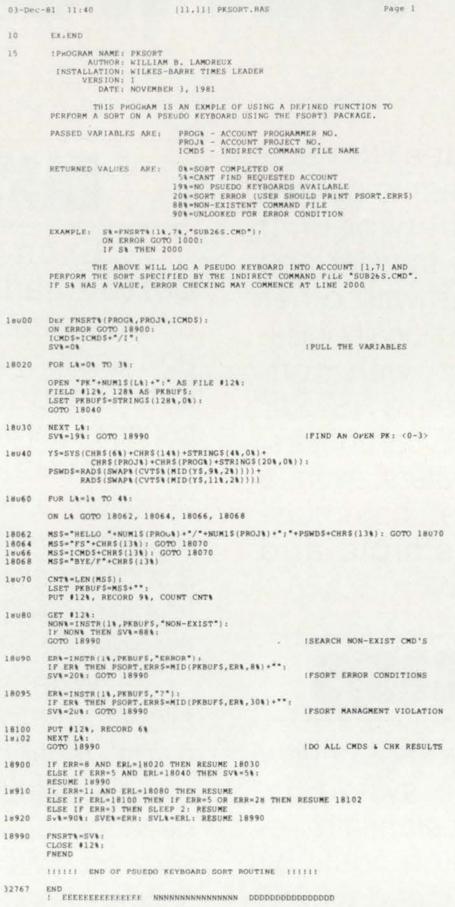
We have just installed the enclosed function [*see below*] on one of our 11/34's. I do not recall seeing anything which performed a similar operation and I thought your readers might be interested. The function performs a sort on a file from within a running program by making use of a pseudo keyboard.

We saw a need for this function because we try to let the various departments within the company do as much of their own processing as possible. So, not only do the departments perform their own data entry, they sort, post and pull reports as well. This lightens the load considerably for the D.P. department and it keeps any questions concerning their data where it belongs, in their own departments. The sort function is incorporated easily into any program and it allows the user to select the kind of order he wants simply by answering a question within a program. Chaining is then possible between multiple programs and there is no user intervention required. We considered using batch jobs but the batch programs would be sitting on the system, taking up job slots and getting very little usage. In addition, any instructional messages or error messages from the running programs will be immediately apparent so that the appropriate action can be taken quickly.

We enjoy the *RSTS Pro* magazine and we have learned a great deal from it. But, one subject that we have never seen addressed is how to gear programs for the non-D.P. user. This is not quite as simple as it sounds. As the System Manager, I have to be constantly on the lookout to be sure that my programmers tailor their work for the user and not for themselves. Programmers seem to enjoy writing programs that can do really 'amazing' things, but if the user doesn't need those things or can't use certain features then the programmer should not be incorporating them into a user oriented program. Experimentation is fine and should always be going on, but not at the expense of the user. The programs should be geared so that the user doesn't have to answer too many questions or have to follow too many procedures but is still able to get everything he is looking for by himself. This is a very fine line, the programmer must understand that he has to tread carefully or he will fall into the trap of giving the user either too much or not enough. One thing that is very important is that the programmer learn to listen to exactly what the user/analyst is asking for. Then be able to extrapolate what they have not asked for and give him that too, but without going so far as giving the user more than he knows what to do with. Many off the shelf packages offer much more than a typical user really needs but those packages have to be geared for any type of user. When a programmer is working on an in-house program and is familiar with the people who will be using that program, he should gear the program towards those people. The program should be as intelligent as possible and be able to make it's own decisions without the user being aware of what is taking place. The user should be presented with concise, understandable questions and not have to refer to either an operations manual or to the Computer Room staff for help. Conversly, the user should not be presented with myriads of information about the program and how to answer the questions because the user doesn't need all that information. This is our basic philosophy on tailoring programs for the user. I could go on, but I believe you see what I am trying to get at by now.

Thank you for letting me vent my feelings and I hope you will address this topic in more detail in future issues as more and more D.P. shops are beginning to take our approach of letting the end user do their own data processing.

Kenneth Clark, D.P. Manager
The Wilkes-Barre Times Leader, Wilkes-Barre, Pa.



```
03-Dec-81  11:40                [11,11] PKSORT.BAS                    Page 1

10      EX:END

15      !PROGRAM NAME: PKSORT
              AUTHOR: WILLIAM B. LAMOREUX
         INSTALLATION: WILKES-BARRE TIMES LEADER
             VERSION: I
                DATE: NOVEMBER 3, 1981

              THIS PROGRAM IS AN EXMPLE OF USING A DEFINED FUNCTION TO
        PERFORM A SORT ON A PSEUDO KEYBOARD USING THE FSORT3 PACKAGE.

        PASSED VARIABLES ARE:   PROG% - ACCOUNT PROGRAMMER NO.
                                PROJ% - ACCOUNT PROJECT NO.
                                ICMD$ - INDIRECT COMMAND FILE NAME

        RETURNED VALUES  ARE:    0%=SORT COMPLETED OK
                                 5%=CANT FIND REQUESTED ACCOUNT
                                19%=NO PSUEDO KEYBOARDS AVAILABLE
                                20%=SORT ERROR (USER SHOULD PRINT PSORT.ERR$)
                                88%=NON-EXISTENT COMMAND FILE
                                90%=UNLOOKED FOR ERROR CONDITION

        EXAMPLE:  S%=FNSRT%(1%,7%,"SUB26S.CMD"):
                  ON ERROR GOTO 1000:
                  IF S% THEN 2000

              THE ABOVE WILL LOG A PSEUDO KEYBOARD INTO ACCOUNT [1,7] AND
        PERFORM THE SORT SPECIFIED BY THE INDIRECT COMMAND FILE "SUB26S.CMD".
        IF S% HAS A VALUE, ERROR CHECKING MAY COMMENCE AT LINE 2000


18000   DEF FNSRT%(PROG%,PROJ%,ICMD$):
        ON ERROR GOTO 18900:
        ICMD$=ICMD$+"/I":
        SV%=0%                                           !PULL THE VARIABLES

18020   FOR L%=0% TO 3%:

        OPEN "PK"+NUM1$(L%)+":" AS FILE #12%:
        FIELD #12%, 128% AS PKBUF$:
        LSET PKBUF$=STRING$(128%,0%):
        GOTO 18040

18030   NEXT L%:
        SV%=19%: GOTO 18990                              !FIND AN OPEN PK: <0-3>

18040   Y$=SYS(CHR$(6%)+CHR$(14%)+STRING$(4%,0%)+
             CHR$(PROJ%)+CHR$(PROG%)+STRING$(20%,0%)):
        PSWD$=RAD$(SWAP%(CVT$%(MID(Y$,9%,2%))))+
              RAD$(SWAP%(CVT$%(MID(Y$,11%,2%))))

18060   FOR L%=1% TO 4%:

        ON L% GOTO 18062, 18064, 18066, 18068

18062   MS$="HELLO "+NUM1$(PROG%)+"/"+NUM1$(PROJ%)+";"+PSWD$+CHR$(13%): GOTO 18070
18064   MS$="FS"+CHR$(13%): GOTO 18070
18066   MS$=ICMD$+CHR$(13%): GOTO 18070
18068   MS$="BYE/F"+CHR$(13%)

18070   CNT%=LEN(MS$):
        LSET PKBUF$=MS$+"":
        PUT #12%, RECORD 9%, COUNT CNT%

18080   GET #12%:
        NON%=INSTR(1%,PKBUF$,"NON-EXIST"):
        IF NON% THEN SV%=88%:
        GOTO 18990                              .        !SEARCH NON-EXIST CMD'S

18090   ER%=INSTR(1%,PKBUF$,"ERROR"):
        IF ER% THEN PSORT.ERR$=MID(PKBUF$,ER%,8%)+"":
        SV%=20%: GOTO 18990                              !FSORT ERROR CONDITIONS

18095   ER%=INSTR(1%,PKBUF$,"?"):
        IF ER% THEN PSORT.ERR$=MID(PKBUF$,ER%,30%)+"":
        SV%=20%: GOTO 18990                              !FSORT MANAGMENT VIOLATION

18100   PUT #12%, RECORD 6%
18102   NEXT L%:
        GOTO 18990                                       !DO ALL CMDS & CHK RESULTS

18900   IF ERR=8 AND ERL=18020 THEN RESUME 18030
        ELSE IF ERR=5 AND ERL=18040 THEN SV%=5%:
        RESUME 18990
18910   IF ERR=11 AND ERL=18080 THEN RESUME
        ELSE IF ERL=18100 THEN IF ERR=5 OR ERR=28 THEN RESUME 18102
        ELSE IF ERR=3 THEN SLEEP 2: RESUME
18920   SV%=90%: SVE%=ERR: SVL%=ERL: RESUME 18990

18990   FNSRT%=SV%:
        CLOSE #12%:
        FNEND

        !!!!!!  END OF PSUEDO KEYBOARD SORT ROUTINE  !!!!!!

32767   END
        !  EEEEEEEEEEEEEEE  NNNNNNNNNNNNNNN  DDDDDDDDDDDDDDD
```

Thanks, Kenneth. We agree with your philosophy and will try to get an article on the subject.

—

I am writing to alert RM02/RM03 sites to a possible problem. It is called non-computer personnel knocking the drive(s) offline.

I first encountered this problem when our new line printer was installed next to the RM03s. Users would pick up their output and proceed to use the drives as TABLES to sort through the listings. Now, we RM03 users know it doesn't take much pressure to push the START button off and I learned in short order this was a problem. It happened 5 times in one week, last time by my own hand(?).

I, however, developed a solution. It is a piece of fiberglass, 5″ by 3″, held over the opening by three strips of strong scotch tape. This can be flipped up on top of the drive when powering up or down and when changing disks.

Granted, this may not be an ideal solution, but it allows the users to get their listings and permits MA bell to service her PBX system, housed in the same room, without my hair falling out from worry.

I hope someone benefits from this. Keep up the good work. This magazine is the greatest thing since RSTS.

Mark Ruggiero, Arlington, MA

P.S. Is it possible to buy a lifetime subscription?

Thanks for the tip, Mark. As far as the lifetime subscription goes — thanks for the compliment and we're enthused enough to think about it as a possibility!

—

I would like to express my appreciation for introduction of a series on 'Have You Stolen Any Good Software Lately?' in *RSTS Professional.*

I would like to read about the software techniques that could be incorporated in the software products to effectively counter this menace. It will be interesting to know about both simple and less costly as well as complex, sophisticated and more costly techniques for appropriate use with low/high cost software products.

May I make a personal request to you? We will like you to mail to us some references of books/articles that may have been published in connection with this topic.

The thought of receiving *RSTS Professional* every alternate month is really pleasing. How thoughtful of you to have given the New Year gift for 1982 in advance!

Thanks for a great magazine.

Girish Shah, Marketing Manager,
Hinditron Computer Systems &
Consultants Pvt. Ltd., Eros Bldg.,
42, Maharshi Karve Road, Churchgate,
Bombay 400 020

Thank you, Girish, for your interest. We are very excited about offering the "RSTS PRO." bimonthly. We'll try to help you with the information you requested. Readers, how about it??

—

DR. JOEL SCHWARTZ

To the Doc: I read your short article in a back issue [*RSTS Professional, Vol. 3, No. 2, p. 76*], so this reply may be a little late. You asked for help with DUNGEON; however, the game you seemed to be playing in the article was ADVENTure. A different animal altogether. After three years of on and off work, I have succeeded in playing perfect games of both. (I am proud to say that I have even conquered the infamous endgame of DUNGEON). I have maps of both games including the various "unmappable" regions like the forests and mazes, so I might be able to help. (I also have hard copies of my "perfect" sessions if you need precise details.) Tell me which one you are exploring and I'll be glad to lend a hand.

In return, I ask one small favour: where or who has an up-to-date version of both of these games? The system I manage has neither and I would like to get them if possible. Who should I contact?

One last point. A friend of mine tells me that you are going to print a map of DUNGEON or ADVENT in the next issue [*Vol. 3, No. 4*]. I strongly disapprove of that. I know this sounds very arrogant coming from one who has finished them, but while I was working at them I desperately wanted a map and in retrospect I know that if I had obtained one it would have ruined the game. Not half but ALL the fun in that game is figuring out—even if it means eight-hour marathon sessions with dozens of restored games after weeks of sleepless nights. It's worth it. Honest. I realize this admonition will likely go unheeded but I had to say it.

Good luck with whichever one you are playing.

John Partridge, Princeton, NJ

Well John, we haven't reached the Doc for comment yet, but we do feel that he would make out better if he knew which game he was working on. Now then, concerning the map, as you probably know by now you are too late to stop the presses for DUNGEON, however, we may consider taking a poll before we print future "solutions". Thanks for the suggestion.

... continued on page 78

Telephone Costs Rival Data Processing Costs?

Software Available to Manage Telephone Costs

By Traudi Tissler, Communications Analysis Corp.

To many of you this may not be a surprise. But if it is, maybe you should consider this. Especially if you belong to the group of managers who have found it necessary to give this issue of telecommunications costs a low priority in the past. Take for example the case of a local Boston-based University. It meant $67,000 in annual savings to the University. The telecommunications manager was in the process of looking at new telephone switches to replace an older piece of equipment. Rather than sort through mounds of telephone bills close to the tune of 11,000 call records monthly, he sought a teleprocessing service bureau to do the job. Within a week, and for less than $1,000, he was presented with a reconfiguration of his voice network which would conservatively save $67,000. Conservatively, since the study was based on an off-peak summer month.

Your telephone system may serve more employees than the 1000 lines at this University in Boston, but your savings may be as astounding even if your company has a smaller telephone system, in the range of 50 to 75 lines.

IN-HOUSE PROCESSING

Up until recently, the customary route to accomplish CDR/SMDR analysis has been to send the call-data to a teleprocessing bureau. Yet, today many of the large users are turning to in-house processing. Complete software packages are available in the neighborhood of $18,000 and up. Although a company may wish to process in-house, the packages are not generally developed in-house. The more reliable teleprocessing companies have the experience and expertise and will custom-tailor a package for you.

WHAT IS TELEPROCESSING?

First, one may ask "What exactly is a teleprocessing service bureau?" Teleprocessing service bureaus came into existence about 10 years ago; about the same time so many other industries were putting data processing power to use. These entrepeneurs recognized a void. There existed a need for a reporting system to control, manage, and analyze telephone use and costs. Until the early 1970's, the local Bell System operating company was the most logical, if not only, place to turn to for an answer. Yet, the Bell System as an answer poses several limitations and drawbacks:

1. Bell System limited the number of studies it would do, therefore you could not monitor PBX traffic activity on a periodic monthly basis.
2. You could not pay for a study or additional studies even if you wanted to make it attractive for Bell to respond to your request.
3. You might have to wait a couple of months, if not longer, for the information.
4. The studies were not a routine matter, so snags could easily arise.
5. Some information that would be helpful was not available.
6. Bell offered no equipment which would allow you to do this job yourself.
7. Bell would not recommend the new special common carriers who offer reduced costs for long distance calls.

Now, back to the entrepeneurs. Before the telephone call records could be processed, this data had to be collected from the telephone switch or PBX (Private Branch Exchange). Basically, a PBX is a switching unit located in the office building allowing calls within the building to be switched or transferred through this unit, rather than through the Bell System Central offices, as was the case with older switches. In addition, the PBX offers significant features, such as the ability to confer with a third party, answer a ringing phone from any phone across the room, redirect calls to another extension when you are unavailable.

So it was off this PBX that each and every call was to be recorded. A collection device was manufactured and attached to the RS232 port off the PBX, and data collection began. At the end of the month, the mag-tape, floppy disc, cassette, cartridge, etc. was ready to be processed. And this process became known as CDR/SMDR processing to the telephone industry and its customers.

CDR/SMDR PROCESSING

In its early stages, CALL DETAIL RECORDS/STATION MESSAGE DETAIL RECORDS processing was an expensive investment, both expertise and dollar wise for end-user, inhouse processing. So teleprocessing service bureaus grew. On a monthly basis, the service bureau, combining the expertise of data processing with telecommunications, received the data and turned a report back to the client.

THE MANAGEMENT TOOL

The reports fall into two broad categories: telephone usage information and trunk/traffic utilization. And to the manager armed with this information, it means significant savings.

The telephone-usage information is built on a pyramid-like scale. Activity is first summarized on a per-user basis (see Diagram A). This information will show exactly who called, where, when and for how long. Also, these reports show how the call was placed, how often.

The more sophisticated programs compare the actual route chosen to complete the call and compare it to the most cost-effective route, highlighting in dollar figures what re-

Word Processing for DEC Systems

HAMILTON

- Spanning the DEC Product Family with a wide variety of system configurations, ranging from single user to large multi-workstation environments.

- Software designed to run under VMS, RSX/11M, RSTS/E, or its own operating system.

- Providing full word processing capability with the emphasis on ease of operation and flexibility.

- Available as layered product or bundled with system hardware for a dedicated word processing system.

- Facilitates integration of Word Processing and Data Processing files.

- **FEATURES INCLUDE:**

Auto Center

Auto Underscore

Global Search/ Replace

Auto Decimal Alignment

User Defined Keys

Auto Page Numbering

List Processing

Paragraph and Abbreviation Library

132 Character Wide Screen

Document Orientation

Dynamic Nature

"Menu" Driven

Operator Prompting

Auto Decimal Alignment

- Excellent Price/Performance ratio for your system dollar.

- Training and Support

WRITE OR CALL FOR INFORMATION AND LITERATURE

SYSTEMS WITH SOLUTIONS

HAMILTON

Hamilton Rentals

Pearl Court,
Allendale, N.J. 07401

TOLL FREE 800-631-0298

IN NEW JERSEY
201-327-1444

LONDON PARIS DÜSSELDORF TORONTO NEW YORK

DEC, VMS, RSX/11M, RSTS/E, MACRO-11, PDP 11 are trademarks of Digital Equipment Corporation

CIRCLE 10 ON READER CARD

routing would mean in savings to you. You will even get an indication of employees productivity since non-business calls, such as "DIAL-A-JOKE," "WEATHER," and other personal calls can be identified. A manager will also have the ability to generate a report for excessively long calls, whereby he sets parameters for length of call or number of calls to be identified.

The hierarchial scheme then summarizes activity within the department, allowing a manager to compare calling patterns and activity within the group. (Diagram B) This is available on additional levels, such as district and division, up to the company or corporate level. (Diagram C)

Before moving on to the topic of trunk/traffic utilization, one should note that the departmental report can be used for cost-allocation purposes. One of the woes of telecommunications managers is that this department operates in the red. Cost-allocation will not only allow them to recover actual costs for phone calls, but recover other overhead expenses, including amoritization of the PBX, cost of telephone lines and set, and salary of PBX operators.

Similarly, certain industries will not have access to information so that clients can be billed for calls made on their behalf. An excellent application is the law market. An account code system is developed so that prior to each call, a lawyer punches a unique I.D. code. When the call data is processed, this information is readily available to be put on the client's next statement. It's as simple as that! (Diagram D)

VOICE NETWORK ANALYSIS

To address the issue of trunk/ traffic utilization, the processed reports will provide information on actual use of private lines, FX (Foreign Exchange), WATS (Wide Area Telephone Service), and local lines. Although it is generally known that long distance calls are the most expensive type of call, some companies do not seem to have the expertise available to them to take full advantage of less-expensive services. The report package available through the teleprocessors provides this information in an easy to read format for the nontechnical managers, as well as enough detail for the manager with traffic engineering expertise.

WHAT IS WATS?

Often when a company installs WATS lines, the correct combination of lines and service areas is not used or the need changes over time. Diagram E shows that 12,400 minutes of WATS calls were placed over a Band 5 line, which serves the entire U.S. calling area. However, this use is not cost-justified. Only 2300 odd minutes were actually placed to cities served in Band 5. The bulk of calls, 26.8%, were placed to areas served by Band 3. This is only one part of the revised configuration, and results in a saving of $5400. (Diagram E)

Another misnomer is that WATS calls are "FREE." It is not even true to say that WATS calls are always less expensive. When an analysis of telephone calls is made, it is not surprising to find calls being made over WATS lines to towns no more than 2 or 3 miles away. Although the figure may vary depending on your location, it is generally less expensive to place a long distance call over local lines to areas as far away as 25 to 40 miles, rather than use your WATS line. This played a significant role in the $15,000 annual savings of a 70 line PBX used by a New England bank. This poses the interesting issue to managers of training employees to use the correct type of line. The easiest solution is to program the PBX with ARS (Automatic Route Selection).

TRAFFIC ANALYSIS CONTINUED

Trunk reports will also show the volume of traffic going out and in some cases, being received by each line. This information can indicate the need for addition or deletion of lines in addition to problems on the line where insufficient traffic is noted. A quick mention of FX potentials should be made here, since this may now be the most cost-effective method to handle traffic where a significant volume is placed to the same locale. A sort is usually included in the report package which shows area code and exchange (i.e. 212-246-XXXX) listings with the number of calls completed and the number of minutes.

Information is available, some in graph form, to identify the peak calling period, so that one can determine if your trunks are engineered for expected grade of service or line availability. Remember, the analysis of only trunk traffic for the Boston-based University resulted in a savings of $67,000 annually. This does not take into account the savings that will be realized when non-business calls are reduced. Just the knowledge that a record of phone calls placed by an individual is available is incentive enough to minimize abuse of this company benefit.

THE "MOSTE" FOR YOUR TELEPHONE SYSTEM

Communications Analysis Corp. has developed a software package, MOSTE, to be used for the control and management of the telephone systems and its costs. MOSTE is proprietary licensed software which will provide comprehensive telecommunications management reports that record and summarize calls made from your telephone switch.

MOSTE (Management of Systems Telephone Expenses) is IBM compatible and another version will be ready to use for DEC equipment. The software package is fully documented and is supported by source code for reports. Performance critical portions are implemented in assembly language, and reports are processed using COBOL. MOSTE provides data security, flexible on-line inquiry, and complete report-program generation. Documentation includes detailed instructions for installation, codebook for text data files, and a comprehensive user's guide. Communications Analysis Corp. will provide on-going maintenance and support of this package to reflect any rate, tariff, and telephone central office changes. The package is ideal for OEM or end-user.

ON-SITE PROCESSING

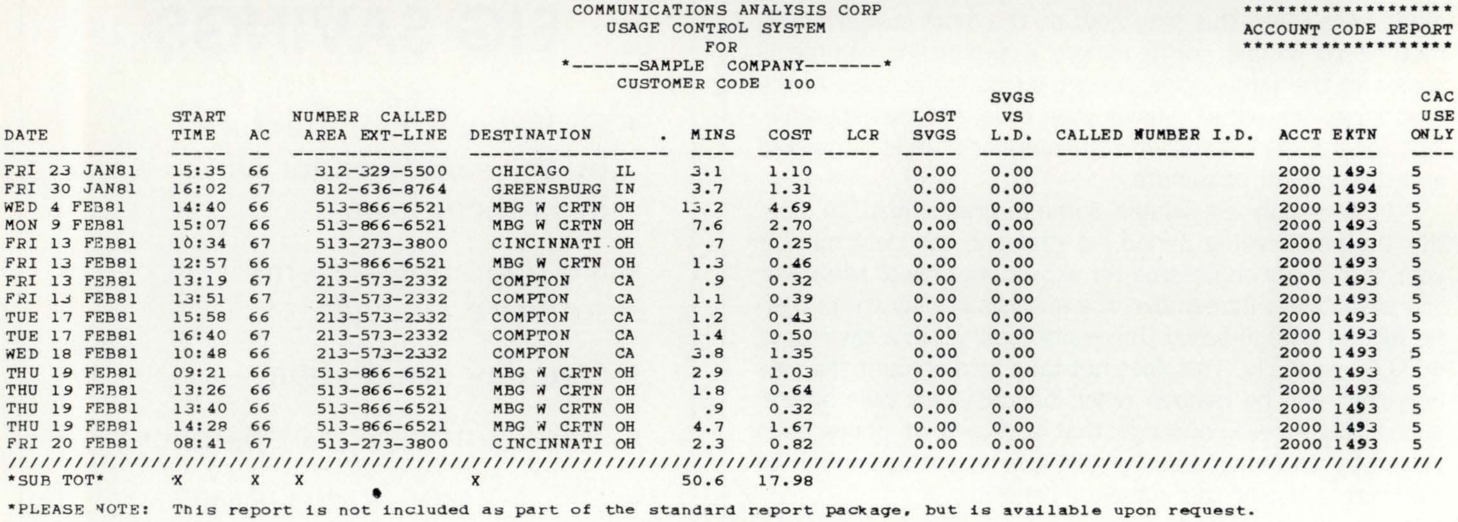
In addition to in-house processing, companies now enhance the PBX with a self-contained SMDR/CDR processing unit. Therefore, a manager could have available to him on an almost immediate basis, a listing of the most recently placed calls. It should be pointed out, however, that these systems generally do not have the capability to store data for long periods, so the in-depth analysis that is available after a monthly run through a teleprocessing in-house program is lacking with the self-contained processor.

A WORD TO THE WISE

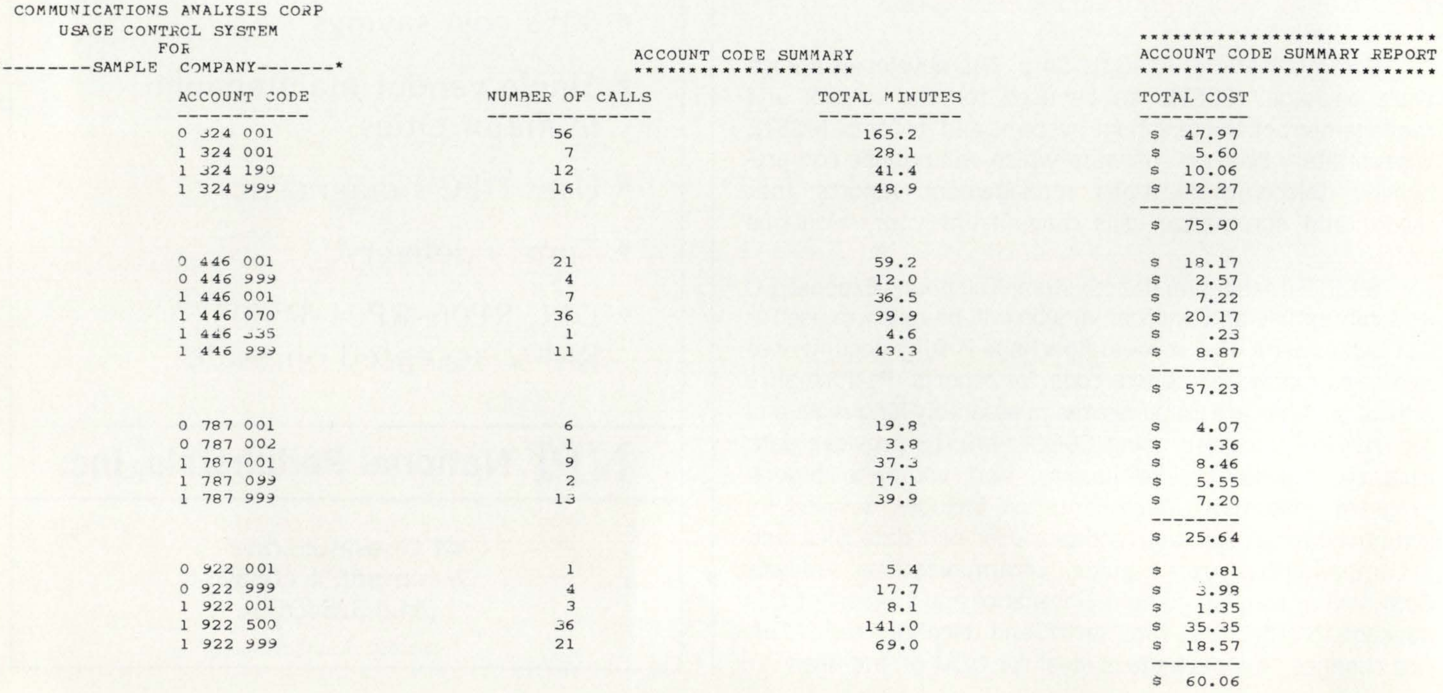
Generally speaking, the call traffic for a company is consistent from month to month. However, certain industries are effected by seasonal variations, such as airlines and universities. So it is always a good idea to look at traffic patterns for a couple of months before actually reconfiguring your trunks. Once you do pursue the area of telephone costs management and control, speak with several companies; this business is a competitive one and some companies offer excellent rates with a good staff of experienced data processing and telecommunications staff employees. Then, keep in close touch with your representative from the teleprocessing company, and ask questions; usually, they will be more than happy to help you interpret your reports.

TELEPHONE COSTS RIVAL DATA PROCESSING COSTS

Back to our original question. This question remains for you to investigate. As with almost any industry, telephone costs are affected by inflation and spiralling costs. It may be true that the percentage increase in phone costs has risen slower than for other services and products, but this is not reason to overlook the management control which can be exercised over telephone usage and costs by you. And today, the software is available to do the job.



```
                                                    COMMUNICATIONS ANALYSIS CORP                          *******************
                                                       USAGE CONTROL SYSTEM                               ACCOUNT CODE REPORT
                                                               FOR                                        *******************
                                                   *------SAMPLE  COMPANY-------*
                                                       CUSTOMER CODE  100
                                                                                         SVGS                                         CAC
               START        NUMBER  CALLED                                         LOST   VS                                          USE
DATE           TIME   AC   AREA EXT-LINE   DESTINATION      .  MINS   COST   LCR   SVGS   L.D.   CALLED NUMBER I.D.   ACCT EXTN    ONLY
------------   -----  --  --------------  ----------------- ----  -----  -----  -----  -----  ------------------   ---- ----    ---
FRI 23 JAN81   15:35  66   312-329-5500    CHICAGO      IL   3.1   1.10          0.00   0.00                        2000 1493    5
FRI 30 JAN81   16:02  67   812-636-8764    GREENSBURG   IN   3.7   1.31          0.00   0.00                        2000 1494    5
WED 4 FEB81    14:40  66   513-866-6521    MBG W CRTN   OH  13.2   4.69          0.00   0.00                        2000 1493    5
MON 9 FEB81    15:07  66   513-866-6521    MBG W CRTN   OH   7.6   2.70          0.00   0.00                        2000 1493    5
FRI 13 FEB81   10:34  67   513-273-3800    CINCINNATI   OH    .7   0.25          0.00   0.00                        2000 1493    5
FRI 13 FEB81   12:57  66   513-866-6521    MBG W CRTN   OH   1.3   0.46          0.00   0.00                        2000 1493    5
FRI 13 FEB81   13:19  67   213-573-2332    COMPTON      CA    .9   0.32          0.00   0.00                        2000 1493    5
FRI 13 FEB81   13:51  67   213-573-2332    COMPTON      CA   1.1   0.39          0.00   0.00                        2000 1493    5
TUE 17 FEB81   15:58  66   213-573-2332    COMPTON      CA   1.2   0.43          0.00   0.00                        2000 1493    5
TUE 17 FEB81   16:40  67   213-573-2332    COMPTON      CA   1.4   0.50          0.00   0.00                        2000 1493    5
WED 18 FEB81   10:48  66   213-573-2332    COMPTON      CA   3.8   1.35          0.00   0.00                        2000 1493    5
THU 19 FEB81   09:21  66   513-866-6521    MBG W CRTN   OH   2.9   1.03          0.00   0.00                        2000 1493    5
THU 19 FEB81   13:26  66   513-866-6521    MBG W CRTN   OH   1.8   0.64          0.00   0.00                        2000 1493    5
THU 19 FEB81   13:40  66   513-866-6521    MBG W CRTN   OH    .9   0.32          0.00   0.00                        2000 1493    5
THU 19 FEB81   14:28  66   513-866-6521    MBG W CRTN   OH   4.7   1.67          0.00   0.00                        2000 1493    5
FRI 20 FEB81   08:41  67   513-273-3800    CINCINNATI   OH   2.3   0.82          0.00   0.00                        2000 1493    5
//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
*SUB TOT*      X      X    X               X                50.6  17.98

*PLEASE NOTE:  This report is not included as part of the standard report package, but is available upon request.
```



```
 COMMUNICATIONS ANALYSIS CORP
     USAGE CONTROL SYSTEM
            FOR                                                                            ***************************
*--------SAMPLE  COMPANY-------*                     ACCOUNT CODE SUMMARY                  ACCOUNT CODE SUMMARY REPORT
                                                     ********************                  ***************************
            ACCOUNT CODE               NUMBER OF CALLS               TOTAL MINUTES         TOTAL COST
            ------------               ---------------               -------------         ----------
            0 324 001                       56                           165.2              $  47.97
            1 324 001                        7                            28.1              $   5.60
            1 324 190                       12                            41.4              $  10.06
            1 324 999                       16                            48.9              $  12.27
                                                                                           ---------
                                                                                           $  75.90

            0 446 001                       21                            59.2              $  18.17
            0 446 999                        4                            12.0              $   2.57
            1 446 001                        7                            36.5              $   7.22
            1 446 070                       36                            99.9              $  20.17
            1 446 335                        1                             4.6              $    .23
            1 446 999                       11                            43.3              $   8.87
                                                                                           --------
                                                                                           $  57.23

            0 787 001                        6                            19.8              $   4.07
            0 787 002                        1                             3.8              $    .36
            1 787 001                        9                            37.3              $   8.46
            1 787 099                        2                            17.1              $   5.55
            1 787 999                       13                            39.9              $   7.20
                                                                                           --------
                                                                                           $  25.64

            0 922 001                        1                             5.4              $    .81
            0 922 999                        4                            17.7              $   3.98
            1 922 001                        3                             8.1              $   1.35
            1 922 500                       36                           141.0              $  35.35
            1 922 999                       21                            69.0              $  18.57
                                                                                           --------
                                                                                           $  60.06
```

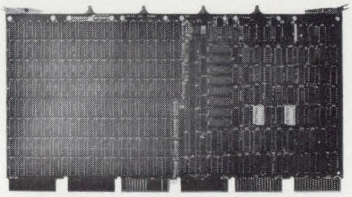
NOW YOU BENEFIT FROM OUR EXPERTISE!

4 YEAR WARRANTY

DEC COMPATIBLE ADD-IN MEMORIES

Yes, 4 Years! Only TRENDATA (Standard Memories) with over 15 years of experience in designing and manufacturing memories, can make this offer.

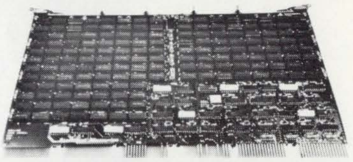
LOOK AT WHAT YOU HAVE TO CHOOSE FROM:



DEC PDP-11/44

PINCOMM 44S

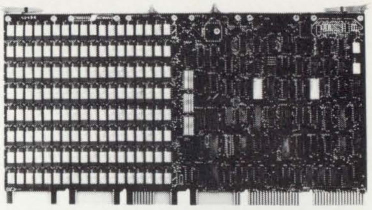
A Megabyte of ECC memory on a single card. Also available in 512KB, 256KB and 128KB increments. Also compatible with diagnostics and operating systems of Regular or Extended Unibus in systems other than PDP-11/44.



DEC PDP-11/70

PINCOMM 70S

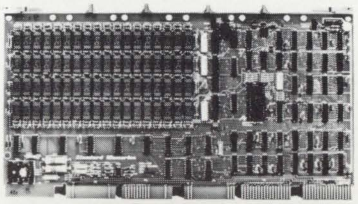
Pin compatible with PDP-11/70 CPUs using the MK-11 memory system. Offered in 256KB increments.



DEC PDP-11/24

PINCOMM 24S

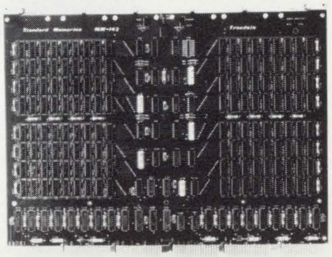
Parity memory. Offered in 1024KB, 512KB, 256KB and 128KB on a single card. Also compatible with Regular or Extended Unibus in systems other than PDP-11/24.



DEC PDP-11

PINCOMM PS

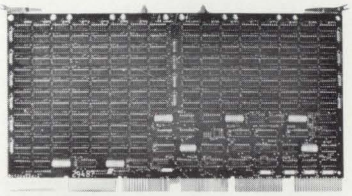
Offered in 128KB, 96KB and 32KB increments. Available with internal parity generation and checking and CSR registers.



DEC VAX-11/780

PINCOMM 780S

A direct replacement for DEC MS780Dx memory. Trendata D780S Memory Diagnostic available on Floppy Disk.



DEC VAX-11/750

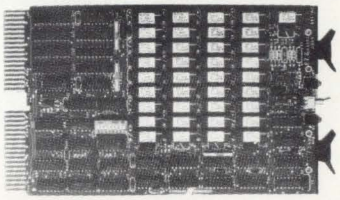
PINCOMM 750S

Replacement for DEC MS750Ax memory. Offered in 256KB increments.

All memories have socketed RAMs, on-board spare RAMs, on line/off line switch, LED indicators.

★ NEW — PINCOMM 23S

256k-byte parity memory for DEC LSI-11/23 microcomputers - Extended Q-bus addressing



On Site maintenance available in major Metropolitan areas

FOR FULL DETAILS CALL TODAY

TOLL FREE: 800-854-3792/IN CALIF: 800-432-7271

STANDARD MEMORIES DIVISION

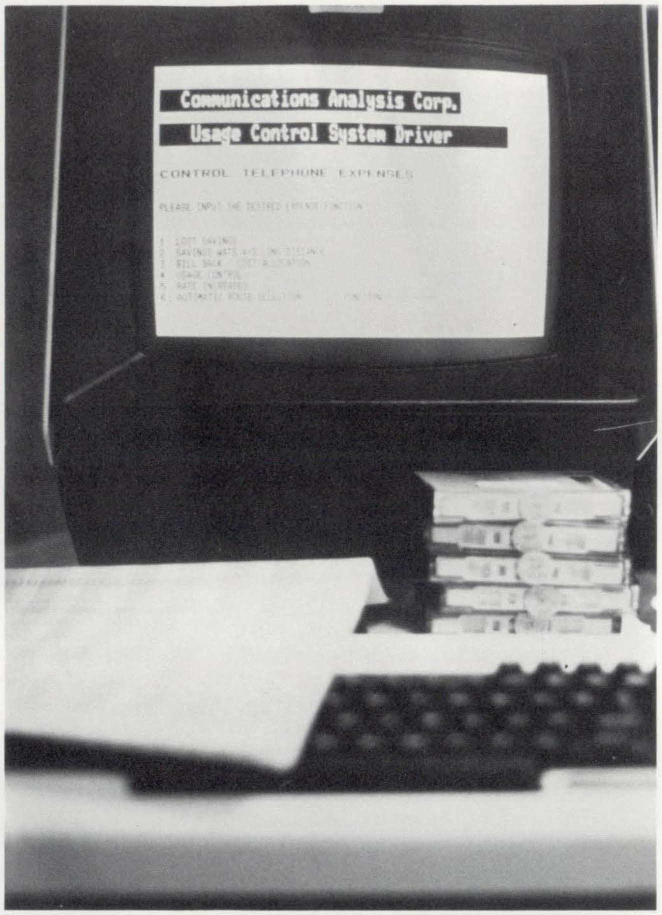
Trendata
CORPORATION

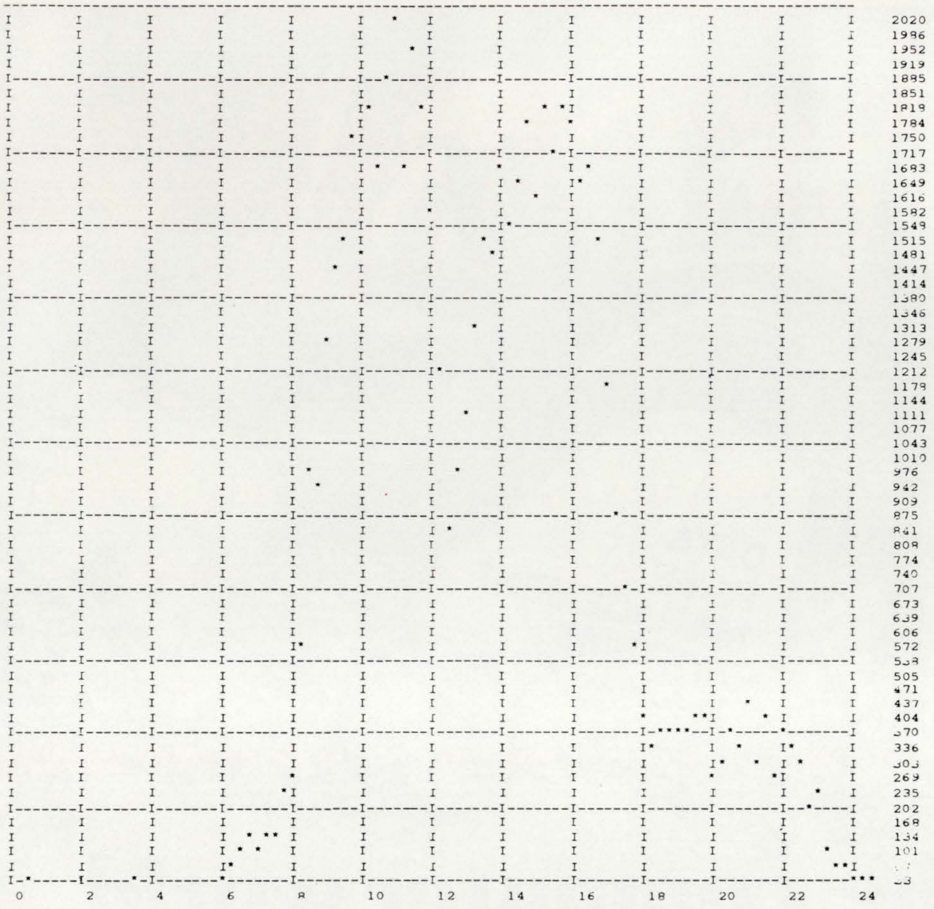
3400 W. SEGERSTROM AVE.
SANTA ANA, CALIFORNIA 92704
(714) 540-3605/TWX 910-595-1596

Digital Equipment Corporation (DEC) and its model designators are recognized registered trademarks.

CIRCLE 89 ON READER CARD

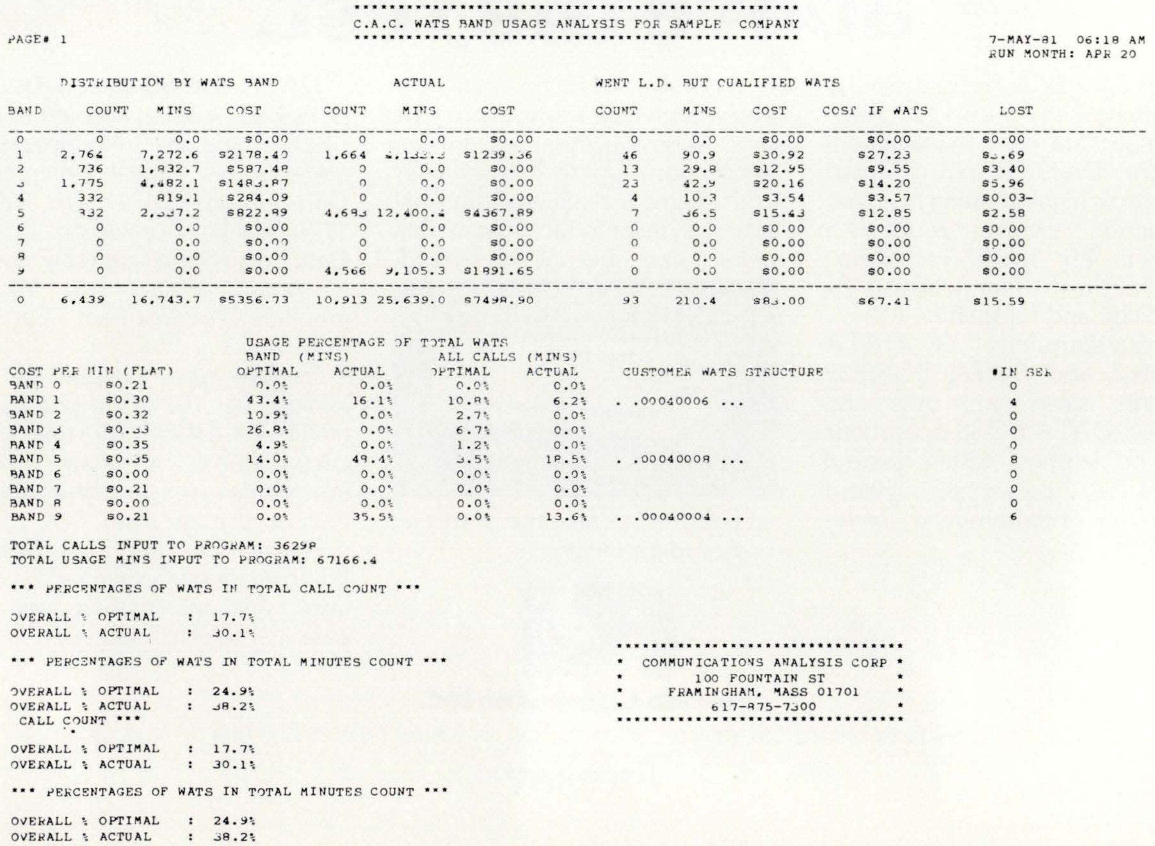
```
FREQUENTLY CALLED NUMBERS                                                                  RUN MONTH: APR 20
***********************                                                                    COMPUTER RUN: 7-MAY-81
 HIGH USAGE CALLS REPORT
 ***********************


CALLED NUMBER       IDENTIFICATION                NO. OF CALLS

312 333 1400        #                                 211
617 481 3640        #                                 196
404 748 4450        #                                 188
617 852 3000        #                                 183
617 839 5821        #                                 167
617 757 3295        #                                 141
516 466 5310        #                                 125
617 852 1000         STATE MUTUAL                     113
411                 #                                 113
617 852 5485        #                                 106
617 753 1411         WPI                              104
617 791 2272         WORC CTY INS SAVIN               100
617 852 0600         FALLON CLINIC                     96
617 886 6805        #                                  88
617 852 1050         ADVANCED BEARING                  88
617 757 7659        #                                  81
617 752 2876        #                                  79
617 791 0961        #                                  79
617 757 8306         FOUR SEASONS TRAVE                78
216 341 1700        #                                  72
617 791 7861        #                                  72
617 829 5401        #                                  70
617 791 2251        #                                  70
617 791 3121        #                                  68
617 839 5300        #                                  65
714 546 2090        #                                  63
617 757 5631        #                                  62
401 421 4901        #                                  61
617 853 1000         NORTON CO.                        60
617 755 4321        #                                  60
617 839 5475        #                                  60
617 752 1955        #                                  59
617 753 7225         WASHBURN GARFIELD                 58
213 981 3740        #                                  54
617 839 5382        #                                  53
617 798 8151        #                                  53
617 366 8911         DATA GENERAL                      52
617 839 6911        #                                  52
617 757 7751        #                                  52
617 832 6511        #                                  50
617 485 9263        #                                  49
513 271 5100        #                                  47
617 791 3657         WEATHER                           45
815 725 9500        #                                  37
617 756 8311         RM ELECTRONICS                    36
617 832 5801        #                                  36
617 852 4000         SHERATON LINCOLN                  35
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                    TOTAL CALLS                      6164
```


Communications Analysis Corp.
Usage Control System Driver
CONTROL TELEPHONE EXPENSES
PLEASE INPUT THE DESIRED EXPENSE FUNCTION
1 LOST SAVINGS
2 SAVINGS WATS/NTS LONG DISTANCE
3 BILL BACK / COST ALLOCATION
4 USAGE CONTROL
5 RATE INCREASES
6 AUTOMATIC ROUTE SELECTION



```
*****************
OUTGOING  TRAFFIC
DISTRIBUTION GRAPH
*****************

HORIZ AXIS = MILITARY
  TIME IN 15 MIN
  SEGMENTS
VERTICAL AXIS =
  TOTAL MINUTES USED
  DURING MONTH ON
  ALL FACILITIES

CONNECT STARS
  SEQUENTIALLY
  TO PRODUCE CURVE
```

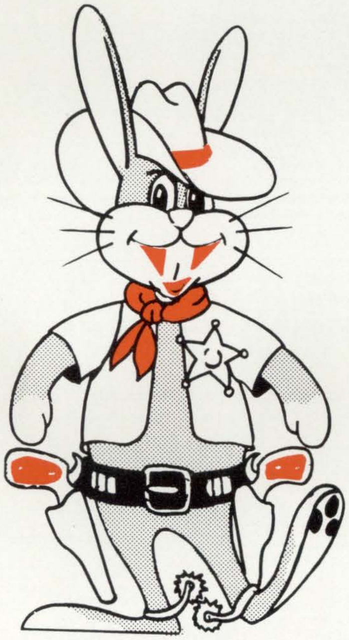
-Busy Hour Graph FOR SAMPLE COMPANY-



```
                                   ****************************************************
                                   C.A.C. WATS BAND USAGE ANALYSIS FOR SAMPLE  COMPANY
PAGE# 1                            ****************************************************                    7-MAY-81  06:18 AM
                                                                                                           RUN MONTH: APR 20

      DISTRIBUTION BY WATS BAND               ACTUAL                    WENT L.D. BUT QUALIFIED WATS

BAND      COUNT    MINS     COST       COUNT    MINS      COST          COUNT     MINS     COST    COST IF WATS        LOST
-----------------------------------------------------------------------------------------------------------------------------
 0          0       0.0    $0.00         0       0.0     $0.00            0        0.0    $0.00       $0.00          $0.00
 1      2,764   7,272.6  $2178.40     1,664  2,132.3  $1239.36           46       90.9   $30.92      $27.23          $3.69
 2        736   1,832.7   $587.48        0       0.0     $0.00           13       29.8   $12.95       $9.55          $3.40
 3      1,775   4,482.1  $1483.87        0       0.0     $0.00           23       42.9   $20.16      $14.20          $5.96
 4        332     819.1   $284.09        0       0.0     $0.00            4       10.3    $3.54       $3.57          $0.03-
 5        332   2,337.2   $822.89     4,683 12,400.4  $4367.89            7       36.5   $15.43      $12.85          $2.58
 6          0       0.0    $0.00         0       0.0     $0.00            0        0.0    $0.00       $0.00          $0.00
 7          0       0.0    $0.00         0       0.0     $0.00            0        0.0    $0.00       $0.00          $0.00
 8          0       0.0    $0.00         0       0.0     $0.00            0        0.0    $0.00       $0.00          $0.00
 9          0       0.0    $0.00     4,566  9,105.3  $1891.65            0        0.0    $0.00       $0.00          $0.00
-----------------------------------------------------------------------------------------------------------------------------
 0      6,439  16,743.7  $5356.73    10,913 25,639.0  $7498.90           93      210.4   $83.00      $67.41         $15.59

                          USAGE PERCENTAGE OF TOTAL WATS
                          BAND  (MINS)          ALL CALLS (MINS)
COST PER MIN (FLAT)       OPTIMAL    ACTUAL     OPTIMAL    ACTUAL     CUSTOMER WATS STRUCTURE                       #IN SER
BAND 0      $0.21            0.0%      0.0%       0.0%       0.0%                                                      0
BAND 1      $0.30           43.4%     16.1%      10.8%       6.2%    . .00040006 ...                                   4
BAND 2      $0.32           10.9%      0.0%       2.7%       0.0%                                                      0
BAND 3      $0.33           26.8%      0.0%       6.7%       0.0%                                                      0
BAND 4      $0.35            4.9%      0.0%       1.2%       0.0%                                                      0
BAND 5      $0.35           14.0%     49.4%       3.5%      18.5%    . .00040008 ...                                   8
BAND 6      $0.00            0.0%      0.0%       0.0%       0.0%                                                      0
BAND 7      $0.21            0.0%      0.0%       0.0%       0.0%                                                      0
BAND 8      $0.00            0.0%      0.0%       0.0%       0.0%                                                      0
BAND 9      $0.21            0.0%     35.5%       0.0%      13.6%    . .00040004 ...                                   6

TOTAL CALLS INPUT TO PROGRAM: 36298
TOTAL USAGE MINS INPUT TO PROGRAM: 67166.4

*** PERCENTAGES OF WATS IN TOTAL CALL COUNT ***

OVERALL % OPTIMAL    :  17.7%
OVERALL % ACTUAL     :  30.1%
                                                                ********************************
*** PERCENTAGES OF WATS IN TOTAL MINUTES COUNT ***              *  COMMUNICATIONS ANALYSIS CORP *
                                                                *         100 FOUNTAIN ST       *
OVERALL % OPTIMAL    :  24.9%                                   *     FRAMINGHAM, MASS 01701    *
OVERALL % ACTUAL     :  38.2%                                   *         617-875-7300          *
 CALL COUNT ***                                                 ********************************

OVERALL % OPTIMAL    :  17.7%
OVERALL % ACTUAL     :  30.1%

*** PERCENTAGES OF WATS IN TOTAL MINUTES COUNT ***

OVERALL % OPTIMAL    :  24.9%
OVERALL % ACTUAL     :  38.2%
```

RABBIT-4 FILE SECURITY GUNS DOWN DATA RUSTLERS ON RSTS/E SYSTEMS

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



RABBIT-4 will help you track'm and catch'm, cause it has a proven record of performance.

RABBIT-4 will let you:

□ Log secured file accesses

□ Signal OPSER of violations

□ Roll-out the bandits

□ Freeze system activities with

6 levels of file security to keep your data safe and secure, **RABBIT-4** will also:

- Secure up to 64 data files
- Provide 32 user descriptions plus wild cards
- Restrict file access to specified programs
- Identify intrusions and intruders
- Recover automatically from system crashes

SEE OUR TALENTED RABBITS AT THE ATLANTA DECUS MEETING

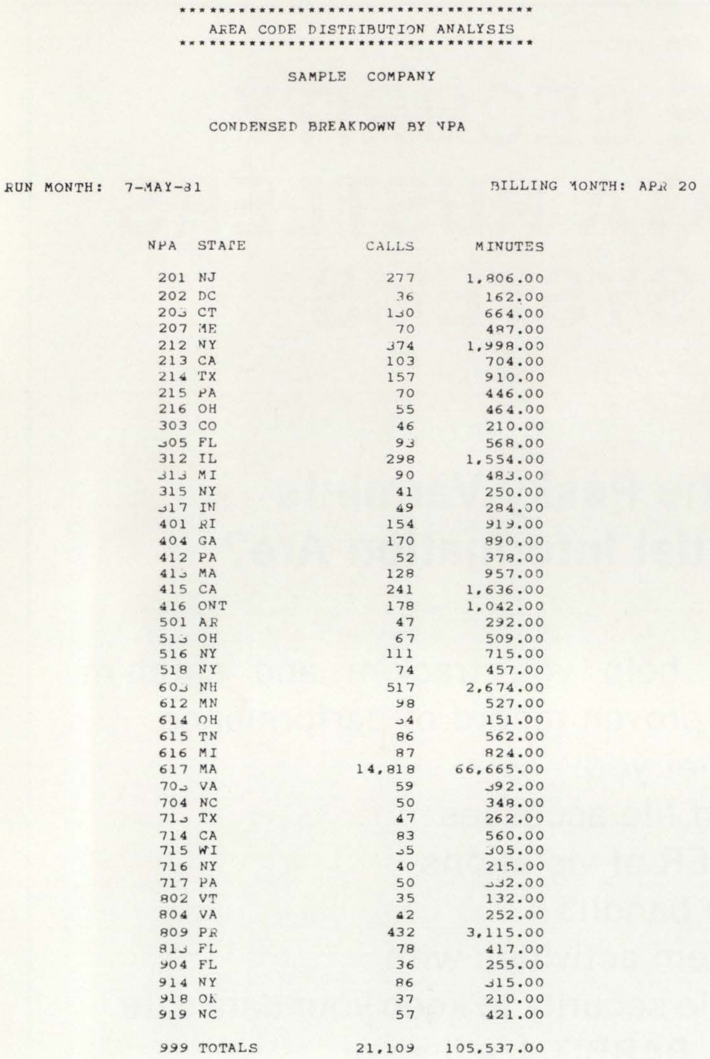
and pick up a free Rabbit Executive Coloring Book Souvenir!
We'll have a hospitality suite where we'll demonstrate
Rabbit's resourcefulness in

* Performance Analysis * Job Accounting * Resource Accounting
* Data Management * Financial Planning * File Security

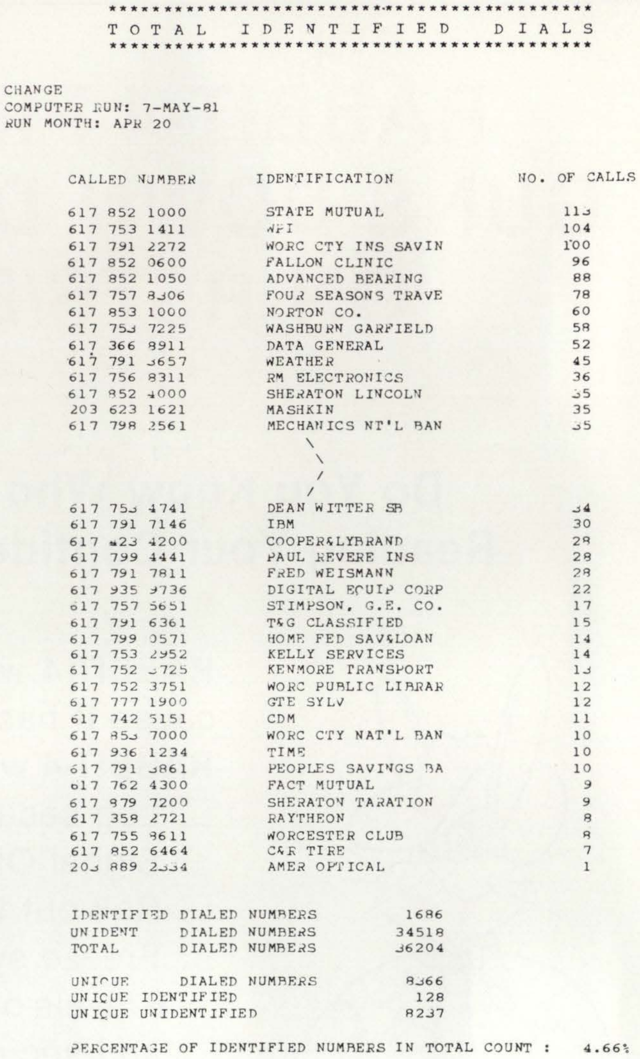
RAXCO INC.

RAXCO Inc., Suite 200, 6520 Powers Ferry Road, Atlanta, GA 30339 • (404) 955-2553

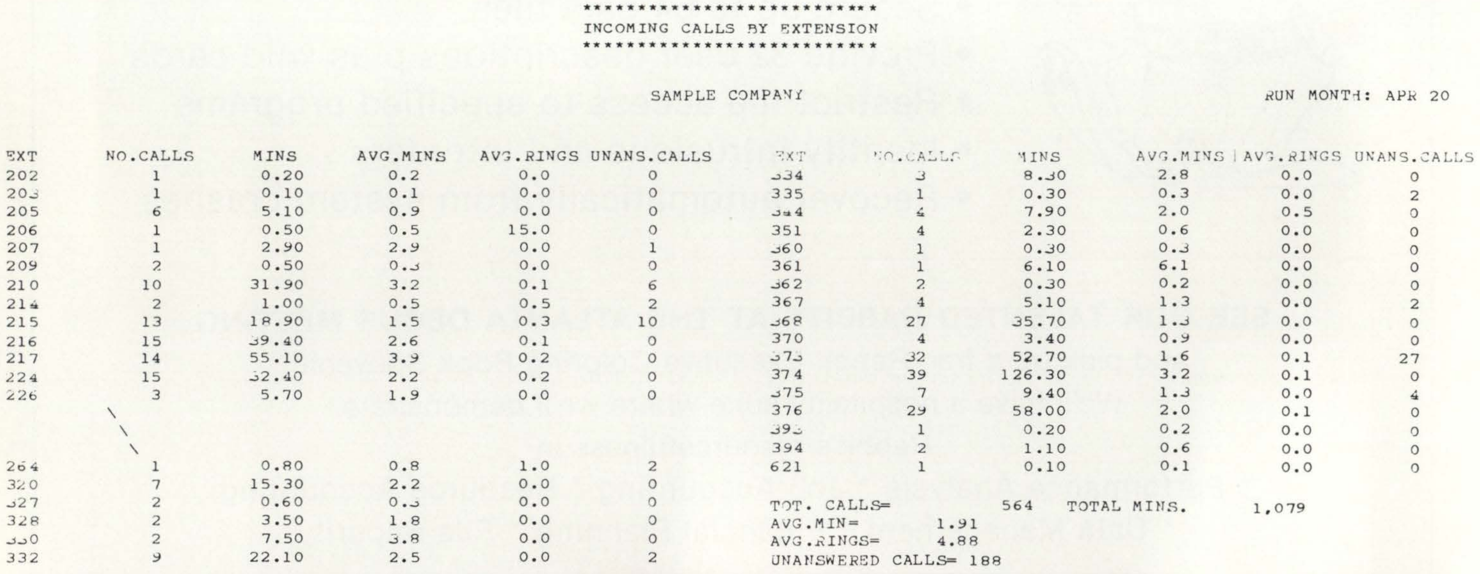
CIRCLE 21 ON READER CARD



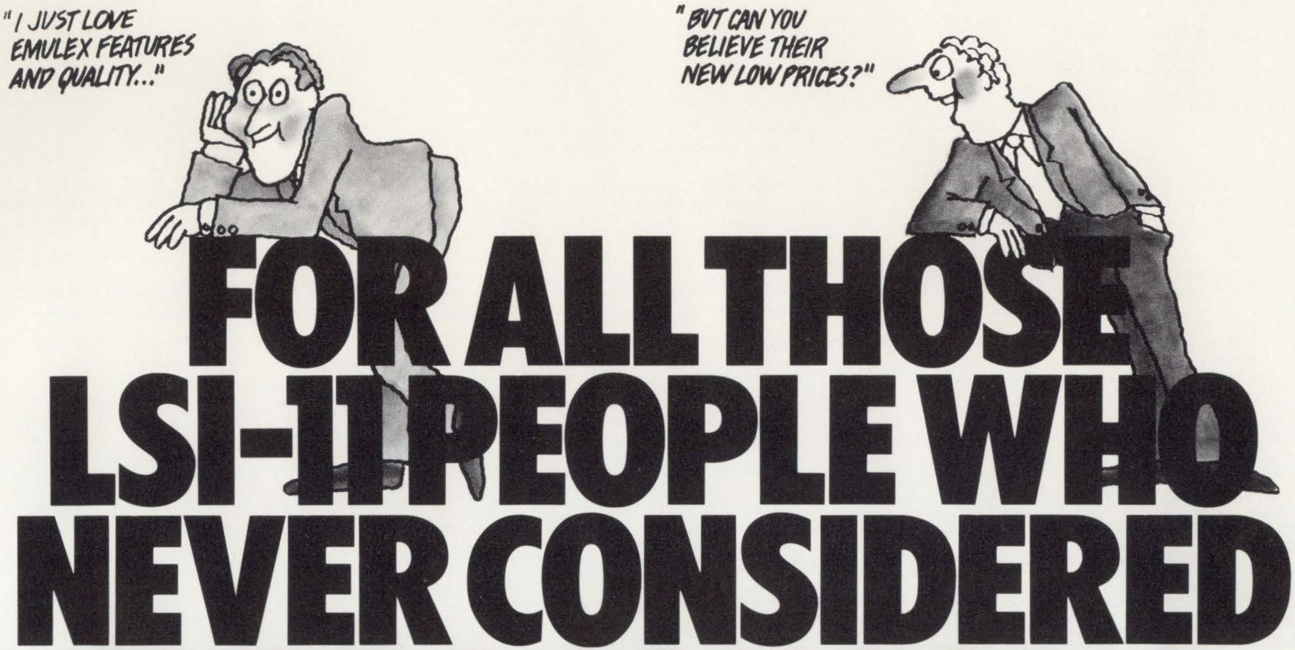
```
                     ***********************************
                      AREA CODE DISTRIBUTION ANALYSIS
                     ***********************************

                               SAMPLE  COMPANY


                      CONDENSED BREAKDOWN BY NPA


RUN MONTH:  7-MAY-81                                   BILLING MONTH: APR 20


             NPA  STATE               CALLS        MINUTES

              201 NJ                    277       1,806.00
              202 DC                     36         162.00
              203 CT                    130         664.00
              207 ME                     70         487.00
              212 NY                    374       1,998.00
              213 CA                    103         704.00
              214 TX                    157         910.00
              215 PA                     70         446.00
              216 OH                     55         464.00
              303 CO                     46         210.00
              305 FL                     93         568.00
              312 IL                    298       1,554.00
              313 MI                     90         483.00
              315 NY                     41         250.00
              317 IN                     49         284.00
              401 RI                    154         919.00
              404 GA                    170         890.00
              412 PA                     52         378.00
              413 MA                    128         957.00
              415 CA                    241       1,636.00
              416 ONT                   178       1,042.00
              501 AR                     47         292.00
              513 OH                     67         509.00
              516 NY                    111         715.00
              518 NY                     74         457.00
              603 NH                    517       2,674.00
              612 MN                     98         527.00
              614 OH                     34         151.00
              615 TN                     86         562.00
              616 MI                     87         824.00
              617 MA                 14,818      66,665.00
              703 VA                     59         392.00
              704 NC                     50         348.00
              713 TX                     47         262.00
              714 CA                     83         560.00
              715 WI                     35         305.00
              716 NY                     40         215.00
              717 PA                     50         332.00
              802 VT                     35         132.00
              804 VA                     42         252.00
              809 PR                    432       3,115.00
              813 FL                     78         417.00
              904 FL                     36         255.00
              914 NY                     86         315.00
              918 OK                     37         210.00
              919 NC                     57         421.00

              999 TOTALS             21,109     105,537.00
```



```
      ************************************************
      T O T A L    I D E N T I F I E D    D I A L S
      ************************************************

CHANGE
COMPUTER RUN: 7-MAY-81
RUN MONTH: APR 20


    CALLED NUMBER      IDENTIFICATION               NO. OF CALLS

    617 852 1000        STATE MUTUAL                     113
    617 753 1411        WPI                              104
    617 791 2272        WORC CTY INS SAVIN               100
    617 852 0600        FALLON CLINIC                     96
    617 852 1050        ADVANCED BEARING                  88
    617 757 8306        FOUR SEASONS TRAVE                78
    617 853 1000        NORTON CO.                        60
    617 753 7225        WASHBURN GARFIELD                 58
    617 366 8911        DATA GENERAL                      52
    617 791 3657        WEATHER                           45
    617 756 8311        RM ELECTRONICS                    36
    617 852 4000        SHERATON LINCOLN                  35
    203 623 1621        MASHKIN                           35
    617 798 2561        MECHANICS NT'L BAN                35
                          \
                           \
                           /
                          /
    617 753 4741        DEAN WITTER SB                    34
    617 791 7146        IBM                               30
    617 423 4200        COOPER&LYBRAND                    28
    617 799 4441        PAUL REVERE INS                   28
    617 791 7811        FRED WEISMANN                     28
    617 935 9736        DIGITAL EQUIP CORP                22
    617 757 5651        STIMPSON, G.E. CO.                17
    617 791 6361        T&G CLASSIFIED                    15
    617 799 0571        HOME FED SAV&LOAN                 14
    617 753 2952        KELLY SERVICES                    14
    617 752 3725        KENMORE TRANSPORT                 13
    617 752 3751        WORC PUBLIC LIBRAR                12
    617 777 1900        GTE SYLV                          12
    617 742 5151        CDM                               11
    617 853 7000        WORC CTY NAT'L BAN                10
    617 936 1234        TIME                              10
    617 791 3861        PEOPLES SAVINGS BA                10
    617 762 4300        FACT MUTUAL                        9
    617 879 7200        SHERATON TARATION                  9
    617 358 2721        RAYTHEON                           8
    617 755 8611        WORCESTER CLUB                     8
    617 852 6464        C&R TIRE                           7
    203 889 2334        AMER OPTICAL                       1


    IDENTIFIED DIALED NUMBERS          1686
    UNIDENT    DIALED NUMBERS         34518
    TOTAL      DIALED NUMBERS         36204

    UNIQUE     DIALED NUMBERS          8366
    UNIQUE IDENTIFIED                   128
    UNIQUE UNIDENTIFIED                8237

    PERCENTAGE OF IDENTIFIED NUMBERS IN TOTAL COUNT :   4.66%
```



```
                                        *************************
                                        INCOMING CALLS BY EXTENSION
                                        *************************


                                            SAMPLE COMPANY                                                 RUN MONTH: APR 20


EXT      NO.CALLS      MINS     AVG.MINS   AVG.RINGS UNANS.CALLS     EXT      NO.CALLS      MINS       AVG.MINS | AVG.RINGS UNANS.CALLS
202          1         0.20       0.2         0.0        0           334          3          8.30         2.8         0.0         0
203          1         0.10       0.1         0.0        0           335          1          0.30         0.3         0.0         2
205          6         5.10       0.9         0.0        0           344          4          7.90         2.0         0.5         0
206          1         0.50       0.5        15.0        0           351          4          2.30         0.6         0.0         0
207          1         2.90       2.9         0.0        1           360          1          0.30         0.3         0.0         0
209          2         0.50       0.3         0.0        0           361          1          6.10         6.1         0.0         0
210         10        31.90       3.2         0.1        6           362          2          0.30         0.2         0.0         0
214          2         1.00       0.5         0.5        2           367          4          5.10         1.3         0.0         2
215         13        18.90       1.5         0.3       10           368         27         35.70         1.3         0.0         0
216         15        39.40       2.6         0.1        0           370          4          3.40         0.9         0.0         0
217         14        55.10       3.9         0.2        0           373         32         52.70         1.6         0.1        27
224         15        32.40       2.2         0.2        0           374         39        126.30         3.2         0.1         0
226          3         5.70       1.9         0.0        0           375          7          9.40         1.3         0.0         4
          \                                                          376         29         58.00         2.0         0.1         0
           \                                                         393          1          0.20         0.2         0.0         0
            \                                                        394          2          1.10         0.6         0.0         0
264          1         0.80       0.8         1.0        2           621          1          0.10         0.1         0.0         0
320          7        15.30       2.2         0.0        0
327          2         0.60       0.3         0.0        0           TOT. CALLS=            564    TOTAL MINS.       1,079
328          2         3.50       1.8         0.0        0           AVG.MIN=        1.91
330          2         7.50       3.8         0.0        0           AVG.RINGS=      4.88
332          9        22.10       2.5         0.0        2           UNANSWERED CALLS= 188
```


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

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

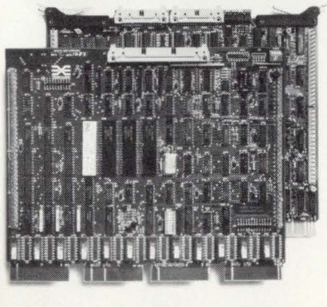
Consider the products and prices below:

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

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

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

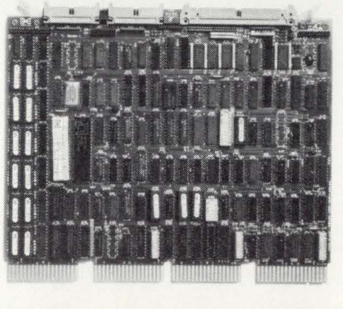
SC01 (RM02/05, RP06) $2528*



Put big SMD drives on your LSI-11.

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

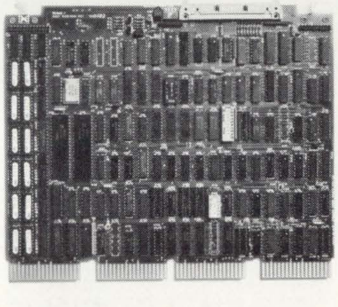
SC02 (RL01/02, RP02/03) $1600*
SC02 (RK06/07) $1792*



Low cost for smaller-sized disks.

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

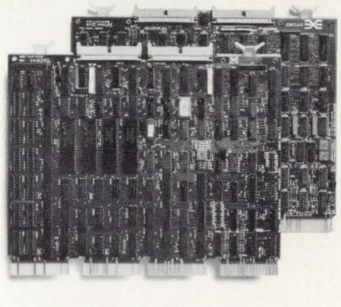
SC04 (RL01/02) $1504*
SC04 (RK06/07) $1696*



New! ANSI interfacing for 8" Winchesters.

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

TC01 (NRZ) $1536*
TC01 (PE) $1920*



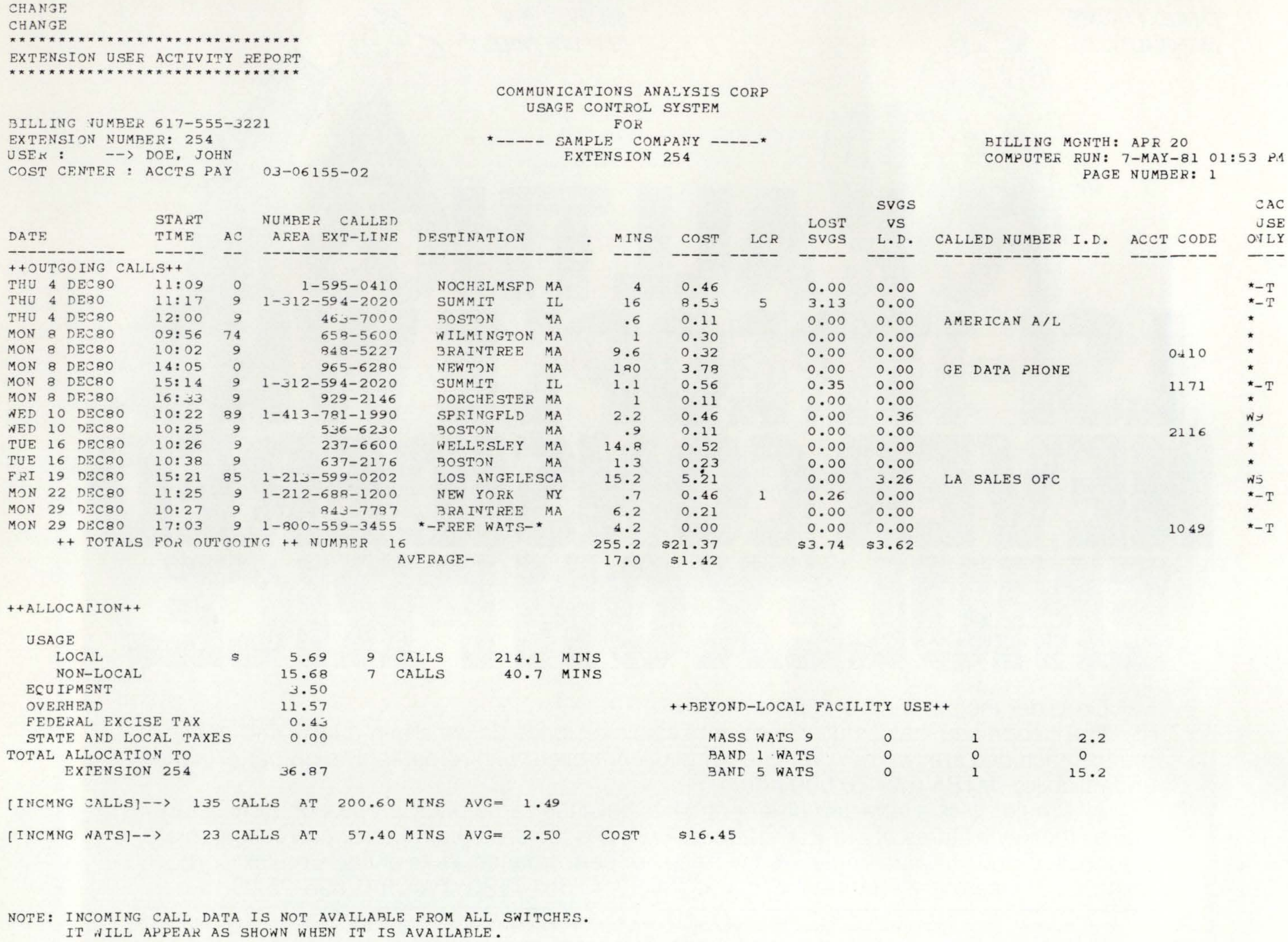
Q-bus embedded dual-density tape controller.

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

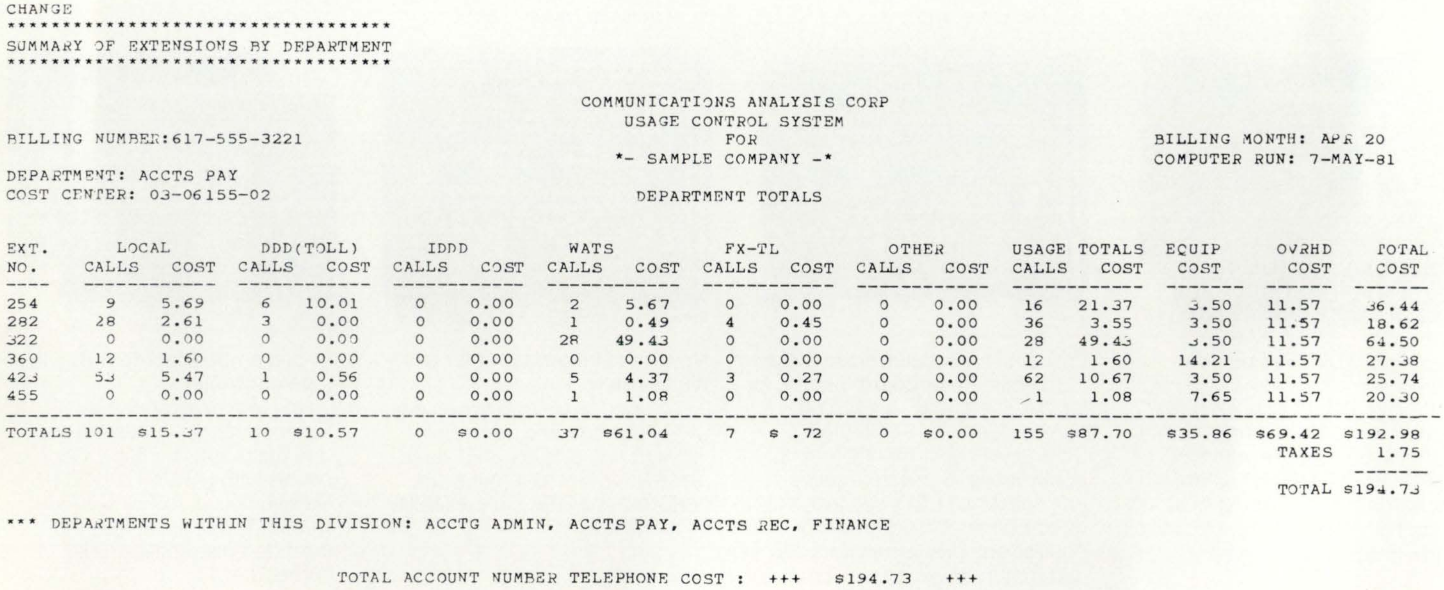
EMULEX
The genuine alternative

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

CIRCLE 58 ON READER CARD



```
CHANGE
CHANGE
******************************
EXTENSION USER ACTIVITY REPORT
******************************
                                                    COMMUNICATIONS ANALYSIS CORP
                                                       USAGE CONTROL SYSTEM
BILLING NUMBER 617-555-3221                                     FOR
EXTENSION NUMBER: 254                               *----- SAMPLE  COMPANY -----*                 BILLING MONTH: APR 20
USER :      --> DOE, JOHN                                   EXTENSION 254                          COMPUTER RUN: 7-MAY-81 01:53 PM
COST CENTER : ACCTS PAY     03-06155-02                                                                     PAGE NUMBER: 1

                                                                                         SVGS                                     CAC
                START          NUMBER  CALLED                                      LOST   VS                                      USE
DATE            TIME   AC   AREA EXT-LINE   DESTINATION        .  MINS  COST   LCR  SVGS  L.D.  CALLED NUMBER I.D.  ACCT CODE   ONLY
------------    -----  --  -------------  -------------------  ----  -----  -----  -----  -----  ------------------  ---------   ---
++OUTGOING CALLS++
THU  4 DEC80    11:09   0       1-595-0410   NOCHELMSFD MA       4   0.46          0.00   0.00                                  *-T
THU  4 DE80     11:17   9   1-312-594-2020   SUMMIT     IL      16   8.53     5    3.13   0.00                                  *-T
THU  4 DEC80    12:00   9         463-7000   BOSTON     MA      .6   0.11          0.00   0.00  AMERICAN A/L                    *
MON  8 DEC80    09:56  74         658-5600   WILMINGTON MA       1   0.30          0.00   0.00                                  *
MON  8 DEC80    10:02   9         848-5227   BRAINTREE  MA     9.6   0.32          0.00   0.00                        0410      *
MON  8 DEC80    14:05   0         965-6280   NEWTON     MA     180   3.78          0.00   0.00  GE DATA PHONE                   *
MON  8 DEC80    15:14   9   1-312-594-2020   SUMMIT     IL     1.1   0.56          0.35   0.00                        1171      *-T
MON  8 DEC80    16:33   9         929-2146   DORCHESTER MA       1   0.11          0.00   0.00                                  *
WED 10 DEC80    10:22  89   1-413-781-1990   SPRINGFLD  MA     2.2   0.46          0.00   0.36                                  W9
WED 10 DEC80    10:25   9         536-6230   BOSTON     MA      .9   0.11          0.00   0.00                        2116      *
TUE 16 DEC80    10:26   9         237-6600   WELLESLEY  MA    14.8   0.52          0.00   0.00                                  *
TUE 16 DEC80    10:38   9         637-2176   BOSTON     MA     1.3   0.23          0.00   0.00                                  *
FRI 19 DEC80    15:21  85   1-213-599-0202   LOS ANGELESCA    15.2   5.21          0.00   3.26  LA SALES OFC                    W5
MON 22 DEC80    11:25   9   1-212-688-1200   NEW YORK   NY      .7   0.46     1    0.26   0.00                                  *-T
MON 29 DEC80    10:27   9         843-7787   BRAINTREE  MA     6.2   0.21          0.00   0.00                                  *
MON 29 DEC80    17:03   9   1-800-559-3455  *-FREE WATS-*      4.2   0.00          0.00   0.00                        1049      *-T
      ++ TOTALS FOR OUTGOING ++ NUMBER  16                   255.2 $21.37         $3.74  $3.62
                                          AVERAGE-            17.0  $1.42


++ALLOCATION++

  USAGE
     LOCAL                 $      5.69    9  CALLS      214.1  MINS
     NON-LOCAL                   15.68    7  CALLS       40.7  MINS
  EQUIPMENT                       3.50
  OVERHEAD                       11.57                                 ++BEYOND-LOCAL FACILITY USE++
  FEDERAL EXCISE TAX              0.43
  STATE AND LOCAL TAXES           0.00                                     MASS WATS 9       0          1          2.2
TOTAL ALLOCATION TO                                                        BAND 1 WATS       0          0          0
      EXTENSION 254              36.87                                     BAND 5 WATS       0          1         15.2

[INCMNG CALLS]-->  135 CALLS  AT  200.60 MINS  AVG=  1.49

[INCMNG WATS]-->    23 CALLS  AT   57.40 MINS  AVG=  2.50      COST     $16.45


NOTE: INCOMING CALL DATA IS NOT AVAILABLE FROM ALL SWITCHES.
      IT WILL APPEAR AS SHOWN WHEN IT IS AVAILABLE.
```



```
CHANGE
**********************************
SUMMARY OF EXTENSIONS BY DEPARTMENT
**********************************
                                                  COMMUNICATIONS ANALYSIS CORP
                                                     USAGE CONTROL SYSTEM
BILLING NUMBER:617-555-3221                                   FOR                                  BILLING MONTH: APR 20
                                                      *- SAMPLE COMPANY -*                         COMPUTER RUN: 7-MAY-81
DEPARTMENT: ACCTS PAY
COST CENTER: 03-06155-02                               DEPARTMENT TOTALS

EXT.      LOCAL       DDD(TOLL)       IDDD          WATS          FX-TL         OTHER       USAGE TOTALS   EQUIP    OVRHD    TOTAL
NO.    CALLS  COST   CALLS  COST   CALLS  COST   CALLS  COST   CALLS  COST   CALLS  COST   CALLS  COST    COST     COST     COST
----   -----------   -----------   -----------   -----------   -----------   -----------   -----------   -------  -------  -------
254       9   5.69      6  10.01      0   0.00      2   5.67      0   0.00      0   0.00     16  21.37     3.50    11.57    36.44
282      28   2.61      3   0.00      0   0.00      1   0.49      4   0.45      0   0.00     36   3.55     3.50    11.57    18.62
322       0   0.00      0   0.00      0   0.00     28  49.43      0   0.00      0   0.00     28  49.43     3.50    11.57    64.50
360      12   1.60      0   0.00      0   0.00      0   0.00      0   0.00      0   0.00     12   1.60    14.21    11.57    27.38
423      53   5.47      1   0.56      0   0.00      5   4.37      3   0.27      0   0.00     62  10.67     3.50    11.57    25.74
455       0   0.00      0   0.00      0   0.00      1   1.08      0   0.00      0   0.00      1   1.08     7.65    11.57    20.30
-----------------------------------------------------------------------------------------------------------------------------------
TOTALS 101 $15.37     10 $10.57      0  $0.00     37 $61.04      7  $ .72      0  $0.00    155 $87.70   $35.86   $69.42  $192.98
                                                                                                                  TAXES     1.75
                                                                                                                          ------
                                                                                                                  TOTAL  $194.73

*** DEPARTMENTS WITHIN THIS DIVISION: ACCTG ADMIN, ACCTS PAY, ACCTS REC, FINANCE


                         TOTAL ACCOUNT NUMBER TELEPHONE COST :  +++   $194.73   +++
```

LETTERS to the RSTS Pro . . .

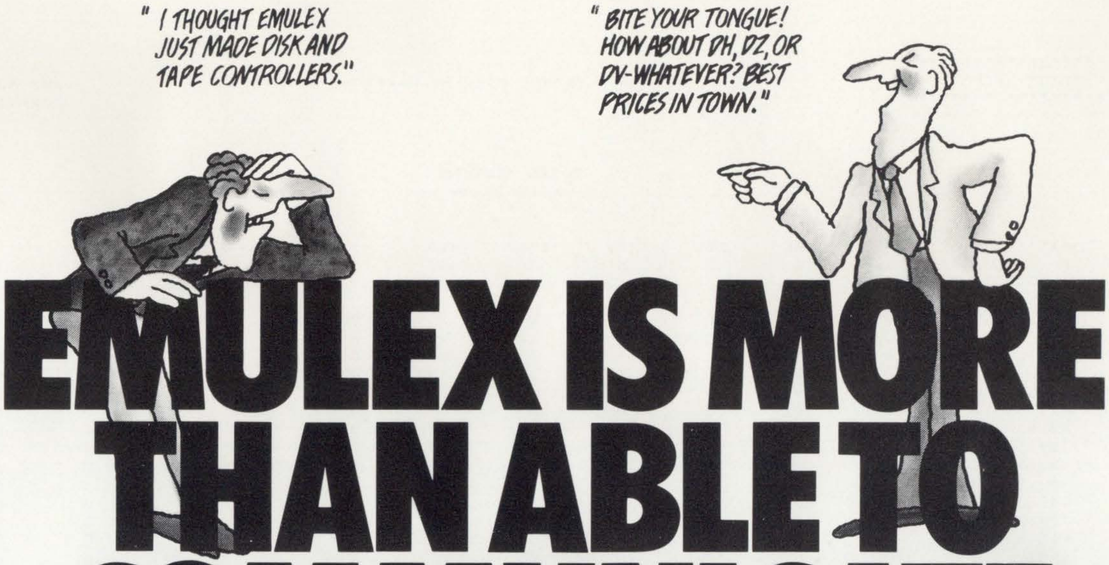
. . . is your column! Send us your comments, suggestions, or notes of interest to the RSTS community. We'd enjoy hearing from you.

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

There's subscription cards for you or a friend.

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

There's a READERS INQUIRY card for your convenience.


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

EMULEX IS MORE THAN ABLE TO COMMUNICATE WITH DEC.

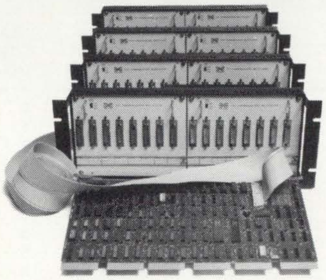
We're also able to save you plenty: For instance, you get DH11 performance for a DZ11 price. Four new space-saving single-board communications multiplexers. And an increase in VAX-11 terminal handling capacity by up to 50%. Maintained nationwide by Control Data.

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

Communicate with Emulex now. Write or call Emulex Corp., 2001 Deere Ave., Santa Ana, CA 92705; (714) 577-7580, TWX 910-595-2521.

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

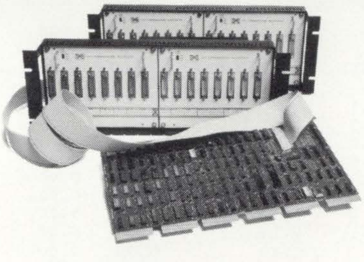
CS11/H (PDP-11) $7560 for 48 lines*
CS11/U (VAX-11) $7884 for 48 lines*



Up to 64 DH11 channels from one board.

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

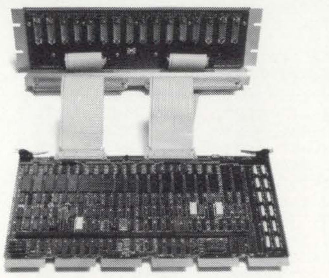
CS11/V $4464 for 16 lines*



Higher DV11 performance, lower price.

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

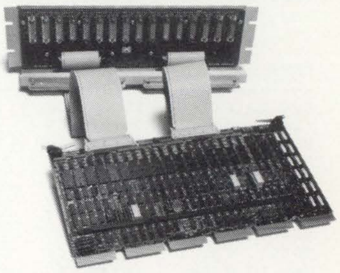
CS21/Z $2520*



Replace DEC DZ11/E and save.

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

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



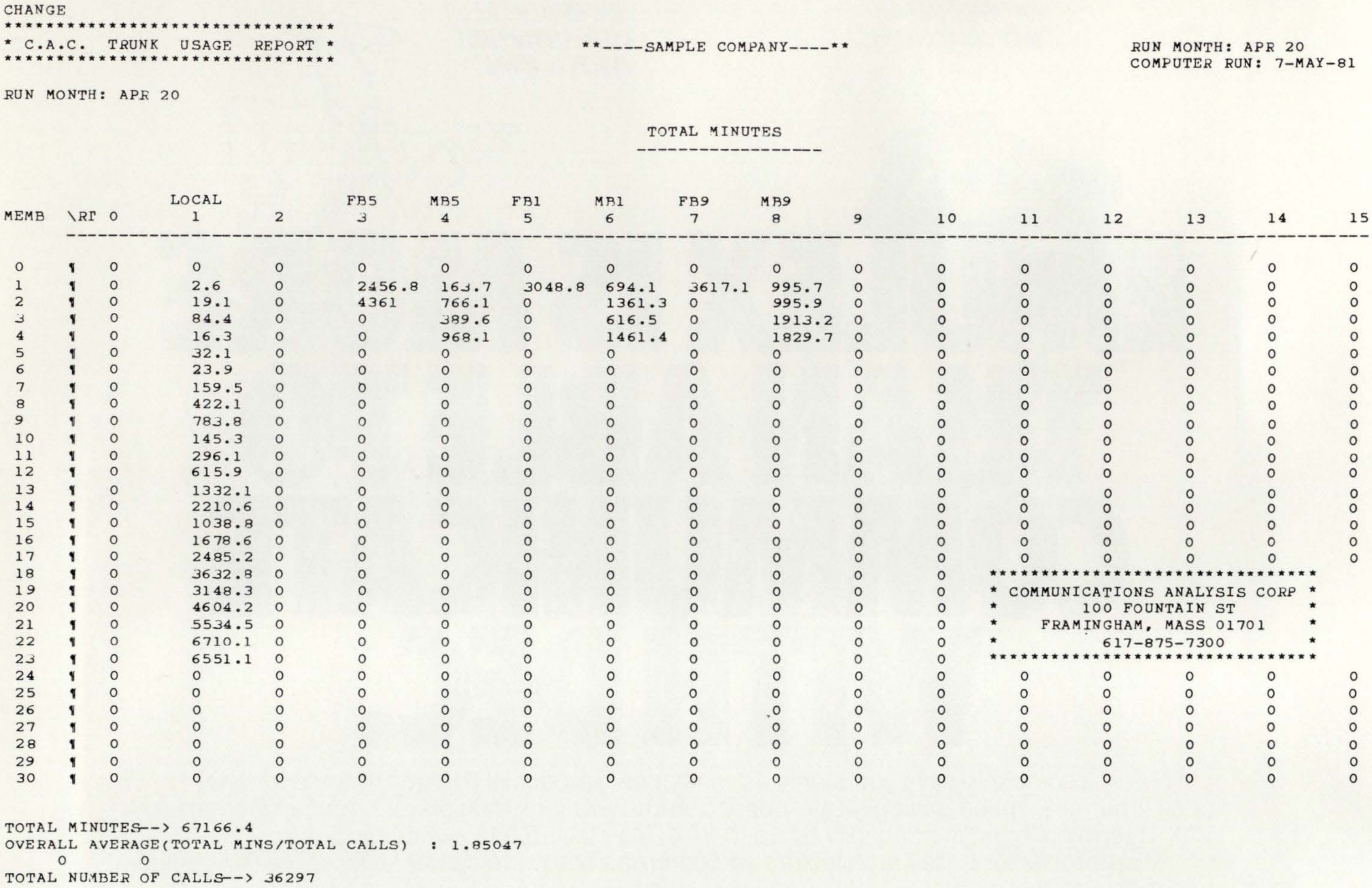
New economical DH11-type multiplexer.

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

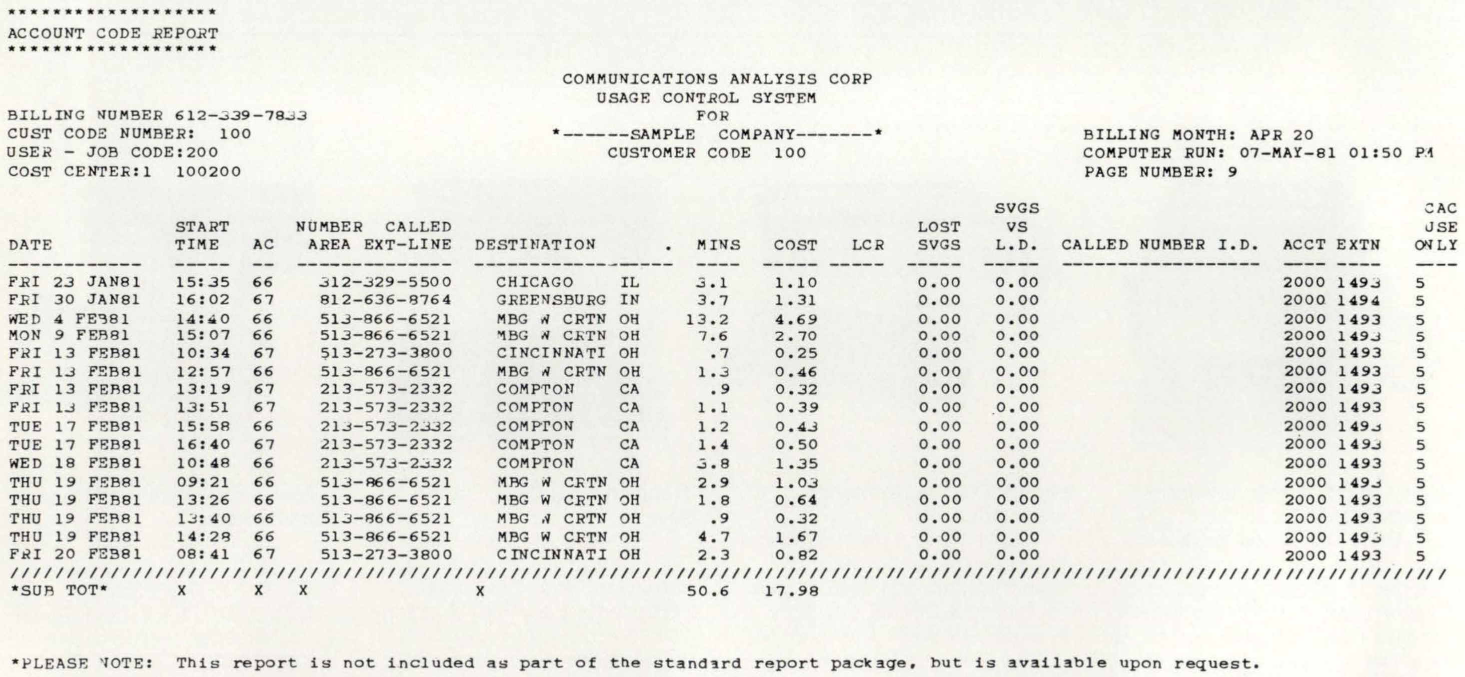
EMULEX
The genuine alternative

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

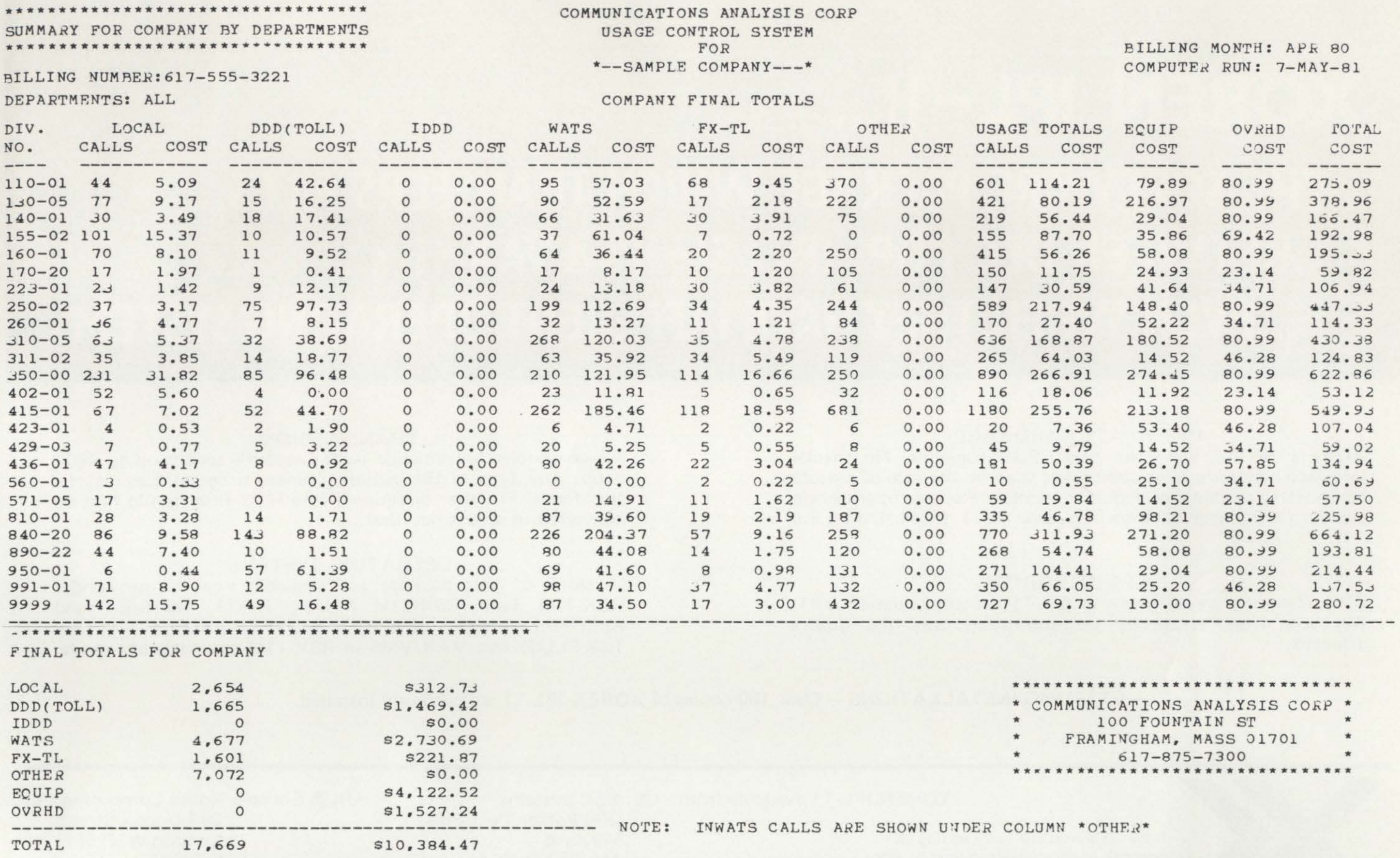
CIRCLE 93 ON READER CARD



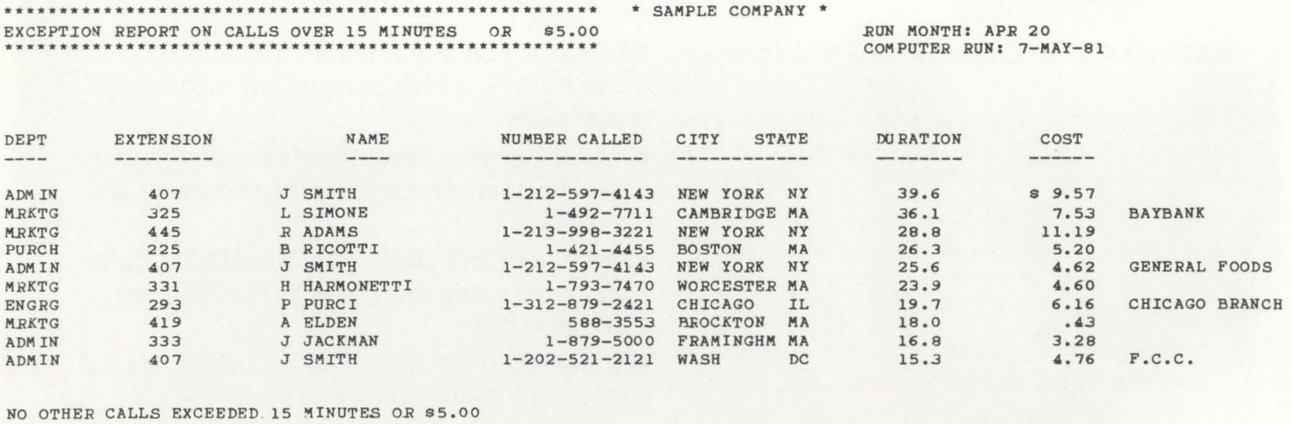
```
CHANGE
********************************
* C.A.C.  TRUNK  USAGE  REPORT *                     **----SAMPLE COMPANY----**                        RUN MONTH: APR 20
********************************                                                                       COMPUTER RUN: 7-MAY-81

RUN MONTH: APR 20

                                                        TOTAL MINUTES
                                                     -----------------


              LOCAL        FB5      MB5      FB1      MB1      FB9      MB9
MEMB \RT 0    1        2   3        4        5        6        7        8        9     10    11    12    13    14    15
-------------------------------------------------------------------------------------------------------------------------

  0  ¶   0    0        0   0        0        0        0        0        0        0     0     0     0     0     0     0
  1  ¶   0    2.6      0   2456.8   163.7    3048.8   694.1    3617.1   995.7    0     0     0     0     0     0     0
  2  ¶   0    19.1     0   4361     766.1    0        1361.3   0        995.9    0     0     0     0     0     0     0
  3  ¶   0    84.4     0   0        389.6    0        616.5    0        1913.2   0     0     0     0     0     0     0
  4  ¶   0    16.3     0   0        968.1    0        1461.4   0        1829.7   0     0     0     0     0     0     0
  5  ¶   0    32.1     0   0        0        0        0        0        0        0     0     0     0     0     0     0
  6  ¶   0    23.9     0   0        0        0        0        0        0        0     0     0     0     0     0     0
  7  ¶   0    159.5    0   0        0        0        0        0        0        0     0     0     0     0     0     0
  8  ¶   0    422.1    0   0        0        0        0        0        0        0     0     0     0     0     0     0
  9  ¶   0    783.8    0   0        0        0        0        0        0        0     0     0     0     0     0     0
 10  ¶   0    145.3    0   0        0        0        0        0        0        0     0     0     0     0     0     0
 11  ¶   0    296.1    0   0        0        0        0        0        0        0     0     0     0     0     0     0
 12  ¶   0    615.9    0   0        0        0        0        0        0        0     0     0     0     0     0     0
 13  ¶   0    1332.1   0   0        0        0        0        0        0        0     0     0     0     0     0     0
 14  ¶   0    2210.6   0   0        0        0        0        0        0        0     0     0     0     0     0     0
 15  ¶   0    1038.8   0   0        0        0        0        0        0        0     0     0     0     0     0     0
 16  ¶   0    1678.6   0   0        0        0        0        0        0        0     0     0     0     0     0     0
 17  ¶   0    2485.2   0   0        0        0        0        0        0        0     0     0     0     0     0     0
 18  ¶   0    3632.8   0   0        0        0        0        0        0        0     0     ********************************
 19  ¶   0    3148.3   0   0        0        0        0        0        0        0     0     * COMMUNICATIONS ANALYSIS CORP *
 20  ¶   0    4604.2   0   0        0        0        0        0        0        0     0     *       100 FOUNTAIN ST        *
 21  ¶   0    5534.5   0   0        0        0        0        0        0        0     0     *   FRAMINGHAM, MASS 01701     *
 22  ¶   0    6710.1   0   0        0        0        0        0        0        0     0     *       617-875-7300           *
 23  ¶   0    6551.1   0   0        0        0        0        0        0        0     0     ********************************
 24  ¶   0    0        0   0        0        0        0        0        0        0     0     0     0     0     0     0
 25  ¶   0    0        0   0        0        0        0        0        0        0     0     0     0     0     0     0
 26  ¶   0    0        0   0        0        0        0        0        0        0     0     0     0     0     0     0
 27  ¶   0    0        0   0        0        0        0        0        0        0     0     0     0     0     0     0
 28  ¶   0    0        0   0        0        0        0        0        0        0     0     0     0     0     0     0
 29  ¶   0    0        0   0        0        0        0        0        0        0     0     0     0     0     0     0
 30  ¶   0    0        0   0        0        0        0        0        0        0     0     0     0     0     0     0

TOTAL MINUTES--> 67166.4
OVERALL AVERAGE(TOTAL MINS/TOTAL CALLS)  : 1.85047
    0       0
TOTAL NUMBER OF CALLS--> 36297
```



```
*******************
ACCOUNT CODE REPORT
*******************

                                                   COMMUNICATIONS ANALYSIS CORP
                                                      USAGE CONTROL SYSTEM
BILLING NUMBER 612-339-7833                                    FOR
CUST CODE NUMBER:  100                             *------SAMPLE  COMPANY-------*                  BILLING MONTH: APR 20
USER - JOB CODE:200                                    CUSTOMER CODE  100                          COMPUTER RUN: 07-MAY-81 01:50 PM
COST CENTER:1  100200                                                                              PAGE NUMBER: 9

                                                                                     SVGS                                          CAC
              START      NUMBER  CALLED                                      LOST     VS                                           USE
DATE          TIME   AC   AREA EXT-LINE  DESTINATION     .  MINS   COST   LCR  SVGS    L.D.   CALLED NUMBER I.D.  ACCT EXTN     ONLY
------------  -----  --  --------------  ----------------- ----   -----  ----- -----   -----  ------------------  ---- ----     ----
FRI 23 JAN81  15:35  66    312-329-5500    CHICAGO     IL    3.1   1.10        0.00    0.00                       2000 1493     5
FRI 30 JAN81  16:02  67    812-636-8764    GREENSBURG  IN    3.7   1.31        0.00    0.00                       2000 1494     5
WED 4 FEB81   14:40  66    513-866-6521    MBG W CRTN  OH   13.2   4.69        0.00    0.00                       2000 1493     5
MON 9 FEB81   15:07  66    513-866-6521    MBG W CRTN  OH    7.6   2.70        0.00    0.00                       2000 1493     5
FRI 13 FEB81  10:34  67    513-273-3800    CINCINNATI  OH     .7   0.25        0.00    0.00                       2000 1493     5
FRI 13 FEB81  12:57  66    513-866-6521    MBG W CRTN  OH    1.3   0.46        0.00    0.00                       2000 1493     5
FRI 13 FEB81  13:19  67    213-573-2332    COMPTON     CA     .9   0.32        0.00    0.00                       2000 1493     5
FRI 13 FEB81  13:51  67    213-573-2332    COMPTON     CA    1.1   0.39        0.00    0.00                       2000 1493     5
TUE 17 FEB81  15:58  66    213-573-2332    COMPTON     CA    1.2   0.43        0.00    0.00                       2000 1493     5
TUE 17 FEB81  16:40  67    213-573-2332    COMPTON     CA    1.4   0.50        0.00    0.00                       2000 1493     5
WED 18 FEB81  10:48  66    213-573-2332    COMPTON     CA    3.8   1.35        0.00    0.00                       2000 1493     5
THU 19 FEB81  09:21  66    513-866-6521    MBG W CRTN  OH    2.9   1.03        0.00    0.00                       2000 1493     5
THU 19 FEB81  13:26  66    513-866-6521    MBG W CRTN  OH    1.8   0.64        0.00    0.00                       2000 1493     5
THU 19 FEB81  13:40  66    513-866-6521    MBG W CRTN  OH     .9   0.32        0.00    0.00                       2000 1493     5
THU 19 FEB81  14:28  66    513-866-6521    MBG W CRTN  OH    4.7   1.67        0.00    0.00                       2000 1493     5
FRI 20 FEB81  08:41  67    513-273-3800    CINCINNATI  OH    2.3   0.82        0.00    0.00                       2000 1493     5
/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
*SUB TOT*     X      X  X                  X                50.6  17.98


*PLEASE NOTE:  This report is not included as part of the standard report package, but is available upon request.
```



```
*********************************           COMMUNICATIONS ANALYSIS CORP
SUMMARY FOR COMPANY BY DEPARTMENTS              USAGE CONTROL SYSTEM
*********************************                     FOR                                 BILLING MONTH: APR 80
                                               *--SAMPLE COMPANY---*                      COMPUTER RUN: 7-MAY-81
BILLING NUMBER:617-555-3221

DEPARTMENTS: ALL                               COMPANY FINAL TOTALS

DIV.      LOCAL        DDD(TOLL)       IDDD           WATS          FX-TL          OTHER       USAGE TOTALS   EQUIP    OVRHD    TOTAL
NO.    CALLS   COST  CALLS   COST  CALLS   COST  CALLS   COST  CALLS   COST  CALLS   COST  CALLS   COST    COST     COST     COST
----   ------------  ------------  ------------  ------------  ------------  ------------  ------------  -------  ------   ------
110-01  44     5.09    24   42.64    0     0.00    95   57.03    68    9.45   370    0.00   601  114.21    79.89   80.99   275.09
130-05  77     9.17    15   16.25    0     0.00    90   52.59    17    2.18   222    0.00   421   80.19   216.97   80.99   378.96
140-01  30     3.49    18   17.41    0     0.00    66   31.63    30    3.91    75    0.00   219   56.44    29.04   80.99   166.47
155-02 101    15.37    10   10.57    0     0.00    37   61.04     7    0.72     0    0.00   155   87.70    35.86   69.42   192.98
160-01  70     8.10    11    9.52    0     0.00    64   36.44    20    2.20   250    0.00   415   56.26    58.08   80.99   195.33
170-20  17     1.97     1    0.41    0     0.00    17    8.17    10    1.20   105    0.00   150   11.75    24.93   23.14    59.82
223-01  23     1.42     9   12.17    0     0.00    24   13.18    30    3.82    61    0.00   147   30.59    41.64   34.71   106.94
250-02  37     3.17    75   97.73    0     0.00   199  112.69    34    4.35   244    0.00   589  217.94   148.40   80.99   447.33
260-01  36     4.77     7    8.15    0     0.00    32   13.27    11    1.21    84    0.00   170   27.40    52.22   34.71   114.33
310-05  63     5.37    32   38.69    0     0.00   268  120.03    35    4.78   238    0.00   636  168.87   180.52   80.99   430.38
311-02  35     3.85    14   18.77    0     0.00    63   35.92    34    5.49   119    0.00   265   64.03    14.52   46.28   124.83
350-00 231    31.82    85   96.48    0     0.00   210  121.95   114   16.66   250    0.00   890  266.91   274.45   80.99   622.86
402-01  52     5.60     4    0.00    0     0.00    23   11.81     5    0.65    32    0.00   116   18.06    11.92   23.14    53.12
415-01  67     7.02    52   44.70    0     0.00  . 262  185.46   118   18.59   681    0.00  1180  255.76   213.18   80.99   549.93
423-01   4     0.53     2    1.90    0     0.00     6    4.71     2    0.22     6    0.00    20    7.36    53.40   46.28   107.04
429-03   7     0.77     5    3.22    0     0.00    20    5.25     5    0.55     8    0.00    45    9.79    14.52   34.71    59.02
436-01  47     4.17     8    0.92    0     0.00    80   42.26    22    3.04    24    0.00   181   50.39    26.70   57.85   134.94
560-01   3     0.33     0    0.00    0     0.00     0    0.00     2    0.22     5    0.00    10    0.55    25.10   34.71    60.69
571-05  17     3.31     0    0.00    0     0.00    21   14.91    11    1.62    10    0.00    59   19.84    14.52   23.14    57.50
810-01  28     3.28    14    1.71    0     0.00    87   39.60    19    2.19   187    0.00   335   46.78    98.21   80.99   225.98
840-20  86     9.58   143   88.82    0     0.00   226  204.37    57    9.16   258    0.00   770  311.93   271.20   80.99   664.12
890-22  44     7.40    10    1.51    0     0.00    80   44.08    14    1.75   120    0.00   268   54.74    58.08   80.99   193.81
950-01   6     0.44    57   61.39    0     0.00    69   41.60     8    0.98   131    0.00   271  104.41    29.04   80.99   214.44
991-01  71     8.90    12    5.28    0     0.00    98   47.10    37    4.77   132    0.00   350   66.05    25.20   46.28   137.53
9999   142    15.75    49   16.48    0     0.00    87   34.50    17    3.00   432    0.00   727   69.73   130.00   80.99   280.72
----------------------------------------------------------------------------------------------------------------------------------
-*************************************************
FINAL TOTALS FOR COMPANY

LOCAL              2,654              $312.73                                 *********************************
DDD(TOLL)          1,665            $1,469.42                                 * COMMUNICATIONS ANALYSIS CORP *
IDDD                   0                $0.00                                 *       100 FOUNTAIN ST         *
WATS               4,677            $2,730.69                                 *    FRAMINGHAM, MASS 01701     *
FX-TL              1,601              $221.87                                 *        617-875-7300           *
OTHER              7,072                $0.00                                 *********************************
EQUIP                  0            $4,122.52
OVRHD                  0            $1,527.24
--------------------------------------------------------  NOTE:  INWATS CALLS ARE SHOWN UNDER COLUMN *OTHER*
TOTAL             17,669           $10,384.47
```



```
*******************************************************     * SAMPLE COMPANY *
EXCEPTION REPORT ON CALLS OVER 15 MINUTES   OR   $5.00                               RUN MONTH: APR 20
*******************************************************                              COMPUTER RUN: 7-MAY-81


DEPT      EXTENSION          NAME         NUMBER CALLED    CITY   STATE      DURATION     COST
----      ---------          ----         -------------    ------------      --------     ------

ADMIN        407          J SMITH         1-212-597-4143   NEW YORK  NY        39.6      $ 9.57
MRKTG        325          L SIMONE            1-492-7711   CAMBRIDGE MA        36.1        7.53   BAYBANK
MRKTG        445          R ADAMS         1-213-998-3221   NEW YORK  NY        28.8       11.19
PURCH        225          B RICOTTI           1-421-4455   BOSTON    MA        26.3        5.20
ADMIN        407          J SMITH         1-212-597-4143   NEW YORK  NY        25.6        4.62   GENERAL FOODS
MRKTG        331          H HARMONETTI        1-793-7470   WORCESTER MA        23.9        4.60
ENGRG        293          P PURCI         1-312-879-2421   CHICAGO   IL        19.7        6.16   CHICAGO BRANCH
MRKTG        419          A ELDEN               588-3553   BROCKTON  MA        18.0         .43
ADMIN        333          J JACKMAN           1-879-5000   FRAMINGHM MA        16.8        3.28
ADMIN        407          J SMITH         1-202-521-2121   WASH      DC        15.3        4.76   F.C.C.

NO OTHER CALLS EXCEEDED 15 MINUTES OR $5.00
```

Back Issues are now available! Have a complete set of the RSTS PROFESSIONAL!
Every issue contains valuable information you will want to have on hand or to give away to a friend.
Check issues desired and send to: RSTS Professional, Back Issues, P.O. Box 361, Ft. Washington, PA 19034-0361.
Back Issues: $10⁰⁰ per issue if paid in advance, $12⁵⁰per issue if billed.

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

World's Biggest Exposition of DEC-Compatibles!

DEXPO™ 82

The First National DEC-Compatible Industry Exposition

Marriott Hotel, Atlanta May 10-12, 1982

Get The Most Out of Your DEC System... Visit the Only Show Big Enough for All Your DEC-Directed Needs...

Now there's a show dedicated to making your DEC system better. At DEXPO 82 every exhibitor will be demonstrating products and services that are compatible with your DEC equipment. That means starting today there's no longer a need to visit any other show, because DEXPO 82 features more DEC-compatibles than you'll find at any other event in the world.

DECUS Conference Registrants: Your Atlanta schedule is not complete until you visit DEXPO 82 — only a few blocks from your meetings!

- Compare thousands of the newest DEC-compatible products and services — many on exhibit for the first time anywhere — all under one roof.
- Select alternatives, enhancements and special solutions from an *international* group of vendors. Choose the best technologies in the world.
- Attend free "Product Forums" demonstrating the latest DEC-compatibles. Discover new capabilities, applications and ideas to improve or expand your system.
- Meet more than 100 vendors of DEC-compatible software, hardware and related services and supplies, including:

Accessories • Applications Software • Books • Bus Links • Consulting Services • Controllers • Data Entry Systems • Data Communications Equipment • Data Preparation Systems • Data Handling Systems • Disk Drives • Educational Services • Enhancements • Floppy Disk Drives • Forms • Graphic Display Terminals • Magazines • Memory Products • Multiplexers • Modems • Optical Character Recognition Equipment • Periodicals • Peripherals • Power Supplies • Printers • Productivity Tools • Software • Software Services • Supplies • Systems Software • Tape Drives • Terminals • Training Services • Video Processors • Video Displays

DEC is a registered trademark of the Digital Equipment Corporation. DEXPO 82 is not affiliated with DECUS or the Digital Equipment Corporation.

Mail Today for FREE Information, Reduced-Rate Tickets and Hotel Discounts

□ Send ______ reduced-rate DEXPO 82 tickets for my associates and myself. Also send complete information on the Show.

□ Send information on discount hotel rates available to DEXPO 82 visitors.

Name ____________________ Title ______________

Company ______________________________

Address ___ ___________________________

City ____________________ State ________ Zip ______

Mail to: Expoconsul International, Inc.
19 Yeger Road
Cranbury, N.J. 08512
(Phone: 609-799-1661)

CIRCLE 94 ON READER CARD

DISK I/O FROM MACRO

By Bob "MACRO MAN" Meyer

The following article describes the use of some basic disk-related monitor calls, and a detailed example of their use.

The calls we'll be using include:

.FSS	Use the file string scanner
OPNFQ	Open an existing file for input
CREFQ	Create a new file
.READ	Read blocks from the input file
.WRITE	Write blocks to the output file
CLSFQ	Close a file

In order to use all of the above calls, the sample program will do the following:

1) Open '$UTILTY.HLP' for input (or any other file of your choice)
2) Create the output file 'OUTPUT.DAT' in the current account
3) Transfer blocks from the input file to the output file, watching for End of File
4) Close the output file

Let's examine each call in detail. The File String Scanner, or FSS, is a monitor function provided for interpreting various file name strings. The FSS routines will accept a file name, parse that name (quite throughly), and exit with the FIRQB setup for an open or close type function. The FSS call understands about many file name specifics, including Protection codes, Account specs, dollar signs, user assigned & system wide logicals, special switches to FIP (/mode:xx, /ro, /filesize:xx, etc...) and several other goodies. For more details on FSS, see your System Directives Manual, page 3-93.

So before opening the input file, we must pass the name & account specification through the file string scanner. This is shown in the example program just after clearing the FIRQB, under the symbol 10$:. To use the FSS, we simply pass (in the XRB) the length of the file name string, and its starting position. If no errors are detected, control is passed to the symbol 20$:. For the sake of simplicity, if any errors occur in the program, we'll just put the error code in RO and crash the program. The BPT instruction will cause the RSX emulator to crash the task and give us a register dump on the terminal (in octal). The first group of numbers will contain the RSTS error code; be sure to translate to decimal before attempting to understand it.

Assuming the FSS worked correctly, the FIRQB should be setup for the open function of FIP. Before calling FIP, we must specify that we want a file open function (OPENFQ) by placing the proper code in the FIRQB. All that is left is to specify the channel number (times two) as shown at symbol 20$:, and the CALFIP directive can be executed. If anything goes wrong here, such as a non-existent file or incorrect protection code, the program will crash at this point, again with the octal error code in RO.

Once the input file is opened, we can create the output file. As with the open for input function above, we must first run the file name through the file string scanner. This is done at symbol 40$:.

Now that the FIRQB is setup, we need to specify that we want to create a new file. This is done at symbol 50$: by moving the create function code (CREFQ) into the FIRQB as well as the channel number (times two) of the output file. From there FIP is called, and we check for errors.

If we get this far, the both files must be open. The next step is to begin transferring data. This is done using the .READ directive as shown at symbol 60$:. The parameters passed on a read are:

1) Number of bytes to read (must be a multiple of 512 for disk)
2) Where to put the data in our workspace
3) The channel number to read from
4) An optional block number to read (zero being sequential)
5) And any device-dependant modifiers (none needed here)

After reading the block, we should check for End of File. This is done by the CMPB (compare byte) instruction a few lines under the .READ; in our example, if no error occurs, we go write a block at 70$:, if error 11 occurs (End of File on device), we branch to a close routine. If any other unexpected errors occur, we crash the program.

Now that we've read a block of data, we have it in our buffer (BUFF), and we have the option of doing anything we please with it. Again, for simplicity, we're just going to move it to the output file.

So, at symbol 70$:, we load the XRB for a .WRITE monitor call. As in the .READ, we specify the buffer address, the buffer length, and the output channel. After the .WRITE we check for errors, and if all is well we branch back to the READ routine, continuing until End of File is reached.

Once we get to the End of File, we close it using the CLSFQ function of FIP. Since no errors are possible with CLSFQ, we can just exit to the system default run-time system.

This program is very self contained, so assembly & linkage is simple:

```
MAC  CREATE=CREATE
TKB  CREATE=CREATE
```

That will do it. Run create, and by using ↑T you can watch the operation of the program. When finished, you should find the file OUTPUT.DAT in your account, and it should be an exact copy of the input file (UTILTY.HLP in our case).

That's all for now; thanks for reading!

```
                        [1,10] CREATE.MAC


.title    create
.ident    /1.0/
.dsabl    gbl

;
;define everything
;
clsfq     =0                                  ;close function code
opnfq     =2                                  ;open function code
crefq     =4                                  ;create function code
```

```
outchn  =2*13.                              ;output channel
chan       =2*14.                           ;input channel

xrb        =442                             ;xrb
firqb      =402                             ;firqb

calfip     =104000                          ;call fip emt
.fss       =104064                          ;file string scanner
.read      =104002                          ;read directive
.exit      =104046                          ;exit to default rts
.write     =104004

file:      .ascii  /[1,2]utilty.hlp/        ;file to open
fillen     =.-file                          ;length of file name
newfil: .ascii /sy:output.dat/              ;output file name
newlen     =.-newfil                        ;length of output file name
.even

buff::     .blkb   512.                     ;disk buffer
buflen     =.-buff                          ;len of the buffer

;
;start of main code
;
create:    mov     #firqb,r0                ;clear the firqb
           mov     #16.,rl
10$:       clr     (r0)+
           sob     rl,10$

;
;run input file name through .fss
;
           mov     #xrb,r0                  ;point to xrb
           mov     #fillen,(r0)+            ;move in name length
           mov     #fillen,(r0)+            ;twice...
           mov     #file,(r0)+              ;move in the name
           clr     (r0)+
           clr     (r0)+
           clr     (r0)+
           clr     (r0)+
           .fss                             ;call the file string scanner
           tstb    @#firqb
           beq     20$
           mov     @#firqb,r0
           bpt

;
;open the input file
;
20$:       movb    #opnfq,@#firqb+3         ;ask for 'open for input' function
           movb    #chan,@#firqb+4          ;specify channel #
           calfip                           ;call fip to do the open
           tstb    @#firqb
           beq     30$
           mov     @#firqb,r0
           bpt

;
;fss the output file name
;
30$:       mov     #firqb,r0                ;clear the firqb
           mov     #16.,rl
40$:       clr     (r0)+
           sob     rl,40$

           mov     #xrb,r0                  ;point to xrb
           mov     #newlen,(r0)+            ;length of new file name
           mov     #newlen,(r0)+            ;length of new file name
```

```
           mov     #newfil,(r0)+            ;ouput file name
           clr     (r0)+
           clr     (r0)+
           clr     (r0)+
           clr     (r0)+
           .fss                             ;file name string scan
           tstb    @#firqb
           beq     50$
           mov     @#firqb,r0
           bpt

;
;create the output file
;
50$:       movb    #crefq,@#firqb+3         ;ask for create
           movb    #outchn,@#firqb+4        ;channel number of output file
           calfip                           ;call faithful fip...
           tstb    @#firqb
           beq     60$
           mov     @#firqb,r0
           bpt

;
;read a block from the input file
;
60$:       mov     #xrb,r0                  ;point to xrb
           mov     #buflen,(r0)+            ;move in buffer length
           clr     (r0)+                    ;must be 0
           mov     #buff,(r0)+              ;move in address of buffer
           mov     #chan,(r0)+              ;move in channel number
           clr     (r0)+                    ;block number to read
           clr     (r0)+                    ;wait time
           clr     (r0)+                    ;modifiers
           .read                            ;call the monitor
           tstb    @#firqb                  ;watch for errors
           beq     70$                      ;none
           cmpb    @#firqb,#11.             ;end of input file?
           beq     close                    ;yes; cleanup & exit
           mov     @#firqb,r0               ;else crash the program
           bpt

;
;write a block to the output file
;
70$:       mov     #xrb,r0                  ;point to xrb
           mov     #buflen,(r0)+            ;buf length
           mov     #buflen,(r0)+            ;buf length
           mov     #buff,(r0)+              ;point to address of buffer
           mov     #outchn,(r0)+            ;channel number to write to
           clr     (r0)+                    ;block number to write to
           clr     (r0)+                    ;
           clr     (r0)+                    ;modifers
           .write
           tstb    @#firqb
           beq     60$                      ;go read next block
           mov     @#firqb,r0
           bpt

;
;close the output file
;
close:     movb    #clsfq,@#firqb+3         ;close chan fun code
           mov     #outchn,@#firqb+4        ;chan to close
           calfip                           ;fip call to close

           .exit                            ;return to default rts

           .end    create
```

♥

GETTING THE MOST OUT OF YOUR DEC FIELD SERVICE

By Mark H. Deibert, Systems Manager, Minicomputer Services E. R. Squibb and Sons, Inc.

The ability of a RSTS System Manager (small shops) or Technical Support Manager (larger shops) to keep his/her RSTS system up and running usually requires more than an ability to deal with RSTS. Effective interaction with Digital's Field Service organization can be the difference between promotability and the need to hastily update one's resume.

In four years of dealing with over nine different DEC Field Service branches, including most U.S. Field Service regions, I have found that the key to a positive and mutually beneficial relationship between the customer and Digital Field Service is that person known as the Branch Manager.

All too often we (customers) tend to deal with Field Service only in crisis mode. How many times has DEC Field Service received this type of call:

> "My system crashed after a lightning storm; the system disk won't boot and I have 53 reports due on the C.I.S. director's desk in two hours . . . What do you mean you 'can't send someone out until about 3 P.M.'? . . . What am I paying for anyway?"

The most salient part of the fictional scenario mentioned above is the question "What am I paying for, anyway?". The time to find this out is well in advance of the first major disaster. There are, however, several answers to this question and the person who controls the implementation of those answers is the Branch Manager.

DEC offers essentially two flavors of Field Service Agreements: BASIC Service and DECservice.

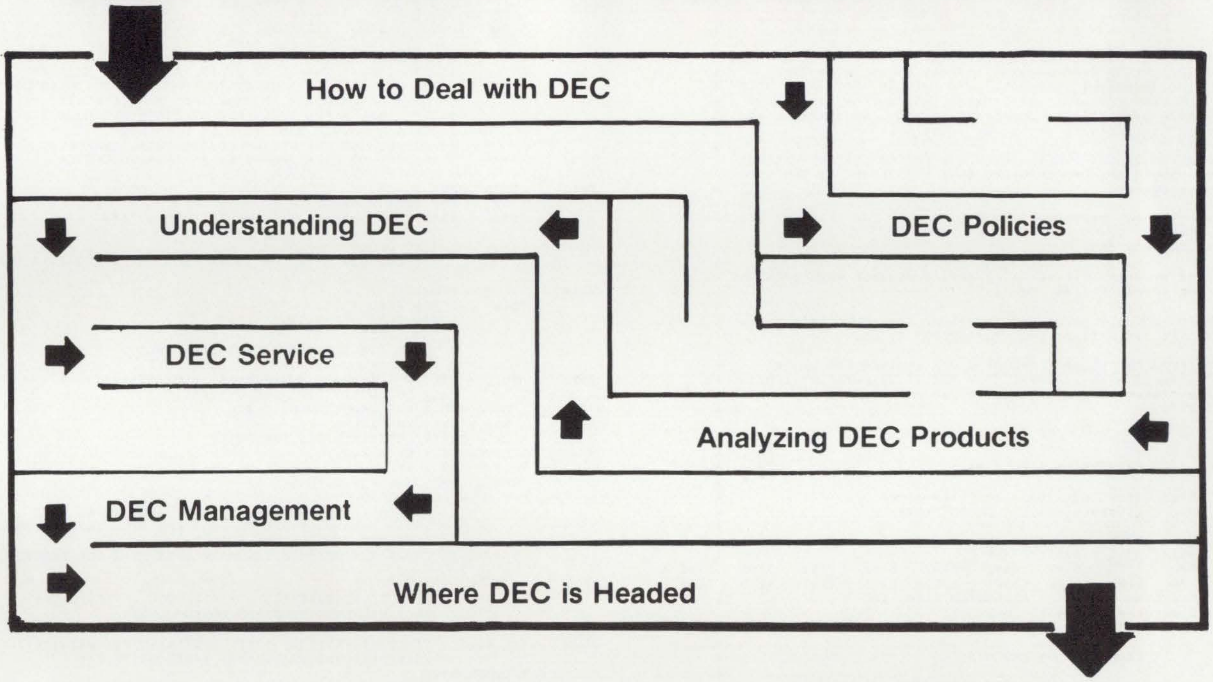
Basic Service provides contractual coverage (parts and labor) for equipment on a BEST AVAILABLE EFFORT basis. Under Basic Service, DEC agrees to furnish a service technician and parts as soon as a technician is available. If you are a Basic Service customer, your ability to affect your local DEC Field Service organization can be pretty much reduced to a "who you know" (friend of a friend of a neighbor of the District Manager) situation. Although some of the suggestions following apply to you, your clout within DEC Field Service will be reduced.

Under DEC service (so the sales brochure says) DEC is committed to providing continuous effort and technical expertise escalation until the problem is resolved. If you are a DEC service customer (about a 25% contract price premium) you have entered into an agreement with your local Field Service organization indicating that you are willing to "put your money where your mouth is" for the best field service that DEC is willing to supply.

I would recommend that every DEC service customer take the following initiative to ensure that the communication pathways are open to the DEC Field Service Branch Manager:

1. Meet with your Branch Manager and Unit Manager during a non-crisis time (hopefully before your first crisis and preferrably over lunch) to discuss your expectation level regarding your Field Service contract as it relates to your Branch's ability to provide service. Discuss such matters as initial response time (with a real person as well as the DDC), escalation timeframes (the ones in the sales brochure are frequently mythical), sparing levels (if you have eighteen RP06's you might reasonably expect the branch to be spared at least two boards deep, if you have the only DEC tape system in the branch the spares should still be in the branch), technician competence level and availability (does the branch manager dispatch his receptionist to DEC service sites so that his response time stays good even though all his senior technicians are tied up), and any special needs that you feel you have (e.g. "I can only give you the system after 3:00 PM for tape drive repairs").
2. Request that your Branch Manager set up a meeting with yourself and his/her District Manager to review the above issues at the District Level.
3. Get a table of organization for your Field Service Branch. The table should begin with your Site Rep and end with Ken Olson. Although you probably won't ever have to go above the Regional Level the purpose of getting the table is to ensure that DEC knows that YOU know how to get to the top if necessary.
4. Establish and maintain your credibility with your Branch Manager by verifying that your problems are hardware related before placing a service call. If the call is on a terminal, and you can wait until Monday to get the terminal repaired, don't insist on a four hour response time at 4:45 P.M. on a Friday afternoon.
5. Having done all of the above, when a crisis does occur insist on staying in the information flow. NEVER hassle a technician who is repairing your machine (you want him/her to WANT to fix it quickly), but request and insist upon periodic updates from the Problem Manager (usually the Unit Manager). There is no worse feeling than finding out that a problem that you thought was resolved last night is still keeping your system down this morning.
6. After a major problem or repair, meet with the Unit Manager and Branch Manager to iron out rough spots in the repair procedure.
7. Be assertive. If the problem is not being resolved in a reasonable and straightforward fashion, you are entitled to know from the Branch Manager how he/she is going to rectify the situation NOW.

Effective communications between DEC customer and DEC Field Service will certainly be enhanced if you proceed from the premise that your Branch Manager's job is to provide an acceptable level of service to you, while efficiently managing the resources at his/her disposal. My observation over the last several years has been that DEC Field Service management is usually willing to meet the customer more than halfway, provided the customer has a reasonable and realistic expectation of the service which he/she has purchased. ♥

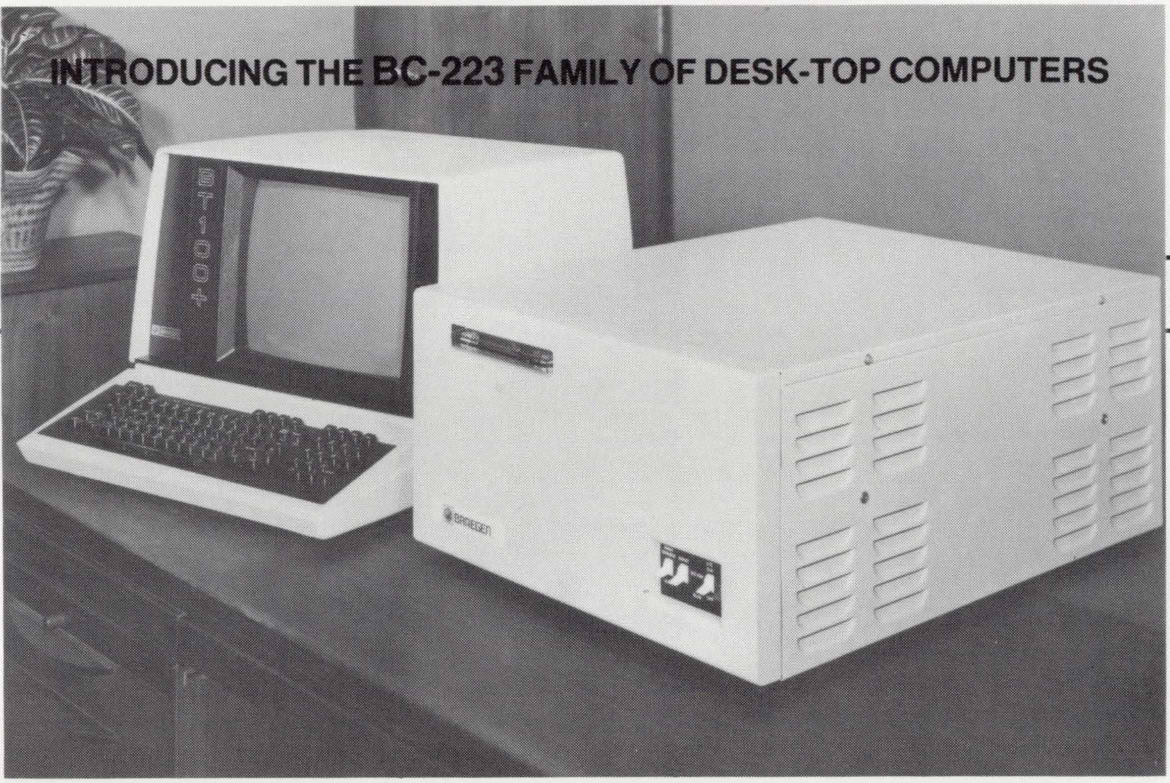

How to Deal with DEC
Understanding DEC
DEC Policies
DEC Service
Analyzing DEC Products
DEC Management
Where DEC is Headed

FILMAP.BAS

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

```
1       EXTEND
2       !                                                                  &
        !                                                                  &
        !                       F I L M A P                                &
        !                                                                  &
        !                                                                  &

3       !       AUTHOR:         JIM SWANSON                                &
        !       DATE:           8-JUL-81                                   &
        !       VERSION:        1.0                                        &

8       !                                                                  &
        !                                                                  &
        !                                                                  &
        !                 C O P Y R I G H T `                              &
        !                                                                  &
        !                                                                  &
   !            Copyright (C) 1981 by                                      &
   !    Area Two Educational Computer Center, Mason City, IA               &
   !                                                                       &
   !                                                                       &
   !    This software is furnished free of charge to members of tne        &
   !    North American DECUS organization and may be copied only           &
   !    with the  inclusion  of  the  above  copyright  notice.   This     &
   !    software  or  any  other copies thereof may not be provided or     &
   !    otherwise made available to any other person.  No title to and     &
   !    ownership of the software is hereby transferred.                   &
   !                                                                       &
   !    The information in this software is subject to change  without     &
   !    notice and should not be construed as a commitment by The Area     &
   !    Two Computer Center.                                               &
   !                                                                       &
   !    ATECC assumes  no  responsibility for the use  or  reliability     &
   !    or its software.
9       !  PROGRAM DESCRIPTION:                                            &
        !                                                                  &
        !       This program will read tne device cluster maps from the    &
        !  UFD's retrieval entries (UAR's).                                &
        !                                                                  &
        !       Wildcards are permitted in all of the input file specs     &
        !  such as: [1,*]*.DAT, or DR2:[*,*]STUF??.*, etc.                 &
        !                                                                  &
        !       To read the cluster map of a UFD include the switch        &
        !  "/UFD" after the device:PPN.  EX: "SY:[1,23]/UFD" will print    &
        !  tne file cluster map of the ufd of account 1,23 on the          &
        !  disk "SY:".                                                     &

10 ! THE OUTPUT:                                                           &
!                                                                          &
!                       EXAMPLE OF A FILE MAP:                             &
!                                                                          &
!DR1:[1,50]SYSTAT.BAS  Cl: 4  Si: 106   <- (file size)                     &
!CLSTR/   DCN(S)  where clusters start ( DCN'S/CLSTR: 1 )                  &
!                                                                          &
!    1/   3764   3765   3766 *  <- (The star indicates non-contiguous)     &
!    4/   4950   4951   4952   4953 *                                      &
!    8/   5187   5188   5189   5190   5191   5192   5193   5194   5195   5196 &
!   18/   5197   5198   5199   5200   5201   5202   5203   5204   5205   5206 &
!               Total clusters: 27   Allocated size: 108                   &
!     ^ (the first cluster #'s in each row)                                &
!                                                                          &
!                                                                          &
!                       EXAMPLE OF A UFD MAP:                              &
!DB0:[15,0]  Cl: 16                                                        &
!CLSTR/   DCN(S)  where clusters start ( DCN'S/CLSTR: 4 )                  &
!                                                                          &
!    1/  16693  16697  16701  16705  16709  16713  16717                   &
!               Total clusters: 7   Allocated size: 112                    &
!        ( NOTE: no stars printed means that this UFD is contiguous )      &
!                                                                          &
!                                                                          &
!
900     DIM #1%, U%(3583%,7%)                                              &
        \ DIM C%(30%)                                                      &
                !> U%(3583,7) = UFD ARRAY                                  &
                !> C%(30) = FOR FILE SPECS TO GO TO WILDCARD LOOKUPS       &

1000    ON ERROR GOTO 19000                                                &
        \ PRINT "FILMAP - Prints list of retrieval DCN's for any file."    &
        \ PRINT                                                            &
                !> 19000, STANDARD ERROR TRAP                              &
                !  A LITTLE SIGNON BLURB                                   &

1010    PRINT 'Output to <KB:>';                                           &
        \ INPUT LINE O$                                                    &
        \ O$ = CVT$$(O$,-1%)                                               &
        \ O% = 12%                                                         &
        \ O% = 0%          UNLESS  LEN(O$)                                 &
        \ IF    O%                                                         &
        THEN    OPEN O$ FOR OUTPUT AS FILE #O%                             &
                !> O$ = OUTPUT FILE NAME                                   &
                !> O% = OUTPUT FILE CHANNEL (0 IF O$ = "KB:")              &

1020    INPUT 'W-ide or N-arrow <N>';A$                                    &
        \ WDTH% = 80%                                                      &
        \ WDTH% = 132%  IF       ASCII(A$) = 87%                           &
        \ WDTH% = 80%   IF       ASCII(A$) = 78%                           &
                !> WDTH% = WIDTH OF PRINTOUT                               &

1030    WILDRAD% = -17947%                                                 &
        \ TOTAL.FILES% = 0%                                                &
                !> WILDRAD% = RAD-50 REPRESNTATION OF "???"                &
                !> TOTAL.FILES% = TOTAL NUMBER OF FILES MAPPED             &

1990    PRINT           .                                                  &
        \ PRINT #O%, 'Cluster maps printed on ';DATE$(0%);' at ';TIME$(0%) &
        \ PRINT #O%                                                        &
                !  A HEADING LINE FOR THE PRINT OUT
2000    !                                                                  &
        !                                                                  &
        !       M A I N   F I L E N A M E   L O O K U P   L O O P          &
        !                                                                  &
        !                                                                  &

2003    PRINT 'File';                                                      &
        \ TOTAL.FILES% = TOTAL.FILES% + FIL.CNT%                           &
        \ FIL.CNT% = 0%                                                    &
        \ INPUT LINE F$                                                    &
        \ F$ = CVT$$(F$,4%)                                                &
        \ UFD.% = INSTR(1%,F$,'/UFD') > 0%                                 &
        \ GOTO 32767    UNLESS  LEN(F$)                                    &
        \ S$ = SYS(CHR$(6%)+CHR$(-23%)+F$)                                 &
        \ CHANGE S$ TO C%                                                  &
        \ C%(7%),C%(9%) = WILDRAD%       IF       (C%(29%) AND 1%) = 0%    &
        \ C%(8%),C%(10%) = SWAP%(WILDRAD%) IF     (C%(29%) AND 1%) = 0%    &
        \ C%(11%) = WILDRAD%             IF       (C%(29%) AND 8%) = 0%    &
        \ C%(12%) = SWAP%(WILDRAD%)      IF       (C%(29%) AND 8%) = 0%    &
        \ CHANGE C% TO S$                                                  &
        \ PPN.INX% = 0%                                                    &
        \ DEV$ = 'SY:'                                                     &
        \ DEV$ = CHR$(C%(23%))+CHR$(C%(24))+NUM1$(C%(25%))+':'             &
                        IF      C%(23%)                                    &
        \ GOSUB 10500                                                      &
        \ GOTO 2000     IF      ERROR%                                     &
                !  GET FILE SPECS FOR MAPS                                 &
                !> F$ = FILESPECS FOR MAPPING                              &
                !> FIL.CNT% = NUMBER OF FILES MAPPED FOR THIS SPEC         &
                !> S$ = SYSCALL RETURN STRING FROM FILNAME SCAN OF "F$"    &
                !  ALSO CHECK FOR FILE NAME (ADD *.* IF NONE)              &
                !> PPN.INX% = PPN INDEX FOR WILDCARD PPN LOOKUP            &
                !> DEV$ = DEVICE NAME (SY: IS DEFAULT)                     &
                !> 10500, GET THE DEVICE CLUSTER SIZE (PAKCLU)             &
                !  RETURN TO FILE PROMPT IF AN INVALID DEVICE              &

2010    GOSUB 10000                                                        &
        \ IF    PPN.INX% < 0%                                              &
        THEN    PRINT 'No matches for for file-spec:';F$                   &
                                UNLESS FIL.CNT%                            &
                \ GOTO 2000                                                &
                !> 10000, GET A PPN ON INDEX FROM THE INPUT FILE NAME      &

2012    IF      UFD.%                                                      &
        THEN    GOSUB 4000                                                 &
                \ PPN.INX% = PPN.INX% + 1%                                 &
                \ GOTO 2010                                                &
                !  THIS FOR LOOKINP UP UFD MAPS ONLY                       &

2015    UFD.$ = DEV$ + '['+NUM1$(SWAP%(PPN%) AND 255%)+','                 &
                        +NUM1$(PPN% AND 255%)+']'                          &
        \ OPEN UFD.$ FOR INPUT AS FILE #1%                                 &
        \ S1$ = SYS(CHR$(12%))                                             &
        \ S% = SWAP%(CVT$%(MID(S1$,13%,2%)))                               &
        \ UFD.CLU% = U%(31%,0%)                                            &
        \ IF    S% = 0%                                                    &
        THEN    PPN.INX% = PPN.INX% + 1%                                   &
                \ GOTO 2000                                                &
                !  RETURN IF THE PPN INDEX COUNTER SHOWS NO MORE           &
                !> UFD.$ = NAME OF UFD TO OPEN FOR FILE INFORMATION        &
                !  OPEN THE UFD FOR THE LOOKUPS                            &
                !  CHECK THE UFD'S SIZE, MAKE SURE IT'S NOT ZEROED         &

2020    I% = 0%                                                            &
        \ CHANGE SYS(CHR$(6%)+CHR$(-10%) + F1$) TO      FILCALL%           &
                !> I% = WILDCARD FILENAME LOOKUP INDEX                     &

2025    FILCALL$ = SYS(CHR$(6%)+CHR$(17%)+CHR$(I%)+CHR$(SWAP%(I%))         &
                +CHR$(PPN%)+CHR$(SWAP%(PPN%))+MID(S$,7%,24%))              &
        \ GOSUB 3000                                                       &
                !> FILCALL$ = RETURNED STRING FROM WILDCARD FILNAME CALL   &

2030    I% = I% + 1%                                                       &
        \ GOTO 2025
3000    !                                                                  &
        !                                                                  &
        !       PRINT A MAP FOR A FILE                                     &
        !                                                                  &
        !                                                                  &

3010    PTR% = 0%                                                          &
        \ F1% = SWAP%(CVT$%(MID(FILCALL$,7%,2%)))                          &
        \ F2% = SWAP%(CVT$%(MID(FILCALL$,9%,2%)))                          &
        \ F3% = SWAP%(CVT$%(MID(FILCALL$,11%,2%)))                         &
                !  GET THE FILNAME IN RAD50                                &
                !> PTR% = CURRENT POINTER INTO THE UFD NAME ENTRY          &
                !> F1% = FIRST 3 CHARACTERS OF FILENAME IN RAD50           &
                !> F1% = LAST 3 CHARACTERS OF FILENAME IN RAD50            &
                !> F1% = EXTENSION OF FILENAME IN RAD50                    &

3040    PTR% = FNL%(U%(PTR%,0%))                                           &
        \ GOTO  3050    UNLESS  PTR%                                       &
        \ GOTO  3040    IF      U%(PTR%,4%) AND 64%                        &
        \ GOTO  3040    UNLESS  U%(PTR%,1%) = F1%                          &
        \ GOTO  3040    UNLESS  U%(PTR%,2%) = F2%                          &
        \ GOTO  3040    UNLESS  U%(PTR%,3%) = F3%                          &
        \ GOTO  3060                                                       &
                !  FIND THE FILE                                           &

3050    PRINT 'File not found.';RAD$(F1%);RAD$(F2%);'.';RAD$(F3%)          &
        \ GOTO 3090                                                        &

3060    UAR% = FNL%(U%(PTR%,7%))                                           &
        \ UAA% = FNL%(U%(PTR%,6%))                                         &
        \ F.CLU% = U%(UAA%,7%)                                             &
        \ F.SIZ% = U%(UAA%,2%)                                             &
        \ CLUS.CNT% = F.CLU%/PAKCLU%                                       &
        \ PRINT #O%                                                        &
        \ PRINT #O%, UFD.$;RAD$(F1%);RAD$(F2%);'.'                         &
                ;RAD$(F3%);                                                &
        \ PRINT #O%, '  Cl:';F.CLU%;                                       &
        \ PRINT #O%, ' Si:';F.SIZ%;                                        &
        \ PRINT #O%                                                        &
        \ PRINT #O%,'CLSTR/   DCN(S)  where clusters start';               &
                " ( DCN'S/CLSTR:";CLUS.CNT%;')'                            &
                        IF      UAR%                                       &
        \ PRINT #O%, ' THERE ARE NO CLUSTERS ASSIGNED TO THIS FILE.'       &
                                IF      UAR% = 0%                          &
```

BRAEGEN MPD HAS THE DESKTOP COMPUTER YOU NEED



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

FEATURES

- Based on *LSI 11/23 Processor
- Up to 256 KBYTES of MOS memory
- 24 MBYTE Winchester Disk
- 20 MBYTE cartridge tape back-up
- Available in desk-top or rack-mount configurations
- Four asynchronous communication ports
- Wide range of options available
- Quantity discounts available

For more information on the BC-223 family of desk-top systems or any of our full line of DEC-compatible disk and tape subsystems, terminals, printers, memories, multiplexors and more, please call or write:

Marketing Dept:

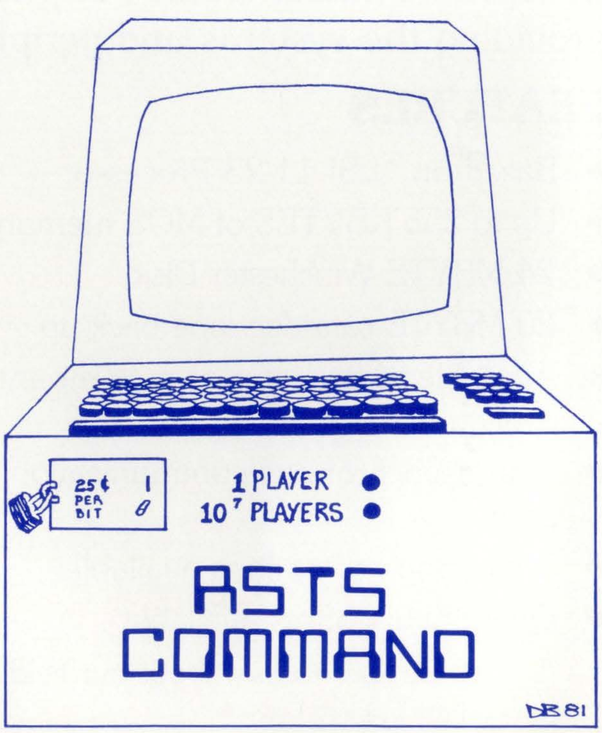
BRAEGEN
MPD
3320 East La Palma Avenue
Anaheim, California 92806
Telephone (714) 520-9200

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

*Trademark of Digital Equipment Corporation

CIRCLE 95 ON READER CARD

```
        \ GOTO 3090       UNLESS  UAR%                                 &
        \ CL.N%,C.WDTH%, U.CNT% = 0%                                   &
        \ DCN = U%(UAR%,1%)                                            &
        \ DCN = DCN + 65536.     IF       DCN < 0%                     &
        \ LAST.DCN = DCN-CLUS.CNT%                                     &
        \ WHILE UAR%                                                   &
        \ FOR Z% = 1% TO 7%                                            &
        \ IF     U%(UAR%,Z%) = 0%                                      &
        THEN     3070                                                  &
        ELSE     CL.N% = U.CNT% * 7%+Z%                                &
                 \ IF     C.WDTH% + 9% > WDTH%                         &
                 OR       C.WDTH% = 0%                                 &
                 THEN     PRINT #O%                                    &
                          \ C.WDTH% = 6%                               &
                          \ PRINT #O%, USING '#####/';CL.N%;           &
                 !> C.WDTH% = CURRENT CHARACTER POSITION FO PRINT OUT  &
                 !> CL.N% = CURRENT CLUSTER NUMBER ENTRY WITHIN MAP    &
                 !> CLUS.CNT% = NUMBER OF  DCN'S / FILE CLUSTER        &

3065    DCN = U%(UAR%,Z%)                                              &
        \ DCN = DCN + 65536.     IF       DCN < 0%                     &
        \ IF     LAST.DCN + CLUS.CNT% = DCN                            &
        THEN     PRINT #O%, USING '  #####',DCN;                       &
                 \ C.WDTH% = C.WDTH% + 7%                              &

3067    IF       LAST.DCN + CLUS.CNT% <> DCN                           &
        THEN     PRINT #O%,' *'                                        &
                 \ PRINT #O%, USING '#####/  #####';CL.N%,DCN;         &
                 \ C.WDTH% = 13%                                       &
                 ! PRINT THE FIRST DCN FOR TEH FILE CLUSTER, AND ANY OTHER DCNS &
                 !       IF THE FILE'S CLUSTERSIZE IS BIGGER THAN THE DEVICE &
                 !       CLUSTER SIZE                                  &

3070    LAST.DCN = DCN                                                 &
        \ NEXT Z%                                                      &
        \ UAR% = FNL%(U%(UAR%,0%))                                     &
        \ U.CNT% = U.CNT% + 1%                                         &
        \ NEXT                                                         &
        \ PRINT #O%,                                                   &
        \ PRINT #O%, '            Total clusters:';CL.N%;              &
        \ PRINT #O%, '  Allocated size:';CL.N%*F.CLU%                  &

3090    FIL.CNT% = FIL.CNT% + 1%                                       &
        \ RETURN
4000    !                                                              &
        !                                                              &
        !        P R I N T   A   M A P   F O R   A   U F D             &
        !                                                              &
        !                                                              &

4010    PPN$ = '['+NUM1$(SWAP%(PPN%) AND 255%)+                        &
                 ','+NUM1$(PPN% AND 255%)+']'                          &
        \ OPEN DEV$+PPN$ FOR INPUT AS FILE #1%                         &
        \ S1$ = SYS(CHR$(12%))                                         &
        \ S% = SWAP%(CVT$%(MID(S1$,13%,2%)))                           &
        \ F.CLU% = ASCII(MID(S1$,21%,1%))                              &

4060    UAR% = 31%                                                     &
        \ UAA% = FNL%(U%(PTR%,6%))                                     &
        \ F.CLU% = U%(UAR%,0%)              IF       S%                &
        \ CLUS.CNT% = F.CLU%/PAKCLU%                                   &
        \ PRINT #O%                                                    &
        \ PRINT #O%, DEV$;PPN$;'  Cl:';F.CLU%;                         &
        \ PRINT #O%                                                    &
        \ PRINT #O%,'CLSTR/   DCN(S)  where clusters start';           &
                 " ( DCN'S/CLSTR:";CLUS.CNT%;')'                       &
        \ PRINT #O%, '     ***ZEROED ACCOUNT***'          UNLESS  S%   &
        \ PRINT #O%                UNLESS  S%                          &
        \ GOTO 4090       UNLESS  S%                                   &
        \ C.WDTH%, U.CNT% = 0%                                         &
        \ DCN = U%(UAR%,1%)                                            &
        \ DCN = DCN + 65536.     IF       DCN < 0%                     &
        \ LAST.DCN = DCN-CLUS.CNT%                                     &
        \ FOR Z% = 1% TO 7%                                            &
        \ IF     U%(UAR%,Z%) = 0%                                      &
        THEN     4070                                                  &
        ELSE     CL.N% = U.CNT% * 7%+Z%                                &
                 \ IF     C.WDTH% + 9% > WDTH%                         &
                 OR       C.WDTH% = 0%                                 &
                 THEN     PRINT #O%                                    &
                          \ C.WDTH% = 6%                               &
                          \ PRINT #O%, USING '#####/';CL.N%;           &
                 !> C.WDTH% = CURRENT CHARACTER POSITION FO PRINT OUT  &
                 !> CL.N% = CURRENT CLUSTER NUMBER ENTRY WITHIN MAP    &
                 !> CLUS.CNT% = NUMBER OF  DCN'S / FILE CLUSTER        &

4065    DCN = U%(UAR%,Z%)                                              &
        \ DCN = DCN + 65536.     IF       DCN < 0%                     &
        \ IF     LAST.DCN + CLUS.CNT% = DCN                            &
        THEN     PRINT #O%, USING '  #####',DCN;                       &
                 \ C.WDTH% = C.WDTH% + 7%                              &

4067    IF       LAST.DCN + CLUS.CNT% <> DCN                           &
        THEN     PRINT #O%,' *'                                        &
                 \ PRINT #O%, USING '#####/  #####';CL.N%,DCN;         &
                 \ C.WDTH% = 13%                                       &
                 ! PRINT THE FIRST DCN FOR TEH FILE CLUSTER, AND ANY OTHER DCNS
                 !       IF THE FILE'S CLUSTERSIZE IS BIGGER THAN THE DEVICE &
                 !       CLUSTER SIZE                                  &

4070    LAST.DCN = DCN                                                 &
        \ NEXT Z%                                                      &
        \ UAR% = FNL%(U%(UAR%,0%))                                     &
        \ U.CNT% = U.CNT% + 1%                                         &
        \ PRINT #O%,                                                   &
        \ PRINT #O%, '            Total clusters:';CL.N%;              &
        \ PRINT #O%, '  Allocated size:';CL.N%*F.CLU%                  &

4090    FIL.CNT% = FIL.CNT% + 1%                                       &

4095    RETURN
10000   !                                                              &
        !        WILD CARD PPN LOOKUP                                  &
        !                                                              &
        !        RETURN A -1 IN "PPN.INX%" IF NO MORE FOUND            &
        !                                                              &
10010   IF       ((C%(30%) AND 1%) OR (C%(30%) AND 2%)) = 0%           &
        THEN     PPN% = C%(5%)+SWAP%(C%(6%))                           &
                 \ PPN% = PEEK(PEEK(PEEK(520%)+8%)+24%)  UNLESS  PPN%  &
                 \ PPN.INX% = -1%          IF       PPN.INX% <> 0%     &
                 \ RETURN                                              &
                 !  HERE ONLY IF NO WILDCARD WAS SPECIFIED             &
                 !  USE CURRENT PPN IF NONE SPECIFIED                  &
                 !  WE WILL ONLY USE THE ZERO-TH INDEX                 &

10020   Z$ = SYS(CHR$(6%)+CHR$(25%)+CHR$(PPN.INX%)+CHR$(SWAP%(PPN.INX%))+ &
                 MID(S$,5%,23%))                                       &
        \ PPN% = SWAP%(CVT$%(MID(Z$,5%,2%)))                           &
        \ PPN.INX% = -1%          UNLESS  PPN%                         &

10030   RETURN
10500   !                                                              &
        !                                                              &
        !        FIND A DEVICE CLUSTER SIZE                            &
        !                                                              &
        !        RETURN DEVICE CLUSTER SIZE IN "PAKCLU%"               &
        !        RETURN ERROR FLAG IN "ERROR%" IF DEVICE NOT FOUND     &
        !                                                              &
        ERROR% = 0%                                                    &

10510   OPEN DEV$+'[1,1]' FOR INPUT AS FILE #1%                        &
        \ PAKCLU% = U%(0%,4%)                                          &

10520   RETURN                                                         &

19000   !                                                              &
        !        STANDARD ERROR TRAP                                   &
        !                                                              &

19010   IF       ERR = 5%                                              &
        THEN     IF       ERL = 10020%                                 &
                 THEN     PRINT 'No matches found for -';F$            &
                                             IF       PPN.INX% = 0%    &
                          \ PPN.INX% = -1%                             &
                          \ RESUME 10030                               &
                 !  OUT OF PPNS IN WILDCARD PPN LOOKUP                 &

19020   IF       ERR = 5%                                              &
        THEN     IF       ERL = 2025%                                  &
                 OR       ERL = 2015%                                  &
                 THEN     PPN.INX% = PPN.INX% + 1%                     &
                          \ RESUME 2010                                &
                 !  OUT OF FILES FOR THIS ACCOUNT                      &
                 !  GO LOOK FOR ANOTHER ACCOUNT                        &

19030   IF       ERL = 10510%                                          &
        THEN     ERROR% = -1%                                          &
                 \ RESUME 10520                                        &
                 !  CAN'T FIND THE MFD ON THE PACK                     &

19040   IF       ERR = 5%                                              &
        AND      ERL = 4010%                                           &
        THEN     RESUME 4095%                                          &
                 !  UFD NOT FOUND ON OPEN                              &

19100   E$ = RIGHT(SYS(CHR$(6%)+CHR$(9%)+CHR$(ERR)),3%)                &
                 !> E$ = ERROR MESSAGE                                 &

19110   IF       ERR = 1%                                              &
        THEN     PRINT #O%,                                            &
                 \ PRINT #O%,    E$;' for ';DEV$+PPN$                  &
                 \ PRINT E$;' for ';DEV$+PPN$    IF      O%            &
                 \ RESUME 4090%  IF       ERL = 4010%                  &
                 \ IF     ERL = 2015%                                  &
                 THEN     PPN.INX% = PPN.INX% + 1%                     &
                          \ RESUME 2010%                               &
                 !  BAD DIRECTORY FOR DEVICE                           &

19120   IF       ERL = 2003%                                           &
        THEN     PRINT E$;' - ';F$                                     &
                 \ RESUME                                              &

19980   IF       ERR > 49%                                             &
        AND      ERR < 53%                                             &
        THEN     PRINT 'Bad nubmer.'                                   &
                 \ RESUME                                              &

19990   PRINT ERR;ERL;E$                                               &
        \ STOP                                                         &

20000   DEF FNL%(Z%) = (((Z% AND 3584%)/512%)*UFD.CLU%+                &
        (SWAP%(Z% AND -4096%)/16%))                                    &
        *32%+((Z% AND 496%)/16%)                                       &
                 !> FNL%(LINK WORD) = FIND LINK TO UFD ENTRY           &

32767   END
```



DILOG NUMBER 1 FOR DEC 11*

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

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

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

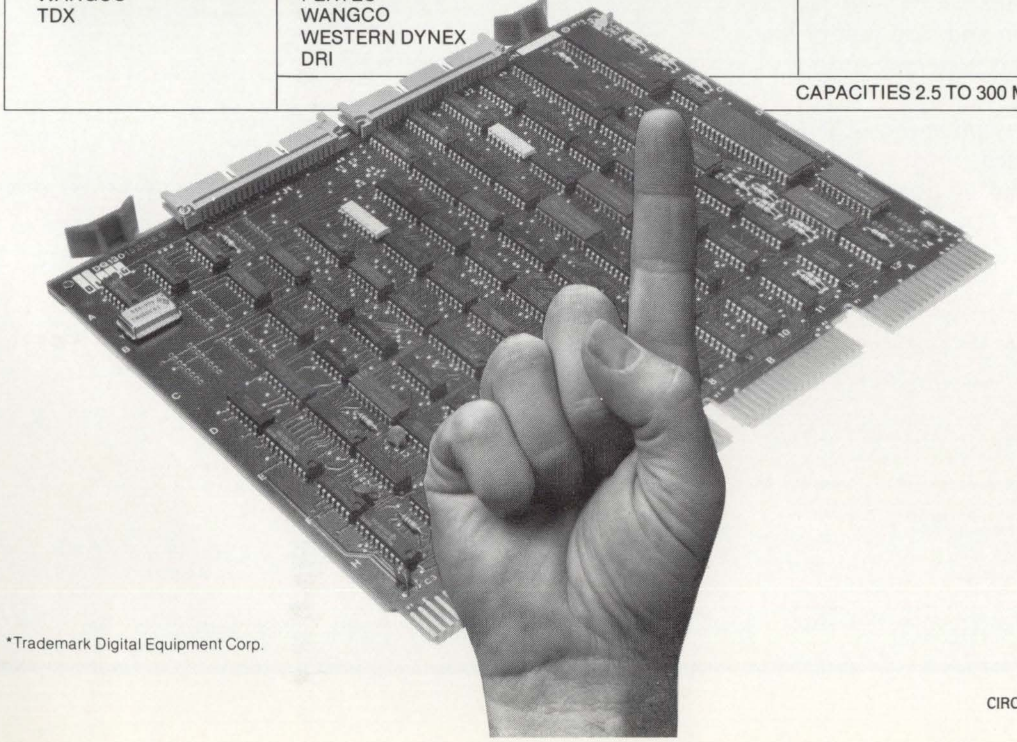
These high performance data storage interface products also feature • minimum bus/space requirements • up to 60% less power • 10 to 50% lower cost • automatic self-test…and numerous other features for easy system integration.

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

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

DISC/TAPE DRIVE MANUFACTURER COMPATIBILITY CHART

MAGNETIC TAPE	DISC			
½" REEL-TO-REEL STD. & STREAMER	**2315/5440/RK05 CARTRIDGE CLASS**	**CMD CARTRIDGE MODULE**	**SMD STORAGE MODULE**	**WINCHESTER 5¼", 8" OR 14"**
AMPEX CIPHER CONTROL DATA DIGI-DATA KENNEDY MICRODATA PERTEC TANDBERG DATA (IDT) WANGCO TDX	AMPEX CAELUS CENTURY DATA CONTROL DATA DEC DIABLO IOMEC MICRODATA PERTEC WANGCO WESTERN DYNEX DRI	AMPEX CONTROL DATA	AMPEX CENTURY DATA CONTROL DATA BALL COMPUTER MITSUBISHI	BASF CONTROL DATA FUJITSU KENNEDY MEMOREX PRIAM SHUGART SEAGATE QUANTUM IMI
	CAPACITIES 2.5 TO 300 MB			



DISTRIBUTED LOGIC CORP.
DILOG

*Trademark Digital Equipment Corp.

CIRCLE 15 ON READER CARD

QUERY.TEC

A Search and Substitute TECO MACRO with QUERY Facility

By Mark J. Diaz, Dataguard Corp., Hinsdale, IL 60521

TECO is generally powerful enough to accomplish any given search and substitute modification throughout your entire text file with a single command string. For example, removing an unknown number of spaces embedded within a file specification in a file produced by DIRECT is easily done.

Occasionally the command string to accomplish the desired substitutions would either take longer to write than "manually" editing each occurance, or would be nearly impossible to write because the occurances are not defined by their context within the text.

For example:

> You have entered an entire file manipulation module. However in about half the instances where you should have typed a variable name corresponding to one file, you have typed a variable name corresponding to another file. Assume your standard for variable names associated with file buffers is a unique 3 or 4 letter prefix, a dot, and suffix unique only within it's prefix. So, you have entered FILE1.FIELD$ when you meant to enter FILE2.FIELD$ and vice versa. There is no reasonable pattern as to which variables should be which.

This very example, and my desire to become more proficient in TECO, caused me to write the QUERY.TEC macro.

FEATURES

- Underlines text to be substituted
- Online instruction always available
- Informs user how to get help on invalid entries
- Options available:
  1) Do the substitution.
  2) Don't do the substitution.
  3) Don't do the substitution and skip rest of line.
  4) Substitute the rest of the matching strings.
  5) Exit search.
- Type only the letter for any option (no RETURN).
- Additional option easily added.

The actual TECO macro, loaded as usual with TECO's EIQUERY$$ command follows.

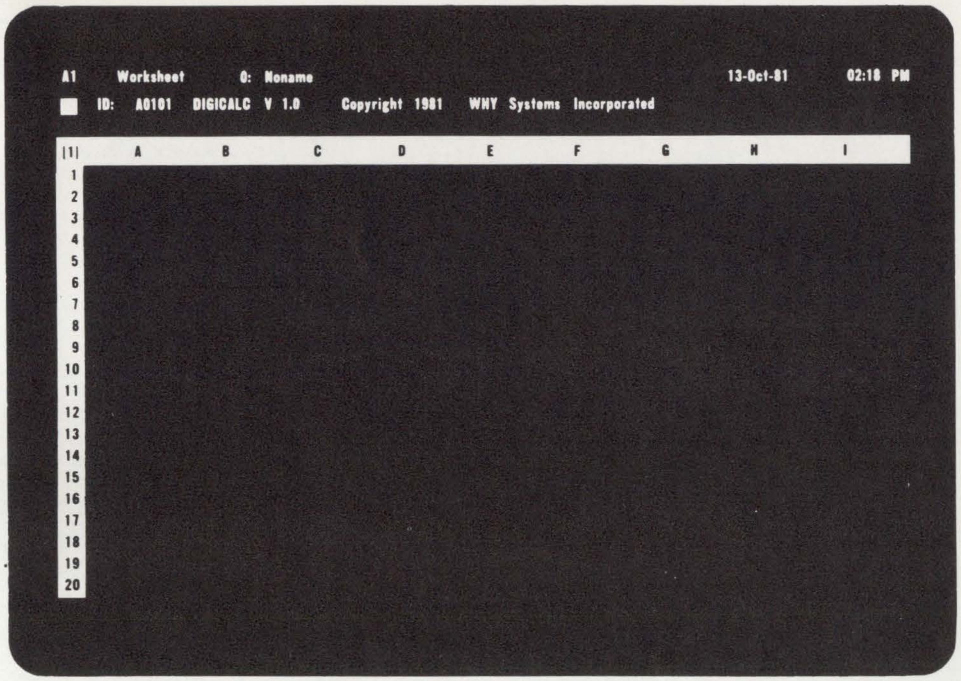
```
!*
        TECO macro:     QUERY.TEC

        This macro performs search and substitute operations with
        an operator query facility.
        It it loaded (into Q-register Q) with the EIQUERY$ command.
*!
$^A
Loading "QUERY.TEC" into Q-reg "Q".
Type "1UQMQ$$" for instructions.
^A
0UQ                     !* Zero the I-want-instructions flag *!
                        !* Load the macro into Q-reg Q *!
@^UQ/
QQ"N                    !* If instructions are desired, print them and exit *!

^ATo use the QUERY macro after loading it into Q-register Q with
the EIQUERY$ command, simply append an MQ command after your
substitute command.

                Example:        FSpront$print$VMQ$

Note:   It would behoove you not to exit this macro with ^C because
        Q-registers I, R, 1, and 2 are used and restored on a normal
        exit (either an E command or a search failure).
```

```
After using the MQ command, QUERY will prompt for one of
several options. Enter a question mark (?) for the help message.

^A
0UQ                     !* Zero the I-want-instructions flag *!
^C'                     !* Exit macro *!

[I                      !* Save all registers used *!
[1
[2
[R

^YXI                    !* Save last string inserted *!
$0UR                    !* Clear the Do-rest-of-matching-strings flag *!
!S!
:S$"E                   !* If the search string can not be found, exit *!

    ^A?SRH Search failure "^A:G_^A"^A !* Display string not found *!
    BJ                  !* Return to beginning of buffer *!
    OX$                 !* Exit QUERY macro *!
   '

V                       !* Display line containing string found *!
QR"N                    !* Is the Do-rest-of-matching-strings flag set ? *!
   FR$ GI$              !* Yes?, then do the substitution *!
   OS$                  !* Go look some more *!
  '

                        !* Underline the string to substitute *!

.U2                     !* Save the current position *!
0L                      !* Get to the beginning of the line *!
<
 (Q2+^S-.)"E            !* Are we at the string to substitute ? *!
          0;            !* Yes?, then exit iteration loop *!
         '
 (0A-9)"E               !* Is the character a tab ? *!
        9^T             !* Yes?, then display a tab *!
       |
        32^T            !* No?, then display a space *!
       '
 C                      !* Advance a character *!
>
Q2J                     !* Restore the buffer pointer *!
-^S<^A^^A>              !* Display pointers to the string to substitute *!
^A
^A
!A!
^A>^A                   !* Prompt the user for what option to do *!
^Tu1                    !* Get response into Q-reg 1 *!
^A
^A
(Q1-89)"E               !* Was a "Y" entered ? *!
        FR$ GI$         !* Yes?, the do the substitution *!
        OS$             !* Go look some more *!
       '
(Q1-78)"E               !* Was an "N" entered ? *!
        OS$             !* Yes?, then go look some more *!
       '
(Q1-76)"E               !* was an "L" entered ? *!
        L               !* Yes?, then skip the rest of this line *!
        OS$             !* Go look some more *!
       '
(Q1-82)"E               !* Was an "R" entered ? *!
        FR$ GI$         !* Yes?, then do the substitution *!
        1UR             !* Set the Do-rest-of-matching-string flag *!
        OS$             !* Go look some more *!
       '
(Q1-69)"E               !* Was an "E" entered ? *!
        OX$             !* Yes?, then exit iteration loop *!
       '
(Q1-63)"E               !* If a question mark is entered, *!
                        !* display the help message *!
^AValid options are:
 Y      - Yes, do the substitution.
 N      - No, do not do the substitution.
 L      - No, do not do the substitution and skip the rest of the line.
 R      - Substitute the rest or the remaining matches *!
 E      - Exit QUERY
 ?      - This help message

^A
        V               !* Redisplay the current line *!
        OA$             !* Go ask again *!
       '

                        !* Entry was not a valid option *!

(Q1-13)"E               !* Was a carraige return entered ? *!
        ^T              !* Yes?, then get the line-feed *!
       '
^APlease enter a valid option, type "?" for HELP.

^A
V                       !* Redisplay the current line *!
OA$                     !* Go ask again *!

!X!                     !* Exit QUERY macro *!
^A
^A

]R                      !* Restore all Q-regiteres used *!
]2
]1
]I
$/EI$$
```

WHY SYSTEMS

announces . . .

DIGICALC™



```
A1    Worksheet    0: Noname                                  13-Oct-81    02:18 PM
■ ID: A0101  DIGICALC  V 1.0   Copyright 1981  WHY Systems Incorporated

|1|    A      B      C      D      E      F      G      H      I
 1
 2
 3
 4
 5
 6
 7
 8
 9
10
11
12
13
14
15
16
17
18
19
20
```

THE ELECTRONIC SPREADSHEET FOR DEC COMPUTER SYSTEMS

APPLICATIONS

Financial statements
Business forecasting
Resource management
Investment analysis
Job costing
Performance analysis
Cash Flow analysis
Error analysis

EASY TO USE

Built-in automatic training procedure
Instant HELP available at the terminal
User-friendly / interactive

SAVES MONEY

Reduces demand on information system personnel
Eliminates long hours of "what if" calculations
Report and form printing costs reduced or eliminated
Low, one-time investment

PROVIDES INSTANT REPORTS

Choice of formats
Working copies or board-room quality

- AUTOMATIC CALCULATION
- EXTENSIVE MATH FUNCTIONS
  - ALGEBRAIC
  - LOGICAL
  - FINANCIAL
  - SCIENTIFIC
  - USER DEFINED FUNCTIONS
- TEN KEY NUMERIC DATA ENTRY
- EXTENSIVE **HELP** AT TERMINAL
- INCLUDES SELF TEACHING MODE
- WORKSHEET CONSOLIDATION
- VARIETY OF "BOARDROOM" QUALITY REPORTS
- SAVES AND RECALLS WORKSHEETS
- EXTERNAL FILE INTERFACE

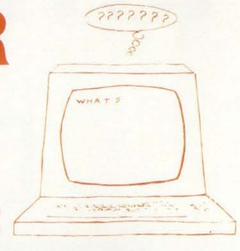
ON RSTS/E, RSX-11M, VMS, TOPS 10/20
VT-100 OR MOST EMULATORS

**WHY
SYSTEMS
INCORPORATED**

17130 Avondale Way, N.E.
Redmond, Washington 98052
(206) 881-2331

CALL TODAY FOR A DIAL-UP DEMONSTRATION

DEAR RSTS MAN:



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

DEAR RSTS MAN:

I have a MICOM error controller that won't work. In fact, It won't transmit or receive at all. The book says something about Data Terminal Ready and Request to Send; What do these mean and what do I do about them? No Go.

Dear NO-GO: *Some terminals (and error controllers) need certain EIA signals (other pins on the EIA connector) to be held "on" or "high" in order to function. These pins can be DTR, CTS, or RTS or a combination. A necessary tool in these matters is a "break-out box", sometimes called a "blue box". These devices show you by means of lights which pins are "high" and which are not. Most terminals that require these signals, also assert one of the leads such as DTR or RTS. What you need to do is jumper this "high" pin over to the one or more your error controller is expecting to see. In reality, you use the error controller to tell the error controller to "go ahead". Most blue boxes have jumpers brought out so that it is easy to experiment to find the combination that will work. I have used jumpers between pins 4, 8 and 20 to make an AJ terminal work. Good luck.*

DEAR RSTS MAN:

I would like to take this opportunity to express my gratitude and appreciation for the help you gave me concerning DTR and my problem. It worked perfectly and our MICOM error controller is finally on-line.

Best wishes for the New Year!

David A. Rooks
Manager of Information Services
Office Specialists, Boston, MA 02108

DEAR RSTS MAN:

HELP!!!! Do you have any information on using "Terminal Concentrators" (those neato thingies that allow a bunch of terminals to come into one port) on RSTS? Will RSTS allow this? It seems it would make job administration a slight impossibility. Anyone out there doing it who could lend a hand???? Thanks!!!!

Ron Barale, Operations Manager
Coast Mailing Corp., Sunnyvale, CA 94086

```
TECO SAMPLE.TXT
*HT$$
FILE1.FIELD1     FILE1.FIELD2     FILE1.FIELD3
FILE1.FIELD4     FILE1.FIELD5     FILE1.FIELD6
FILE1.FIELD7     FILE1.FIELD8     FILE1.FIELD9
*EIQUERY$$

Loading "QUERY.TEC" into Q-reg "Q".
Type "1UQMQ$$" for instructions.
*1UQMQ$$
To use the QUERY macro after loading it into Q-register Q with
the EIQUERY$ command, simply append an MQ command after your
substitute command.

                Example:        FSpront$print$VMQ$

Note:   It would behoove you not to exit this macro with ^C because
        Q-registers I, R, 1, and 2 are used and restored on a normal
        exit (either an E command or a search failure).

After using the MQ command, QUERY will prompt for one of
several options. Enter a question mark (?) for the help message.


*T$$
FILE1.FIELD1     FILE1.FIELD2     FILE1.FIELD3
*FSFILE1$FILE2$VMQ$$
FILE2.FIELD1     FILE1.FIELD2     FILE1.FIELD3
FILE2.FIELD1     FILE1.FIELD2     FILE1.FIELD3
                 ^ ^^^
```

```
>?
Valid options are:
 Y      - Yes, do the substitution.
 N      - No, do not do the substitution.
 L      - No, do not do the substitution and skip the rest of the line.
 R      - Substitute the rest of the remaining matches *!
 E      - Exit QUERY
 ?      - This help message

FILE2.FIELD1     FILE1.FIELD2     FILE1.FIELD3
>Y
FILE2.FIELD1     FILE2.FIELD2     FILE1.FIELD3
                                  ^^^^^

>N
FILE1.FIELD4     FILE1.FIELD5     FILE1.FIELD6
^   ^^

>L
FILE1.FIELD7     FILE1.FIELD8     FILE1.FIELD9
^   ^^

>R
FILE2.FIELD7     FILE1.FIELD8     FILE1.FIELD9
FILE2.FIELD7     FILE2.FIELD8     FILE1.FIELD9
?SRH Search failure "FILE1"
*HT$$
FILE2.FIELD1     FILE2.FIELD2     FILE1.FIELD3
FILE1.FIELD4     FILE1.FIELD5     FILE1.FIELD6
FILE2.FIELD7     FILE2.FIELD8     FILE2.FIELD9
*EX$$

Ready
```

DEC USERS: Stop and think. Our disk systems are the only viable storage alternative. . .

TAKE FIVE.

1. **More reliability.** We'll provide great alternative solutions—like our Add-On, Non-Stop Reliability (ANSR) systems. . . or our 160 or 675 Mbyte, SMD-compatible Winchester storage drives—for added dependability and increased throughput.

2. **More flexibility and floor space.** We've eliminated the problem of limited choice. Name it; you've got it. And our alternative solutions take up much less floor space.

3. **Faster delivery.** We've built our solid reputation on quick delivery. Typically, we can give DEC users value-packed storage systems in 30 to 45 days ARO.

4. **Responsive service.** We have service centers both in the U.S. and Europe. We offer 4-hour response to any call within a 50-mile radius of major U.S. cities. And 24 hours/ day, 7 days/week service contracts for all key metropolitan areas.

5. **Big money savings.** For example, two RMO3s with a total capacity of 160 Mbytes cost $50,000. Three of our 160 Mbyte Winchester drives in support of a DEC VAX-11/780 give you 480 Mbytes for $42,500.

For more information about the only *viable* alternative to DEC data storage systems, contact us today at any of the addresses listed below.

System Industries

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

CIRCLE 20 ON READER CARD

LINK-TIME INITIALIZATION OF BP2 MAPS

By Peter Ehrenstrom, Lincoln Standard Enterprises, 2519 W. Peterson Ave., Chicago, IL 60659

There are several features of the Macro-11 assembler which can be used to advantage by the BP2 programmer with no knowlege of the PDP-11 instruction set. In this article I will describe and give some examples of the use of the data storage directives to initialize variables in a BP2 common or map at link time.

To begin with, the BP2 compiler generates a PSECT, or program section, for each COMMON or MAP. This PSECT is given a name and a set of attributes. This can be seen by compiling with the /MAC switch. It looks something like this:

```
.PSECT  XAMPLE,RW,D,GBL,REL,OVR
```

By using a .PSECT directive in your own macro module which is identical to the one BP2 produces you can map to the same area. That done, all that remains is to fill that area with the desired values. This is done with data storage directives.

We will first consider the .WORD, .BYTE, .FLT2, .FLT4, and .ASCII directives.

The .WORD directive generates successive words of data in the object module:

```
.WORD   15.             ; Reserves one word and initializes it
                        ; to decimal 15

.WORD   10,11           ; Reserves two words and initializes
                        ; them to octal 10 and 11
```

The .FLT2 and .FLT4 directives accomplish the same thing for floating point numbers (.FLT2 for single precision, .FLT4 for double):

```
.FLT2   75.05           ; Reserves two words of storaage and
                        ; initializes them to decimal 75.05

.FLT4   75.05           ; Reserves four words of storage and
                        ; initializes them to decimal 75.05
```

The .BYTE directive generates successive bytes of binary data in the same fashion:

```
.BYTE   13.,10.         ; Reserves two bytes and initializes
                        ; them to decimal 13 and 10
```

Both the .WORD and .BYTE directives store a value of zero if no argument is given. The .FLT2 and .FLT4, on the other hand, do not even reserve space if not given an argument. That is, a zero must be explicitly stated as an argument if it is desired.

The .ASCII directive generates a string of ASCII data:

```
.ASCII  /HELLO/         ; Reserves 5 successive bytes and stores
                        ; the ASCII string "HELLO" in them
```

The following example illustrates the use of some of these directives to initialize a simple map.

First the BP2 program, XAMPLE.B2S

```
1       ON ERROR GOTO 0                                         &
                                                                &
         !                                                      &
         !      Program    : XAMPLE.B2S                         &
         !      Programmer : Peter Ehrenstrom                   &
         !                                                      &
         !      Simple program to illustrate the use of assembler &
         !      directives to initialize a BP2 map              &
         !                                                      &

100      !                                                      &
         !                                                      &
         !      P R E - I N I T I A L I Z E D   M A P           &
         !                                                      &
         !                                                      &
                                                                &
        MAP (XXXXXX)                                            &
                                                                &
                A%,                     ! This will be set to 10      &
                B,                      ! This will be set to 20.05   &
                C$ = 5%                 ! This will be set to "HELLO" &
```

Next, a portion of the file generated by compiling /MAC, XAMPLE.MAC

```
                .
                .
                .
        .PSECT  XXXXXX,RW,D,GBL,REL,OVR
XXXXXX:
        .PSECT  $PDATA
        .WORD   0
        .WORD   0
        .WORD   16512
                .
                .
                .
```

Next, the MACRO module which will initialize the map, XXXXXX.MAC

```
        .TITLE  XXXXXX

        .RADIX  10              ; The default is octal.
                                ; This isn't necessary, but it's
                                ; certainly handy if you preter
                                ; to think in decimal.

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

XXXXXX::

        .WORD   10              ; This is A%
        .FLT2   20.05           ; This is B
        .ASCII  /HELLO/         ; This is C$

        .END
```

Now, compile XAMPLE.B2S without the /MAC switch . . . Assemble XXXXXX with the following command

```
MAC XXXXXX=XXXXXX
```

Modify the ODL to include XXXXXX.OBJ in the task . . .

```
        .ROOT   USER
USER:   .FCTR   SY:XAMPLE-XXXXXX-LIBR
LIBR:   .FCTR   LB:BP2COM/LB
        .END
```

Link, and run it

```
RUN XAMPLE
 10
 20.05
HELLO

Ready
```

Note the fact that the .PSECT directives in XAMPLE.MAC and XXXXXX.MAC are identical, as well as the one-to-one correspondence between the elements of the map and the data storage directives used to initialize them.

Now then, while using the .WORD, .BYTE, .FLT2, and .FLT4 directives in this way is rather straightforward, the .ASCII directive is something of a pain because one must count the number of characters in the argument to see that it is neither longer nor shorter than the corresponding string in the map, e.g. in the example both C$ and "HELLO" are five bytes long. I will therefore present, without explanation, a macro which will pad the string to the desired length, or generate an error at assembly time if the string is already too long. One can lift this macro from the example and use it as it stands, from the .MACRO to the .ENDM directives inclusive.

A BP2 map,

```
MAP (INIMAP)                                                    &
                                                                &
                 A%,                      ! Initialized to 100%   &
                 B,                       ! Initialized to .05    &
                 C$ = 10%                 ! "STRING1    "         &
                 D$ = 15%                 ! "STRING2         "    &
```

And a macro module to initialize it,

```
        .TITLE  EXAMPLE OF STRING PADDING MACRO
        .RADIX  10

        .MACRO  PADSTR  MAXLEN,STRING              ; Pad STRING to MAXLEN
                                                   ; with spaces
        .NCHR   STRLEN,STRING                      ; STRLEN is current length
        .IF     LT      MAXLEN-STRLEN              ; If STRING is already longer
                                                   ; than MAXLEN then generate
                                                   ; the following error message

        .ERROR  STRING  ; STRING TOO LONG
        .ENDC                                      ; End of error checking
        .ASCII  /STRING/                           ; Reserve space for STRING
        .REPT   MAXLEN-STRLEN                      ; And then fill it to MAXLEN
        .ASCII  <32>                               ; with spaces
        .ENDR                                      ; End of fill
        .ENDM                                      ; End of macro
```

```
        .PSECT  INIMAP,RW,D,GBL,REL,OVR

INIMAP::
        .WORD   100                                ; A%
        .FLT4   .05                                ; B
        PADSTR  10,<STRING1>                       ; C$, padded with 3 spaces
        PADSTR  15,<STRING2>                       ; D$, padded with 8 spaces.

        .END
```

Finally, one can see that there is no CALL or other reference to the initialization in the BP2 code itself, so one must pay close attention to internal documentation, as I have tried to show in my examples.

In the interest of brevity I have neither discussed all of the data storage directives nor been as thorough as I might have been in my treatment of those I did discuss, but I hope that, in conjunction with the MACRO-11 Language Reference Manual, these examples can be of use. ♥

RSTS/E'S SMALL BUFFERS

By Timothy P. Hart, Evans Griffiths & Hart, Inc., Lexington, MA

RSTS/E[1] V7.0 uses "small buffers" as storage for internal management purposes. A small buffer is a 32 byte block of "permanently mapped memory," a segment of the RSTS/E operating system. Other components of RSTS compete for space in this permanently mapped address space, limiting the total number of small buffers. Often the result is a shortage of small buffers with unpleasant consequences for the utility of the RSTS system.

As a rule of thumb, a minimum of 75 free buffers should be seen on a SY/F display. A system with less than 30 is likely to be in distress. There are a number of symptoms of small buffer starvation. For example, when the number of free buffers falls below a certain point (about 40), logins are inhibited; a special error message "No buffer space available" results when certain operations, such as sending an interjob message or opening a file, are attempted; I/O operations to certain devices are stalled; and deadlocks sometimes occur which stall the whole system.

A small buffer shortage can be alleviated by either increasing the number of small buffers, reducing the demand for small buffers, or both. We will discuss both approaches.

The number of small buffers which can be sysgened is limited by address space in permanently mapped monitor memory. One way to increase the space available for small buffers is to reduce the needs of those system components which are competing for this space. These components are listed below:

component	rough equivalent small buffers
device DDB's	varies; e.g. about 1.5 per KB
system-wide logicals	about .25 per logical
job slots	about .25 per slot
device slots	about .25 per slot
statistics package	20 buffers
unmapped (un-"phased") device drivers	depends on size of driver; up to 20 saved by selecting the phased disk driver

Small buffers are used by the system dynamically, for example as disk queue entries, and are also used relatively statically, for example, as storage for CCL definitions. The more dynamic uses are:

- I/O data buffers for: KB, LP, PP, PR, CR, XM, 2780
- queue entries: disk queue, FIP queue
- interjob ("send-receive") messages
- FIP-buffering of disk retrieval information

The system itself controls the use of small buffers for I/O data by limiting the number in use for any particular device when the overall system-wide supply is short. The "XBUF" LP driver will not use small buffers at all if there is more than 10K of XBUF allocated. (However, it is larger than the non-XBUF LP driver, and is permanently mapped.) The faster a device can clear its queued data the less strain on the small buffer supply — i.e. usually it is better to run terminals at the fastest possible output speed. The user can limit the number of small buffers used for messages held in queues by appropriately setting the buffer quota parameter when declaring a message receiver. One small buffer is always used for each message in a queue; additional buffers will be used for long messages only if XBUF is unallocated (but only up to the declared buffer quota limit). One buffer is tied up for the period during which a job is declared as a receiver (in addition to those which are used as message queue entries). Small buffers used for FIP buffering are always retrievable for other uses by the system when in need; if the system is in distress the advantages of FIP directory buffering will be lost; if more than 8K of XBUF is allocated, FIP buffering will always be in XBUF.

The more static uses of small buffers are the following:

job data	4 per job
CCL's	1 per CCL
run-time systems	1 per installed RTS
resident libraries	1 per installed library 1 to 3 per job using libraries: 1 if 1 or 2 window used 2 if 3, 4, or 5 3 if 6 or 7
extended block locking	1 per additional locked segment
open files	if large files option: 1 per file plus 1 per channel on which opened without large files option: 2 per open file

These more-or-less static uses are vulnerable to conservation efforts, and the most vulnerable of all is the use of small buffers for CCL's. (Among the approaches for saving CCL small buffers is a runtime system which EGH offers called "J," which reduces to one the number of small buffers used for CCL's.)

[1]RSTS/E is a trademark of Digital Equipment Corp. ♥

TIPS & TECHNIQUES

A Column For The Advanced RSTS/E User

By Steven L. Edwards, Software Techniques

Basic-Plus and MACRO

In this issue we will demonstrate how to call MACRO subroutines from Basic-Plus. While this feature is completely unsupported by DEC, and may not be available in future releases of Basic-Plus (not likely), it is useful.

This feature can give your program access to functions which are not available in Basic-Plus. Please note that this feature in itself will not breach system security. The MACRO code is executed in exactly the same mode as the Basic-Plus program. This feature can also be used to re-code functions which are too slow or clumsy, coded in Basic-Plus.

If you look at a Basic-Plus link map, you will see 2 program sections, named UI and UI2. These program sections can be overlaid with MACRO code that your program can access via sys-call 10 (UI), and sys-call 13 (UI2). After you have assembled the new code, it must be linked into a new Basic-Plus run-time system before it can be used. The relevant portion of the sysgen control file is included below.

As a demonstration of this feature, I have implemented UI as a debugging aid to dump the current Basic-Plus program to a specified file, and UI2 as a general purpose interface to user written MACRO code.

UI — Dump current program

Executing the code in the UI program section will dump the contents of the currently running Basic-Plus program to the specified file. This file can be analyzed by the CUSP BPDA. The output of the BPDA program will show the contents of all of the variables and buffers. The format of the call is:

```
2010 V$ = SYS(CHR$(10%) + "BEFORE.PMD")
     ! DUMP TO FILE BEFORE.PMD
```

UI2 — Execute machine code

Executing the code in the UI2 program section will execute the machine code string passed in the sys call. The format of the call is:

```
2010 V$ = SYS(CHR$(13) + CHR$(C)
     + CVT%$(SWAP%(5599%)) + CVT%$(SWAP%(-18208%)) + CVT%$(SWAP%(266%))
     + CVT%$(SWAP%(5599%)) + CVT%$(SWAP%(2429%)) + CVT%$(SWAP%(268%))
     + CVT%$(SWAP%(-30684%))
     + CVT%$(SWAP%(135%)))

     ! CHANGE OUR SYSTAT NAME TO '?WHAT?'
     ! INVOKE UI2, AND WORD ALIGN FOLLOWING CODE.
     ! 'MOV #^R?WH,@#FIRQB+FQNAM1'
     ! 'MOV #^RAT?,@#FIRQB+FQNAM1+2'
     ! '.NAME'
     ! 'RETURN'
```

This method of executing machine code is quite cumbersome. An alternate method is to link the assembly language program, extract the code from the task image and then execute it:

```
2010 OPEN "NEWNAM.TSK" FOR INPUT AS FILE #1%
\    FIELD #1%, 512% AS MC$
\    GET #1%, RECORD 5%
\    V$ = SYS(CHR$(13%) + CHR$(0%) + MC$)
     ! OPEN THE TASK IMAGE FILE.
     ! FIELD THE BUFFER.
     ! READ THE CODE (SKIP THE TASK HEADER STUFF).
     ! AND DO IT.
```

Note that the machine code string to be executed by UI2 must be written as position independent code and terminated by a RETURN (RTS PC).

Sysgen control file

```
  .
  .
  .
 $R LINK.SAV
 BASIC/Z,UI/A/W,BASIC=IN:RTS,DK:$ERR.STB/X/H:#177776/U:#4000/C
  .
  .
  .
 UI/C -- this is your code.
 IN:VE
 PA
  .
  .
  .
```

Ui.Lst

```
 Ui      BASIC-PLUS UN-IMPLEMENT MACRO V04.00  29-DEC-81 16:11:28 PAGE 1

                                .ENABL  LC

 000000                 Title   UI,<BASIC-PLUS UN-IMPLEMENTED SYS CALLS>,01,30-NOV-81,<SLE>
                        .TITLE  UI      BASIC-PLUS UN-IMPLEMENTED SYS CALLS
                        .IDENT  "07.001"
                        .SBTTL
                        .SBTTL  UI      BASIC-PLUS UN-IMPLEMENTED SYS CALLS
                        .SBTTL
                        .SBTTL  EDIT:           DATE:           BY:
                        .SBTTL   01           30-NOV-81         SLE
                        .SBTTL

                        ;       Written by:     STEVEN L. EDWARDS
                        ;
                        ;       Date:           30-NOV-81
                        ;
                        ;       Package:        In-House
                        ;
                        ;       Description:    IMPLEMENT THE UN-IMPLEMENTED SYS CALLS

                        ;       Copyright (C) 1981
                        ;       Software Techniques
                        ;       Los Alamitos, CA 90720
                        ;
                        ;       This  software  is  furnished  under  a license for use only on a
                        ;       single  computer system and may be copied only with the inclusion
                        ;       of  the  above  copyright  notice.   This  software, or any other
                        ;       copies  thereof,  may not be provided or otherwise made available
                        ;       to  any other person except for use on such system and to one who
                        ;       agrees  to  these  license  terms.  Title to and ownership of the
                        ;       software shall at all times remain in Software Techniques.
                        ;
                        ;       The  information  in  this  document is subject to change without
                        ;       notice  and  should  not be construed as a commitment by Software
                        ;       Techniques.
                        ;
                        ;       This  software  is  un-released  and  Software  Techniques has no
                        ;       commitment to support it at this time, unless stated elsewhere in
                        ;       writing.

                        ;
                        .Sbttl          Calling Format
                        ;
                        ;       UI
                        ;       --
                        ;       TEMP.0$ = SYS(CHR$(13) + FILENAME$)
                        ;
                        ;       Arguments:
                        ;
                        ;       Name            Description
                        ;       ----            -----------
                        ;       FILENAME$       FILE NAME TO DUMP IMAGE TO.
                        ;
                        ;       UI2
                        ;       ---
                        ;       TEMP.0$ = SYS(CHR$(10) + CHR$(0) + MC$)
                        ;
                        ;       Arguments:
                        ;
                        ;       Name            Description
                        ;       ----            -----------
                        ;       CHR$(0)         ALIGN MC$ ON WORD BOUNDRY.
                        ;       MC$             MACHINE CODE STRING.

                        ;
                        .Sbttl          Modification History
                        ;
                        ;       Ver/Edit        Date            Reason (Who)
                        ;       --------        ----            ------------

                        ;       V7.0-01         30-NOV-81       Initial conception.

                        ;
                        .Sbttl          Program Description
                        ;

                        ;       THIS MODULE IMPLEMENTS THE UN-IMPLEMENTED BASIC-PLUS SYS
                        ;       CALLS.  THE FIRST CALL (SYS 10) DUMPS THE CURRENT PROGRAM TO THE
                        ;       SPECIFIED FILE.  THE SECOND CALL (SYS 13) EXECUTES THE MACHINE
                        ;       LANGUAGE CODE STRING PASSED TO US.

                        ;
                        .Sbttl          Assembly instructions:
                        ;

                        ;       MACRO UI = COMMON/P:1, UI

                        ;
                        .Sbttl          Global Symbols
                        ;

                                .Globl  UI2, B.4

                        ;
                        .Sbttl          Variable Description and Initialization
                        ;

 000000                         .Psect  UI, RW, I, GBL, REL, OVR

         000036         CHAN    =       17*2                    ; CHANNEL 15.
         000200         FATAL   =       200                     ; FATAL ERROR BIT.
         000004         LENGTH  =       4                       ; OFFSET INTO STRING HEADER.

                        .MACRO CHKERR   ?A
                                TSTB    @#FIRQB                 ; ERROR?
                                BEQ     A                       ; GOOD.
                                TRAP    FATAL+1                 ; LET BASIC HANDLE IT.
                        A:
                        .ENDM   CHKERR

                        ;
                        .Sbttl          SYS CALL 10.
                        ;

 000000                         .Psect  UI, RW, I, GBL, REL, OVR

 000000                 UI:     PUSH    <R0,R1,R2,R3,R4,R5>     ; SAVE THE REGISTERS.
 000014  104040                 .STAT                           ; GET JOB STATS
 000016  013702                 MOV     @#XRB+XRLEN,R2          ; OLD CORE SIZE.
         000442
 000022                         CALL    SETXRB                  ; CLEAR XRB.
```

. . . continued on page 44

What Are You Doing This Weekend?

If you're spending this weekend in the computer room, think about this: DISKIT, the disk management tool kit, can save you hundreds of hours of valuable time.

DISKIT provides all of the tools you need to create and manage well-structured disks. Use DISKIT to race through backup at 10,000 blocks/minute. At the same time, DISKIT will optimize your disks, improving your system's throughput by as much as 50 percent*. A faster system means happier users and more free time for you.

Want more information? Just tear out the Business Reply Card. Or give us a call. You really don't have time not to.

*Based on information supplied by our customers.

Software
Techniques
Incorporated

SOFTWARE CONSULTING FOR DEC SYSTEMS

5242 Katella Avenue
Los Alamitos, CA 90720 USA
[213] 594-9405 [714] 995-0533

74/76 Northbrook St.
Newbury, Berkshire RG131AE UK

CIRCLE 65 ON READER CARD

THE CASE FOR NFF

By Philip G. Anthony, Technical Systems, Fidelity Bank, Philadelphia, PA

Conventional wisdom in the RSTS community has it that the new-files-first (NFF) option for account directories represents the height of folly, the path to perdition, and the surest way to slow file processing down to a crawl. The resulting complexity of directory linkage under NFF, opponents charge, increases the time required for file creation and bounces the user all around the directory during retrieval. Even DEC, which includes the option in its DSKINT procedure, recommends an automatic 'no' answer to the option. Taking the option, declares DEC, will somehow bury such frequently used programs as LOGIN irrevocably at the bottom of SY:(1,2).

In fact, the NFF option can speed file processing significantly. Careful design of accounts on the system, plus a few simple procedures, will almost entirely eliminate the disadvantages of NFF while freeing up processing time and increasing throughput.

'Careful design of accounts' can be defined — with variations depending on the applications of the system and the programming philosophy — as follows:

- All permanent and long-term data files have their own account or accounts. If possible, these accounts are located on a private disk (which need not be structured NFF). The files in question are the major data bases that will be opened for input only or for in-place update (Mode 1). They may even be extended (Mode 2) as necessary, though his leads to a highly fragmented disk structure; still, extension retains the existing directory links, adding only new retrieval information as required. Because the directories don't change, the question of old or new files first is moot.
- Executable programs in operating modules and their source files. These file types again each have their own account or accounts. They change more frequently than do the data file accounts, but still very slowly; at least, one would hope that no more than a couple of pieces of operating software need revision each week. Once again, because of the infrequency of change, directory order is inconsequential.
- Executive control files, menus, and the other overhead files that make the executable programs accessible to the users. Another nonvolatile type with its own account.
- Medium-term files. Typically, they are holding files, containing information to be posted into the main data files at the end of a specified period — daily, weekly, monthly, quarterly. Shorter-term notes, such as daily files, are often kept around for a week or a month before final posting into longer-term summary files, which may be permanent data storage or other medium-term files. Like the others, they have their own account or accounts.
- Temporary files, work files, and print files. All go into one or several accounts that contain no permanent, frequently accessed files and an absolute minimum of the medium-term variety. If there is only one such account, it ideally should be the one from which programs are run. This permits short-term work space for system utilities, such as sorts, without extra lines of code being necessary to place the work files elsewhere. If several accounts are available for these short-term file types, separating the print files from the others is desirable.
- The less said about program development accounts, the better. With a hard-working staff of programmers, directories to these accounts are going to be a bloody mess no matter what directory protocol is chosen.

The volatile accounts are the ones in which the advantages of placing new files first in the directory are realized. In my own experience, more than three quarters of file accesses in transient-file accounts involve the ten most recently created files. Better than 95 per cent go after one of the fifty newest files even in accounts containing queued print files. And more than half the files in print and work file accounts are deleted within an hour of their creation. It's a rare file in one of these accounts, and probably reflects bad programming practice or account management, that stays on disk for more than five working days. Because of the last-in, first-out character of the temporary and work files, LIFO directory ordering speeds processing noticeably.

Decreasing access overhead to temporary files becomes even more important when over-all file handling is examined. As a class, transient files are likely to be accessed more frequently than medium-term and permanent files.

Once a user enters a data file or an index file, for instance, he is likely to continue using it for a period of several minutes to several hours; and under V7.0, the file opening time is greatly reduced when somebody else already has the file open. The volatile files, on the other hand, are typically short. They are usually kept open for much shorter periods, and a program is likely to open and close several of them in the course of its run. Also, they are single-purpose entities, intended for one-user handling. Thus, they are the files on the system involving the highest processing overhead and most in need of help from a friendly directory structure.

Medium-term files are most hurt by NFF; in general, they are accessed in FIFO order. But on most installations, they are not accessed frequently. And several strategies are available for speeding their access times if it becomes important. The most obvious of these is to keep them on a private disk not structured NFF.

If this isn't an available option, the medium-term data can be stored in permanent, random-access files with control records handling space utilization within the file — intermediate files (IMFs). Instead of these files' being deleted at the end of the their useful lives, they are simply zeroed out and recycled. Being permanent, they have no directory-order problem.

Another method is to request the files' being placed at the bottom of the directory on file creation (Mode 1024). This open-for-output mode overrides the system default and defeats NFF locally.

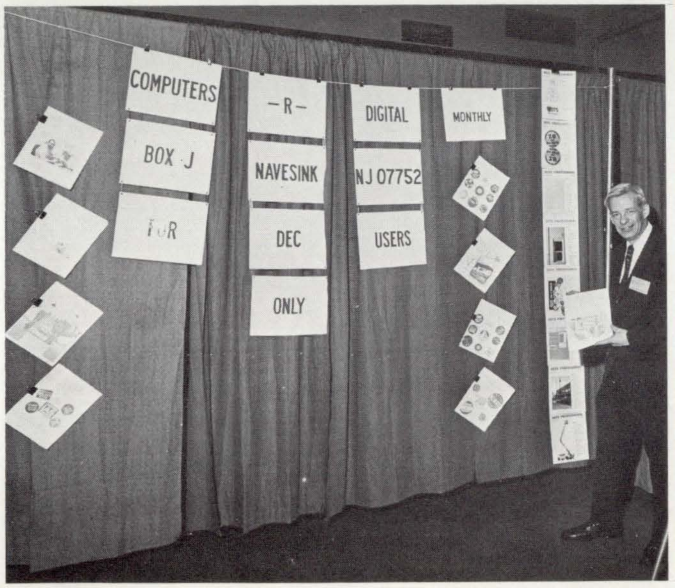
Fourth, REORDR can be used periodically to sort the medium-term file directory entries in forward order on creation date. This is probably the least efficient way to handle the problem, from the system's point of view. But it defers the inefficiency until users are off the system, covers up any programmer forgetfulness, eliminates the need for re-edit/recompile of existing programs, and throws only a minor burden on the overburdened system manager.

Finally, these files can merely be PIPped to another account and PIPped back in a preselected order (use a command file), appending the '/NE' switch to put the current date on. This last method is most applicable to executable-code accounts, bringing LOGIN and the like to the top of the stack. Add /MO:16 for executable code and do a REORDR afterward.

Yes, NFF does require daily reorder of at least all the volatile accounts on the system. But anybody who cares enough about his system to read an article on directory ordering probably uses REORDR every day on his entire disk structure already. If he doesn't, he might want to consider it. . .

New or old files first, carefully designed account structure is bound to boost system efficiency. Performance is degraded under either protocol if the user has to plow through a volatile directory to reach data, programs, or temporary files. At least one of these classes is going to take a long time to access — slowing everybody else down as well — if all three are lumped together in one account. (I recall a payroll module I once saw — but that's too painful even to think about.)

Still, file manipulation represents the single operation that does the most to slow a system down. An option that speeds access to the peskiest files on anybody's system, while affecting other files only minimally, will provide more processing time for everybody using it. ♥



RSTS Professional & Computers-R-Digital at N.Y. Coliseum, 1981.

TIPS & TECHNIQUES ... continued from page 40

```
000026  016110            MOV     LENGTH(R1),(R0)           ; LENGTH OF FILENAME.
        000004
000032  005310            DEC     (R0)                      ; ADJUST FOR FUNCTION CODE.
000034  012020            MOV     (R0)+,(R0)+               ;     "    "    "
000036  010310            MOV     R3,(R0)                   ; ADDRESS OF FILENAME.
000040                    CALL    SETFQB                    ; CLEAR FIRQB.
000044  104064            .FSS                              ; SCAN THE FILENAME.
000046                    CHKERR                            ; CHECK FOR AN ERROR.

000056  112720            MOVB    #CREFQ,(R0)+              ; CREATE FILE FUNCTION CODE.
        000004
000062  012710            MOV     #CHAN,(R0)                ; CHANNEL 15.
        000036
000066  006302            ASL     R2                        ; OLD IMAGE SIZE TIMES 2
000070  010237            MOV     R2,@#FIRQB+FQSIZ          ;    FOR FILESIZE.
        000420
000074  104000            CALFIP
000076                    CHKERR                            ; CHECK FOR AN ERROR.

000106                    CALL    SETXRB                    ; CLEAR XRB.
000112  072227            ASH     #11,R2                    ; OLD IMAGE SIZE TIMES 2048
        000011
000116  010210            MOV     R2,(R0)                   ;    FOR BYTE COUNT.
000120  012020            MOV     (R0)+,(R0)+               ;    "    "    "    "    "
000122  005720            TST     (R0)+                     ; STARTING AT 0.
000124  012710            MOV     #CHAN,(R0)                ; CHANNEL 15.
        000036
000130  104004            .WRITE
000132                    CHKERR                            ; CHECK FOR AN ERROR.

000142                    CALL    SETFQB                    ; CLEAR FIRQB.
000146  112720            MOVB    #CLSFQ,(R0)+              ; CLOSE CHANNEL FUNCTION CODE.
        000000
000152  012710            MOV     #CHAN,(R0)                ; CHANNEL NUMBER.
        000036
000156  104000            CALFIP
000160                    CHKERR                            ; CHECK FOR AN ERROR.

000170          RETURN:   POP     <R5,R4,R3,R2,R1,R0>       ; RESTORE THE REGISTERS.
000204                    RETURN                            ; BACK TO BASIC(S).

000206          SETFQB:   PUSH    <#FIRQB+FQFUN,R1>         ; SAVE R0, R1.
000214  012700            MOV     #FIRQB,R0                 ; POINT AT FIRQB
        000402
000220  012701            MOV     #<FQBSIZ/2>,R1            ; SIZE OF FIRQB IN WORDS.
        000020
000224  000407            BR      CLEAR                     ; JOIN COMMON CODE.

000226          SETXRB:   PUSH    <#XRB+XRLEN,R1>           ; SAVE R0, R1.
000234  012700            MOV     #XRB,R0                   ; POINT AT XRB.
        000442
000240  012701            MOV     #<XRBSIZ/2>,R1            ; SIZE OF XRB IN WORDS.
        000007
000244                    .BR     CLEAR                     ; JOIN COMMON CODE.

000244  005020  CLEAR:    CLR     (R0)+                     ; ZAP A WORD.
000246  077102            SOB     R1,CLEAR                  ; UNTIL WE ARE DONE.
000250                    POP     <R1,R0>                   ; RESTORE R0, R1.
000254                    RETURN                            ; BACK TO MAINLINE CODE.


                ;
                .Sbttl            SYS CALL 13.
                ;

000000                    .Psect  UI2, RW, I, GBL, REL, OVR

000000  032761  UI2::     BIT     #1,LENGTH(R1)             ; ODD STRING LENGTH?
        000001
        000004
000006  001401            BEQ     10$                       ; BRANCH IF EVEN.
000010    200G            TRAP    FATAL+B.4                 ; FATAL OUT WITH ODD ADDRESS.
          211
000012          10$:      PUSH    <R0,R1,R2,R3,R4,R5>       ; SAVE THE REGISTERS.
000026                    CALL    1(R3)                     ; CALL THEIR CODE.
000032  000167            JMP     RETURN                    ; BACK TO BASIC(S).
        000170'

        000001            .End
```

♥

CORRECTION!

David Spencer's "TTYSET Optional Patch for VT100 Width Changes," which appeared in the last issue of the RSTS PROFESSIONAL, was printed with two lines missing from the patch to the TTYSET program. Following is a reprint highlighting the missing lines. [We apologize to David and to our readers.]

```
*G/2/V<cr>
2!<tab><tab>PROGRAM<tab><tab>: TTYSET.BAS<cr>
*H/1040<tab>/V<cr>
1040<tab>GOTO 1530 IF C$="HELP" &<cr>
*I<cr>
WIDTH% = 0% &<cr>
<tab>\ <esc>
*V<cr>
<tab>\ GOTO 1530 IF C$="HELP" &<cr>
*H/1240<tab>/V<cr>
1240<tab>GOSUB 12100 &<cr>
*AI<cr>
<tab>\ WIDTH% = V% &<cr>
<esc>
*V<cr>
<tab>\ IF E%=0% AND V%>1% AND V%<256% THEN &<cr>
*H/1430<tab>/V<cr>
1430<tab>GOTO 1500 UNLESS LEN(F$) &<cr>
*0AI<cr>
1425<tab>M%(4%) = ASCII(MID(SYS(CHR$(6%)+CHR$(9%)),2%,1%))/2% &<cr>
<tab><tab><tab>IF M%(4%) = 255% &<cr>
<tab>\ IF WIDTH% = 81% AND M%(9%) = 255% THEN &<cr>
<tab><tab>C1$ = SYS(CHR$(6%)+CHR$(-5%)+CHR$(M%(4%)) &<cr>
<tab><tab>    + CHR$(155%)+"<"+CHR$(155%)+"[?3l"+CHR$(155%)+"[?2l") &<cr>
<tab><tab>! GET KB #, GET OUR OWN TO SET UP COLUMN CHANGE FOR VT100 &<cr>
<tab><tab>! IF WE HAVE XON AND WIDTH OF 80 THEN &<cr>
<tab><tab>!    BECOME VT100, SWITCH TO 80 COLUMNS, BECOME VT52 AGAIN &<cr>
<cr>
1427<tab>IF WIDTH% = 133% AND M%(9%) = 255% THEN &<cr>
<tab><tab>C1$ = SYS(CHR$(6%)+CHR$(-5%)+CHR$(M%(4%)) &<cr>
<tab><tab>    + CHR$(155%)+"<"+CHR$(155%)+"[?3h"+CHR$(155%)+"[?2l") &<cr>
<tab><tab>! IF WE HAVE XON AND WIDTH OF 132 THEN &<cr>
<tab><tab>!    BECOME VT100, SWITCH TO 132 COLUMNS, BECOME VT52 AGAIN &<cr>
<cr>
<esc>
*V<cr>
1430<tab>GOTO 1500 UNLESS LEN(F$) &<cr>
*EX<cr>
```

♥

RSTS PROFESSIONAL

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

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

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

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

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

Name ____________________

Address ____________________

____________________ Suite ________

City/State/Zip ____________________

Country ____________ Phone ( ) ____________

FREE CLASSIFIED AD WITH SUBSCRIPTION!!

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

____

____

V4.1

HOW DO YOU READ A RSTS/E DISK STRUCTURE?

By Michael H. Koplitz,

There are many articles printed whose topic is about the RSTS/E disk structure. These articles generally discuss what the basic entries are in the disk structure, but never how to write the algorithms necessary to read the disk structure. There are several ways to read the RSTS/E disk structure, but only one way will be addressed. It is assumed that the reader has some basic understanding about the disk structure. Figure A gives the RSTS/E disk structure indicating what all the entries are.

The MFD (master file directory) and UFD (user file directory) are opened in the same manner. They are opened as virtual arrays dimensioned MFD%(3583%,7%) and UFD%(3583%,7%). The algorithms will be written in BASIC-PLUS, so an understanding of it will also be assumed. The open statement for the MFD is as follows:

```
OPEN "[1,1]" + DEVICE$ FOR INPUT AS FILE #X%, MODE 8192%
```

The variable DEVICE$ is the device to inspect, MODE 8192% is for read only mode. The UFD open statement is as follows:

```
OPEN ACCOUNT$ FOR INPUT AS FILE #Y%, MODE 8192%
```

Where the variable ACCOUNT$ is the account to inspect.

The MFD label entry words are MFD%(0%,X%) where X% = 0% to 7%. Therefore:

```
     PCS = MFD%(0%,4%)
  STATUS = MFD%(0%,5%)
PACK.ID$ = RAD$(MFD%(0%,6%)) + RAD$(MFD%(0%,7%))
```

All disk directory links are of the same format:

bits 0 — 3	flags
bits 4 — 8	entry offset within block
bits 9 — 11	offset into FDCM
bits 12 — 15	block offset within cluster

The link is a combination of the offsets. The following function is an algorithm to retrieve the link and put it into a format usable in the array.

```
DEF FNGET.LINK%(LINK%)
  CLO.MASK% = 7% * 512%
  ENO.MASK% = 31% * 16%
     UL.BLO% = (SWAP%(LINK%) AND 240%) * 2%
     UL.ENO% = (LINK% AND ENO.MASK%)/16%
FNGET.LINK% = UL.BLO% + UL.CLO% + UL.ENO%
FNEND
```

To get the link to the first MFD name entry:

```
NEXT.MFD% = FNGET.LINK%(MFD%(0%,0%))
```

The value of NEXT.MFD% is the row index into the MFD% array for the first MFD name entry. Therefore the password of the first MFD name entry is:

```
PASSWORD$            = RAD$(MFD%(NEXT.MFD%,2%))
                         + RAD$(MFD%(NEXT.MFD%,3%))
```

The project-programmer number is:

```
PROJ% = SWAP%(MFD%(NEXT.MFD%,1%)) AND 255%
PROG% = MFD%(NEXT.MFD%,1%) AND 255%
```

The status byte, protection code, access count are:

```
   STATUS% = MFD%(NEXT.MFD%,4%) AND 255%
PROT.CODE% = SWAP%(MFD%(NEXT.MFD%,4%))
                  AND 255%
   ACCESS% = MFD%(NEXT.MFD%,5%)
```

The starting UFD cluster is in DCN form, it is:

```
UFD.CLUSTER = MFD%(NEXT.MFD%,7%)
```

The following algorithm converts the DCN to a physical cluster number:

```
DEVICE.CLUSTER = ((DCN -1 ) * DCS)/PCS
```

Where DCN is the device cluster number (a positive value), DCS is the disk cluster size (dependent on the hardware), PCS is the pack cluster size from the MFD label entry. To get a positive value for the DCN, if the array value is negative, the following function can be used to convert the integer into a positive number:

```
DEF FNPOS(NEG%) = 65535 - NEG%
```

Then:

```
DCN = FNPOS(DCN) IF DCN < 0%
```

The links in the MFD name entry can be found by using the FNGET.LINK function already described. When the function returns a zero, it is indicating that there are not any more entries of that kind.

The following statement retrieves the accounting entry link:

```
LINK.TO.ACCOUNTING% = FNGET.LINK%(MFD%(NEXT.MFD%,6%))
```

Then the words of the accounting entry are as follows:

```
 LSB.CPU.TIME = MFD%(LINK.TO.ACCOUNTING%,1%)
CONNECT.TIME = MFD%(LINK.TO.ACCOUNTING%,2%)
      LSB.KCT = MFD%(LINK.TO.ACCOUNTING%,3%)
  DEVICE.TIME = MFD%(LINK.TO.ACCOUNTING%,4%)
      MSB.CPU = 16384% * (SWAP%(MFD%(LINK.TO.ACCOUNTING%,5%)) AND 127%)
      MSB.KCT = 65535 * (MFD%(LINK.TO.ACCOUNTING%,5%) AND 511%)
        QUOTA = MFD%(LINK.TO.ACCOUNTING%,6%)
  UFD.CLUSTER = MFD%(LINK.TO.ACCOUNTING%,7%)
```

Then convert any of the values if they are negative by calling the FNPOS function. Once this is accomplished the following statements combine the LSB (least significant byte) and MSB (most significant byte).

```
CPU.TIME = MSB.CPU + LSB.CPU
     KCT = MSB.KCT + LSB.KCT
```

At this point all of the MFD information about an account has been gathered. Below is a simple procedure (in pseudo code) to read the MFD name entries.

```
MFD.LINK% = MFD%(0%,0%)
DO WHILE MFD.LINK% <> 0%
  MFD.LINK% = FNGET.LINK%(MFD.LINK%,0%)
  IF MFD.LINK% <> 0%
    THEN do your procedure
  ENDIF
ENDDO
```

Now that the MFD has been examined, the UFD will be examined. Several of the algorithms defined for the MFD apply to the UFD. The UFD label entry words are UFD%(0%,0%) through UFD%(0%,7%). The words in the UFD label entry are:

```
            UFD.LINK% = FNGET.LINK%(UFD%(0%,0%))
      PROJECT.NUMBER% = SWAP%(UFD%(0%,6%)) AND 255%
   PROGRAMMER.NUMBER% = UFD%(0%,6%) AND 255%
```

The link (UFD.LINK%) is to the first UFD name entry. The words of the UFD name entry are as follows:

```
           UFD.NEXT.LINK% = FNGET.LINK%(UFD%(0%,0%))
               FILENAME$ = RAD$(UFD%(UFD.LINK%,1%))
                           + RAD$(UFD%(UFD.LINK%,2%))
              EXTENSION$ = RAD$(UFD%(UFD.LINK%,3%))
       PROTECTION.CODE$ = SWAP%(UFD%(UFD.LINK%,4%)) AND 255%
                STATUS% = UFD%(UFD.LINK%,4%) AND 255%
            ACCESS.COUNT = UFD%(UFD.LINK%,5%)
ACCOUNTING.ENTRY.LINK% = FNGET.LINK%(UFD%(UFD.LINK%,6%))
  RETRIEVAL.ENTRY.LINK% = FNGET.LINK%(UFD%(UFD.LINK%,7%))
```

The links have the same structure as in the MFD. The value returned by FNGET.LINK is the row index into the UFD array.

The words to the accounting entry of the UFD are as follows:

```
   LINK.TO.ATTRIBUTE% = FNGET.LINK(UFD%(ACCOUNTING.ENTRY.LINK%,0%))
     DATE.LAST ACCESS$ = DATE(UFD%(ACCOUNTING.ENTRY.LINK%,1%))
             FILE.SIZE = UFD%(ACCOUNTING.ENTRY.LINK%,2%)
     DATE.OF.CREATION$ = DATE$(UFD%(ACCOUNTING.ENTRY.LINK%,3%))
     TIME.OF.CREATION$ = TIME$(UFD%(ACCOUNTING.ENTRY.LINK%,4%))
      RUNTIME.SYSTEM$ = RAD$(UFD%(ACCOUNTING.ENTRY.LINK%,5%))
                       + RAD$(UFD%(ACCOUNTING.ENTRY.LINK%,6%))
      FILE.CLUSTERSIZE = UFD%(ACCOUNTING.ENTRY.LINK%,7%)
```

The attribute entry words are expressed as follows:

```
LINK.TO.SECOND.ATTRIBUTE% = FNGET.LINK(UFD%(ATTRIBUTE.ENTRY%,0%))
    FILE.ORGANIZATION% = UFD%(ATTRIBUTE.ENTRY%,1%)
           RECORD.SIZE% = UFD%(ATTRIBUTE.ENTRY%,2%)
HIGHEST.VIRTUAL.BLOCK = UFD%(ATTRIBUTE.ENTRY%,4%)
            EOF.BLOCK% = UFD%(ATTRIBUTE.ENTRY%,6%)
        OFFSET.INTO.EOF = UFD%(ATTRIBUTE.ENTRY%,7%)
```

These values are used by RMS (record management services). The following algorithm evaluates the file organization word:

```
     PRINT.CONTROL% = SWAP%(FILE.ORGANIZATION%) AND 127%
     PRINT.CONTROL$ = "FORTRAN" IF PRINT.CONTROL% = 1%
     PRINT.CONTROL$ = "CARRIAGE RETURN" IF PRINT.CONTROL% = 2%
     PRINT.CONTROL$ = "UNUSED" IF PRINT.CONTROL% = 4%
     PRINT.CONTROL$ = "DOES NOT SPAN BLOCKS"
                      IF PRINT.CONTROL% = 10%
          FILE.ORG% = FILE.ORGANIZATION% AND 120%
          FILE.ORG$ = "SEQUENTIAL" IF FILE.ORG% = 0%
          FILE.ORG$ = "RELATIVE" IF FILE.ORG% = 1%
          FILE.ORG$ = "INDEXED" IF FILE.ORG% = 2%
    RECORD.FORMAT% = FILE.ORGANIZATION% AND 9%
    RECORD.FORMAT$ = "UNDEFINED" IF RECORD.FORMAT% = 0%
    RECORD.FORMAT$ = "FIXED" IF RECORD.FORMAT% = 1%
    RECORD.FORMAT$ = "VARIABLE" IF RECORD.FORMAT% = 2%
    RECORD.FORMAT$ = "VFC" IF RECORD.FORMAT% = 3%
    RECORD.FORMAT$ = "STREAM" IF RECORD.FORMAT% = 4%
```

The second file attribute entry is as follows:

```
         BUCKET.SIZE% = UFD%(LINK.TO.SECOND.ATTRIBUTE%,1%)
MAX.LENGTH.RECORD.RMS = UFD%(LINK.TO.SECOND.ATTRIBUTE%,2%)
```

The following procedure will read all of the name entries in the UFD (written in pseudo code)

```
UFD.LINK% = UFD%(0%,0%)
DO WHILE UFD.LINK% < > 0%
  UFD.LINK% = FNGET.LINK%(UFD.LINK%,0%)
  IF UFD.LINK% < > 0%
    THEN do your procedure
  ENDIF
ENDDO
```

The last part of the UFD are the retrieval entries:

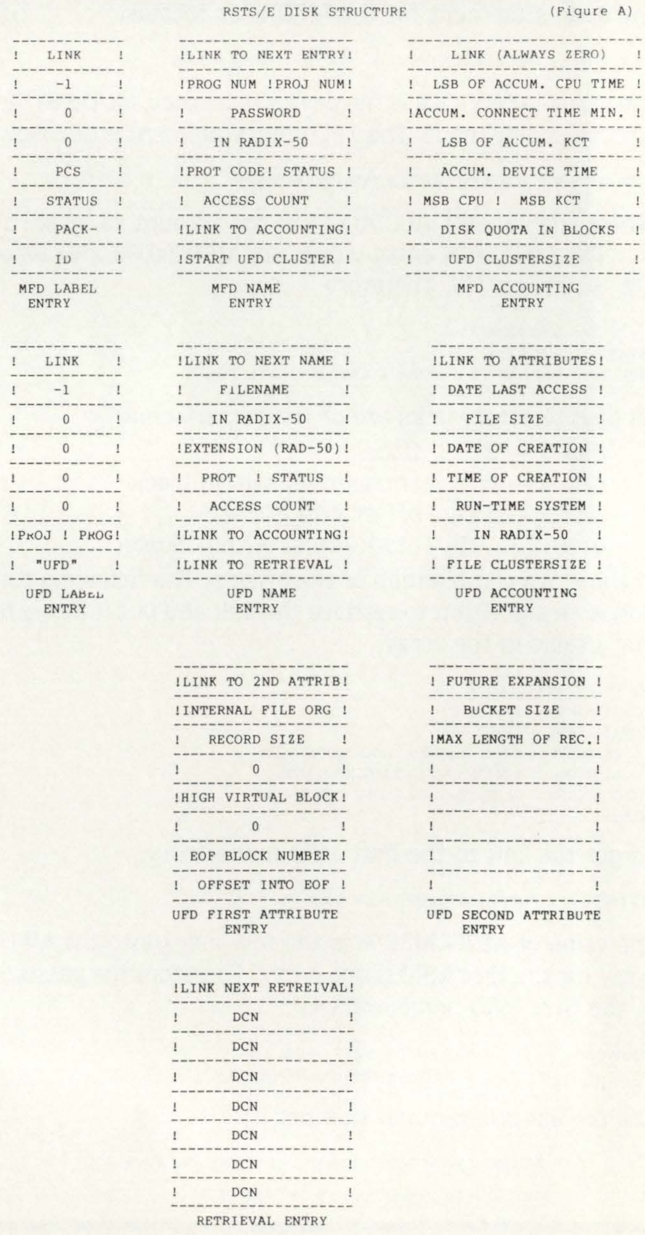
```
RETRIEVAL.ENTRY.NEXT% = FNGET.LINK(UFD%(RETRIEVAL.ENTRY.LINK%,0%))
     DEVICE.CLUSTERS% = UFD%(RETRIEVAL.ENTRY%,Z%)
```

Where Z% ranges from 1% to 7%. If a retrieval entry word is zero then there are not any more entries. If the retrieval link is zero then there are not any more retrieval entries.

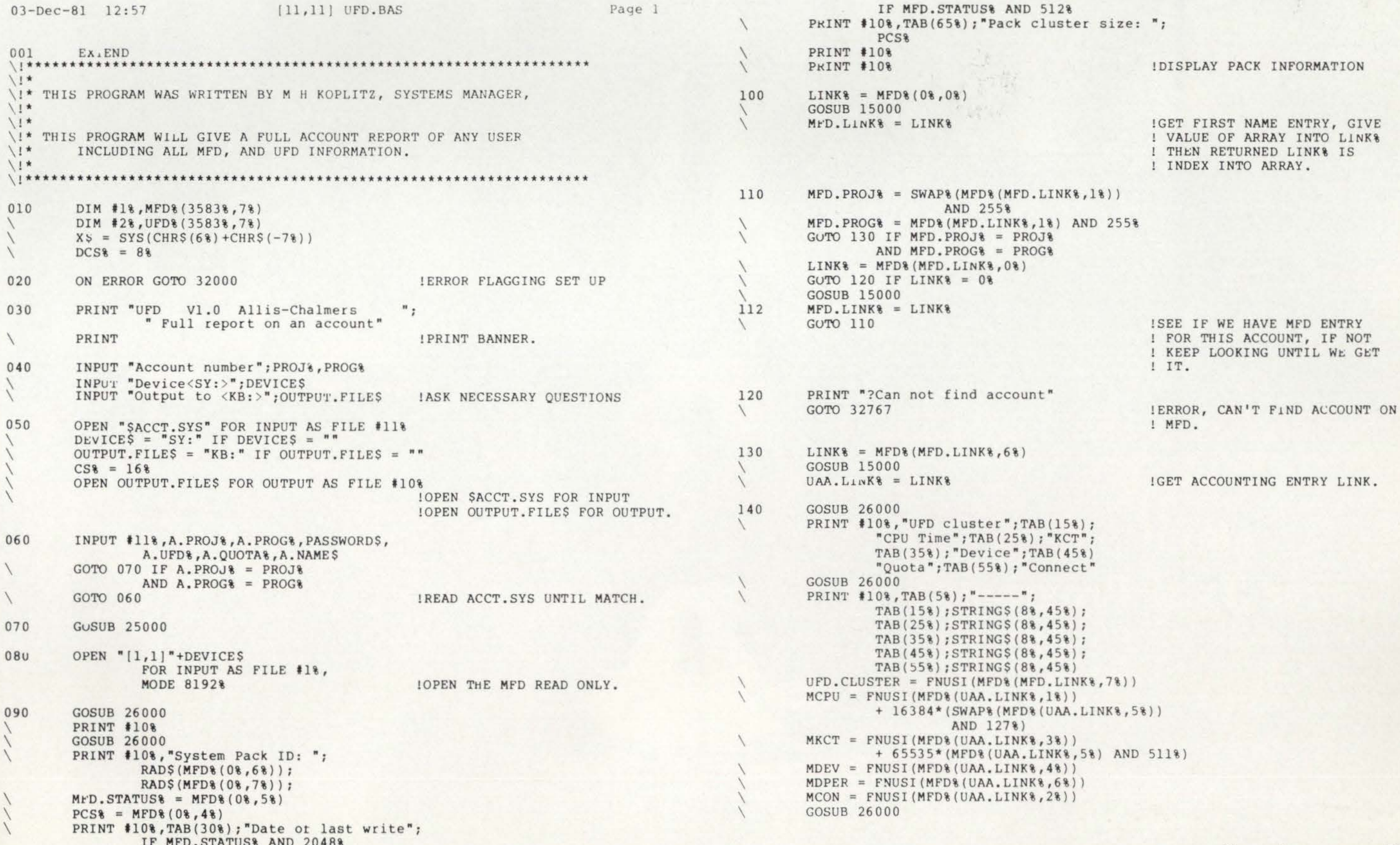
Below is a procedure (in pseudo code) to print the retrieval entries (note that the retrieval entry device clusters are in DCN form):

```
NEXT% = FNGET.LINK(UFD%(UFD.LINK%,7%))
IF NEXT% = 0% THEN FLAG% = 1%
DO WHILE FLAG = 0%
  X% = 1%
  DO WHILE UFD%(NEXT%,X%) 56 0% OR X% < 8%
    OUTPUT FNPOS(UFD%(NEXT%,X%)) IF UFD%(NEXT%,X%) < 0%
    OUTPUT UFD%(NEXT%,X%) IF UFD%(NEXT%,X%) > 0%
    X% = X% + 1%
  ENDDO
  NEXT% = FNGET.LINK(UFD%(NEXT%,0%))
  IF NEXT% = 0% THEN FLAG% = 1%
ENDDO
```

Now that the mechanics of the RSTS/E disk structure is understood a full report can be produced of all the files on a disk. The following program illustrates the kind of report that can be produced.



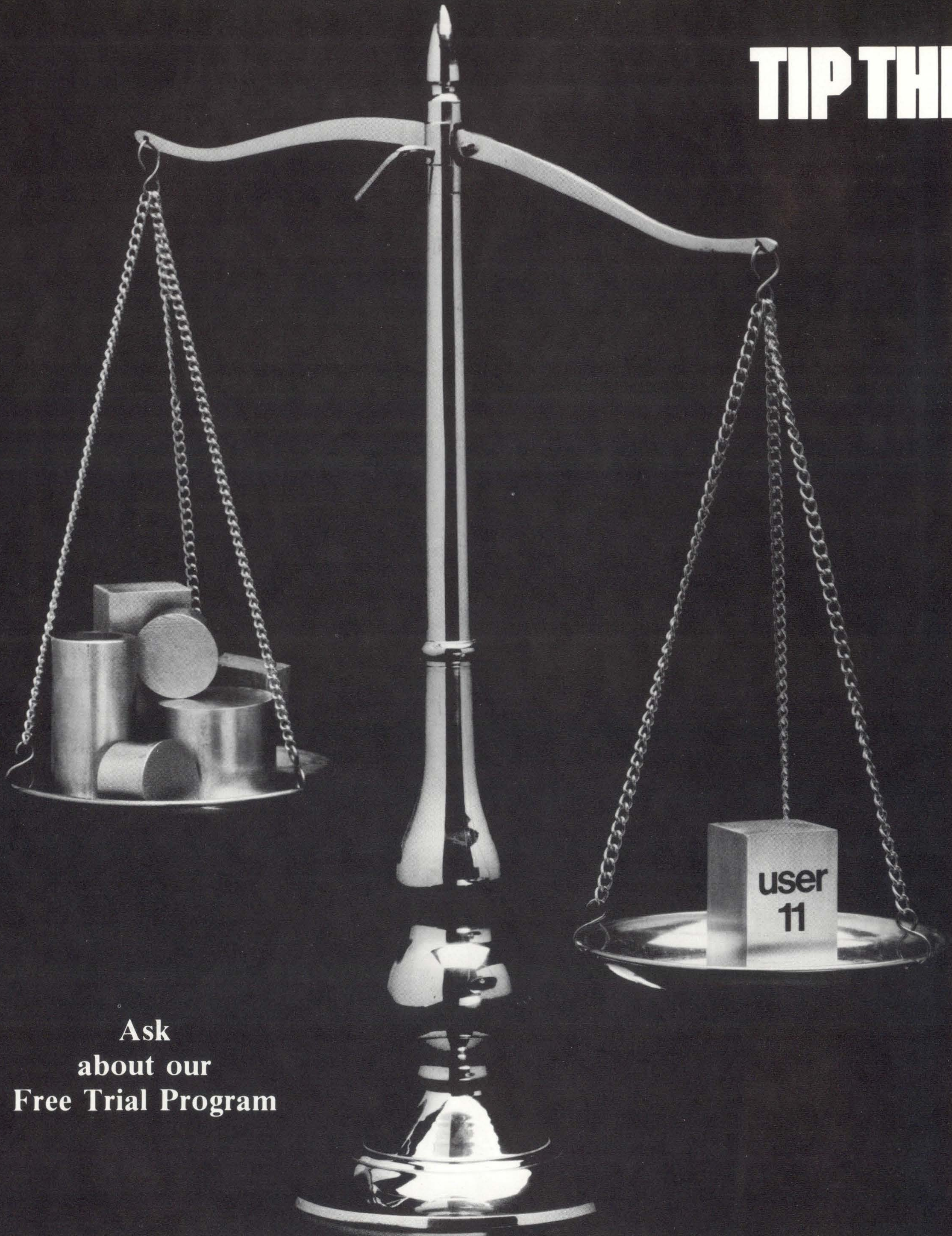
```
                          RSTS/E DISK STRUCTURE                   (Fiqure A)

-------------     --------------------     ---------------------------
!   LINK    !     !LINK TO NEXT ENTRY!     !   LINK (ALWAYS ZERO)    !
-------------     --------------------     ---------------------------
!    -1     !     !PROG NUM !PROJ NUM!     ! LSB OF ACCUM. CPU TIME  !
-------------     --------------------     ---------------------------
!     0     !     !     PASSWORD     !     !ACCUM. CONNECT TIME MIN. !
-------------     --------------------     ---------------------------
!     0     !     !   IN RADIX-50    !     !   LSB OF ACCUM. KCT     !
-------------     --------------------     ---------------------------
!    PCS    !     !PROT CODE! STATUS !     !   ACCUM. DEVICE TIME    !
-------------     --------------------     ---------------------------
!  STATUS   !     !  ACCESS COUNT    !     ! MSB CPU !  MSB KCT      !
-------------     --------------------     ---------------------------
!   PACK-   !     !LINK TO ACCOUNTING!     !  DISK QUOTA IN BLOCKS   !
-------------     --------------------     ---------------------------
!    ID     !     !START UFD CLUSTER !     !    UFD CLUSTERSIZE      !
-------------     --------------------     ---------------------------
 MFD LABEL             MFD NAME                 MFD ACCOUNTING
  ENTRY                 ENTRY                       ENTRY


-------------     --------------------        --------------------
!   LINK    !     !LINK TO NEXT NAME !        !LINK TO ATTRIBUTES!
-------------     --------------------        --------------------
!    -1     !     !    FILENAME      !        ! DATE LAST ACCESS !
-------------     --------------------        --------------------
!     0     !     !   IN RADIX-50    !        !    FILE SIZE     !
-------------     --------------------        --------------------
!     0     !     !EXTENSION (RAD-50)!        ! DATE OF CREATION !
-------------     --------------------        --------------------
!     0     !     !  PROT  ! STATUS  !        ! TIME OF CREATION !
-------------     --------------------        --------------------
!     0     !     !   ACCESS COUNT   !        !  RUN-TIME SYSTEM !
-------------     --------------------        --------------------
!PROJ ! PROG!     !LINK TO ACCOUNTING!        !    IN RADIX-50   !
-------------     --------------------        --------------------
!   "UFD"   !     !LINK TO RETRIEVAL !        ! FILE CLUSTERSIZE !
-------------     --------------------        --------------------
  UFD LABEL             UFD NAME                UFD ACCOUNTING
   ENTRY                 ENTRY                      ENTRY


                  --------------------        --------------------
                  !LINK TO 2ND ATTRIB!        ! FUTURE EXPANSION !
                  --------------------        --------------------
                  !INTERNAL FILE ORG !        !   BUCKET SIZE    !
                  --------------------        --------------------
                  !   RECORD SIZE    !        !MAX LENGTH OF REC.!
                  --------------------        --------------------
                  !        0         !        !                  !
                  --------------------        --------------------
                  !HIGH VIRTUAL BLOCK!        !                  !
                  --------------------        --------------------
                  !        0         !        !                  !
                  --------------------        --------------------
                  ! EOF BLOCK NUMBER !        !                  !
                  --------------------        --------------------
                  !  OFFSET INTO EOF !        !                  !
                  --------------------        --------------------
                  UFD FIRST ATTRIBUTE         UFD SECOND ATTRIBUTE
                        ENTRY                       ENTRY


                  ---------------------
                  !LINK NEXT RETREIVAL!
                  ---------------------
                  !       DCN         !
                  ---------------------
                  !       DCN         !
                  ---------------------
                  !       DCN         !
                  ---------------------
                  !       DCN         !
                  ---------------------
                  !       DCN         !
                  ---------------------
                  !       DCN         !
                  ---------------------
                  !       DCN         !
                  ---------------------
                     RETRIEVAL ENTRY
```



```
03-Dec-81  12:57              [11,11] UFD.BAS                  Page 1

001     Ex.END
\!*****************************************************************
\!*
\!* THIS PROGRAM WAS WRITTEN BY M H KOPLITZ, SYSTEMS MANAGER,
\!*
\!*
\!* THIS PROGRAM WiLL GIVE A FULL ACCOUNT REPORT OF ANY USER
\!*      INCLUDING ALL MFD, AND UFD INFORMATION.
\!*
\!*****************************************************************

010     DIM #1%,MFD%(3583%,7%)
\       DIM #2%,UFD%(3583%,7%)
\       X$ = SYS(CHR$(6%)+CHR$(-7%))
\       DCS% = 8%

020     ON ERROR GOTO 32000                      !ERROR FLAGGING SET UP

030     PRINT "UFD   V1.0  Allis-Chalmers     ";
                " Full report on an account"
\       PRINT                                    !PRINT BANNER.

040     INPUT "Account number";PROJ%,PROG%
\       INPUT "Device<SY:>";DEVICE$
\       INPUT "Output to <KB:>";OUTPUT.FILE$     !ASK NECESSARY QUESTIONS

050     OPEN "$ACCT.SYS" FOR INPUT AS FILE #11%
\       DEVICE$ = "SY:" IF DEVICE$ = ""
\       OUTPUT.FILE$ = "KB:" IF OUTPUT.FILE$ = ""
\       CS% = 16%
\       OPEN OUTPUT.FILE$ FOR OUTPUT AS FILE #10%
\                                                !OPEN $ACCT.SYS FOR INPUT
                                                 !OPEN OUTPUT.FILE$ FOR OUTPUT.

060     INPUT #11%,A.PROJ%,A.PROG%,PASSWORD$,
                A.UFD%,A.QUOTA%,A.NAME$
\       GOTO 070 IF A.PROJ% = PROJ%
                AND A.PROG% = PROG%
\       GOTO 060                                 !READ ACCT.SYS UNTIL MATCH.

070     GOSUB 25000

08u     OPEN "[1,1]"+DEVICE$
                FOR INPUT AS FILE #1%,
                MODE 8192%                       !OPEN THE MFD READ ONLY.

090     GOSUB 26000
\       PRINT #10%
\       GOSUB 26000
\       PRINT #10%,"System Pack ID: ";
                RAD$(MFD%(0%,6%));
                RAD$(MFD%(0%,7%));
\       MFD.STATUS% = MFD%(0%,5%)
\       PCS% = MFD%(0%,4%)
\       PRINT #10%,TAB(30%);"Date of last write";
                IF MFD.STATUS% AND 2048%
\       PRINT #10%,TAB(50%);"New files first";
                IF MFD.STATUS% AND 512%
\       PRINT #10%,TAB(65%);"Pack cluster size: ";
                PCS%
\       PRINT #10%
\       PRINT #10%                               !DISPLAY PACK INFORMATION

100     LINK% = MFD%(0%,0%)
\       GOSUB 15000
\       MFD.LINK% = LINK%                        !GET FIRST NAME ENTRY, GIVE
                                                 ! VALUE OF ARRAY INTO LINK%
                                                 ! THEN RETURNED LINK% IS
                                                 ! INDEX INTO ARRAY.

110     MFD.PROJ% = SWAP%(MFD%(MFD.LINK%,1%))
                        AND 255%
\       MFD.PROG% = MFD%(MFD.LINK%,1%) AND 255%
\       GOTO 130 IF MFD.PROJ% = PROJ%
                AND MFD.PROG% = PROG%
\       LINK% = MFD%(MFD.LINK%,0%)
\       GOTO 120 IF LINK% = 0%
\       GOSUB 15000
112     MFD.LINK% = LINK%
\       GOTO 110                                 !SEE IF WE HAVE MFD ENTRY
                                                 ! FOR THIS ACCOUNT, IF NOT
                                                 ! KEEP LOOKING UNTIL WE GET
                                                 ! IT.

120     PRINT "?Can not find account"
\       GOTO 32767                               !ERROR, CAN'T FIND ACCOUNT ON
                                                 ! MFD.

130     LINK% = MFD%(MFD.LINK%,6%)
\       GOSUB 15000
\       UAA.LINK% = LINK%                        !GET ACCOUNTING ENTRY LINK.

140     GOSUB 26000
\       PRINT #10%,"UFD cluster";TAB(15%);
                "CPU Time";TAB(25%);"KCT";
                TAB(35%);"Device";TAB(45%)
                "Quota";TAB(55%);"Connect"
\       GOSUB 26000
\       PRINT #10%,TAB(5%);"-----";
                TAB(15%);STRING$(8%,45%);
                TAB(25%);STRING$(8%,45%);
                TAB(35%);STRING$(8%,45%);
                TAB(45%);STRING$(8%,45%);
                TAB(55%);STRING$(8%,45%)
\       UFD.CLUSTER = FNUSI(MFD%(MFD.LINK%,7%))
\       MCPU = FNUSI(MFD%(UAA.LINK%,1%))
                + 16384*(SWAP%(MFD%(UAA.LINK%,5%))
                        AND 127%)
\       MKCT = FNUSI(MFD%(UAA.LINK%,3%))
                + 65535*(MFD%(UAA.LINK%,5%) AND 511%)
\       MDEV = FNUSI(MFD%(UAA.LINK%,4%))
\       MDPER = FNUSI(MFD%(UAA.LINK%,6%))
\       MCON = FNUSI(MFD%(UAA.LINK%,2%))
\       GOSUB 26000
```

. . . continued on page 59

TIP THE



Ask
about our
Free Trial Program

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

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

PRODUCTIVITY... A MATTER OF TIME.

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

PERFORMANCE... SIMPLY INCREDIBLE.

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

RELIABILITY... A PROVEN FACT.

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

SECURITY... MORE THAN RSTS.

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

STANDARDIZATION... A BYPRODUCT.

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

ADAPTABILITY... NO PROBLEM.

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

DOCUMENTATION... GOOD AND PLENTIFUL.

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

TRAINING...ALL KINDS.

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

FEATURES...ON AND ON.

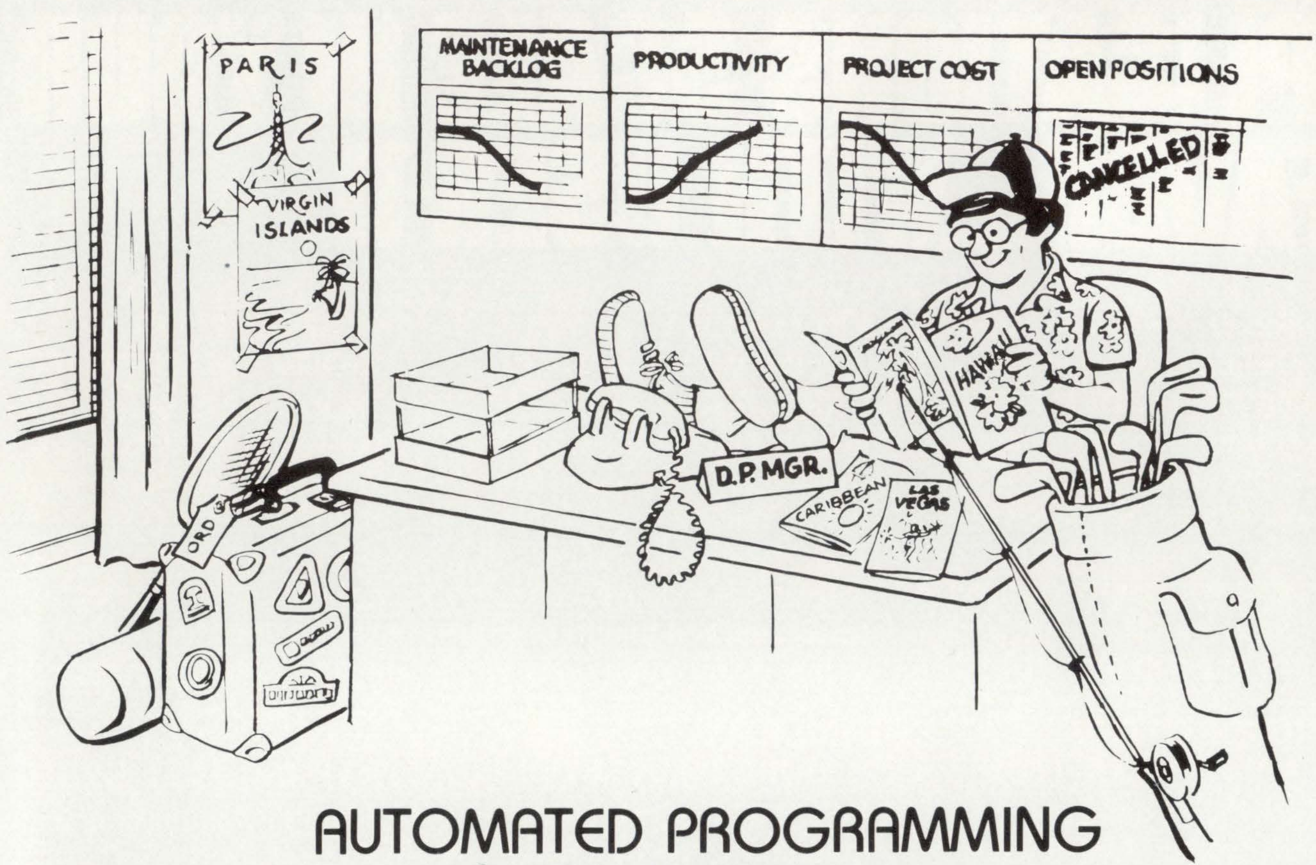
USER-11 includes virtually every facility needed to quickly construct high performance management applications—nothing else is required. If you find this hard to believe or would like more information, contact us; we will furnish you with solid user proof!

NCCS
USER
•11•

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

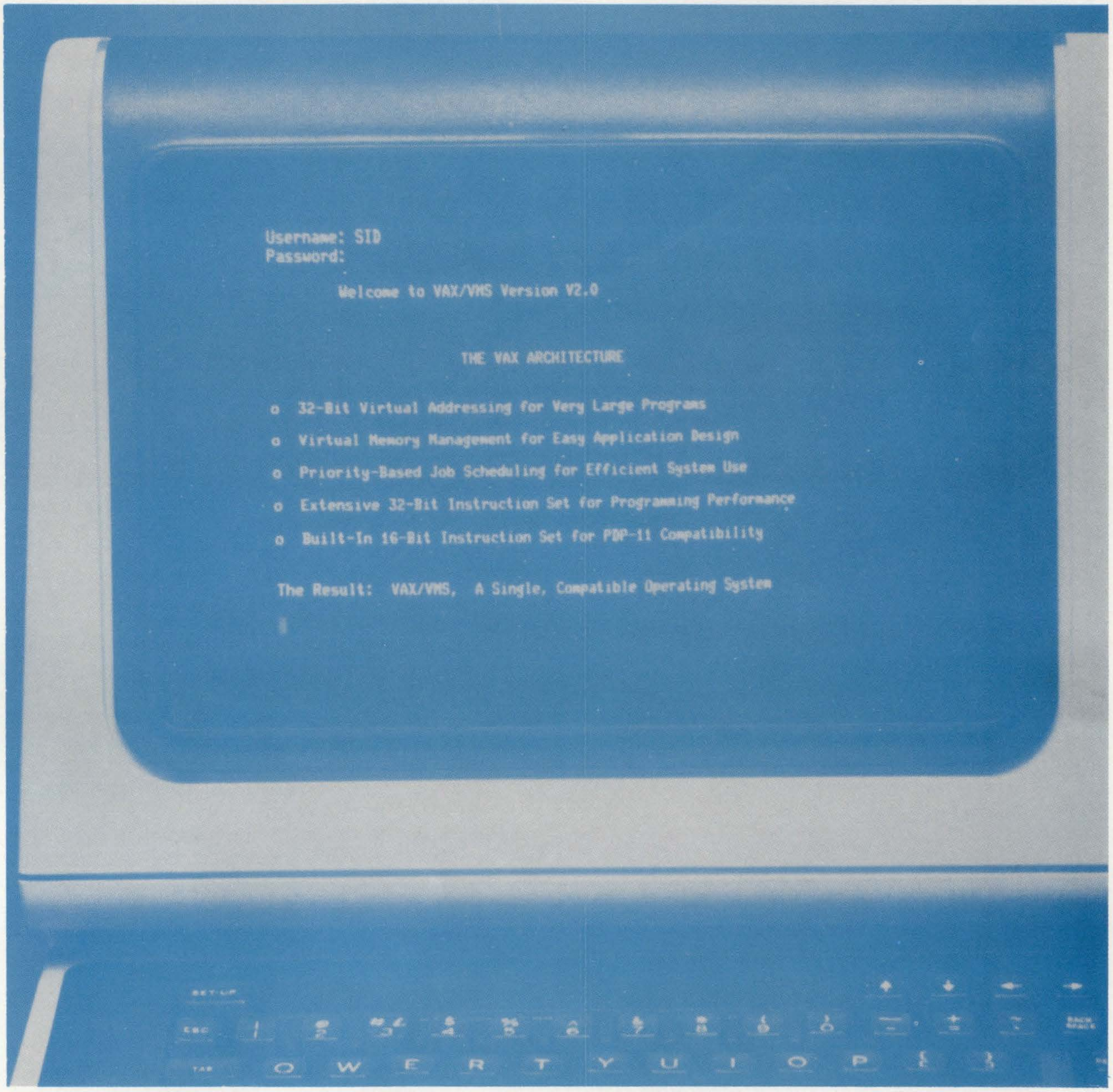
DEC and RSTS are registered trademarks of Digital Equipment Corporation.

CIRCLE 30 ON READER CARD


PARIS
VIRGIN ISLANDS
MAINTENANCE BACKLOG
PRODUCTIVITY
PROJECT COST
OPEN POSITIONS
CANCELLED
HAWAII
D.P. MGR.
CARIBBEAN
LAS VEGAS
ORD

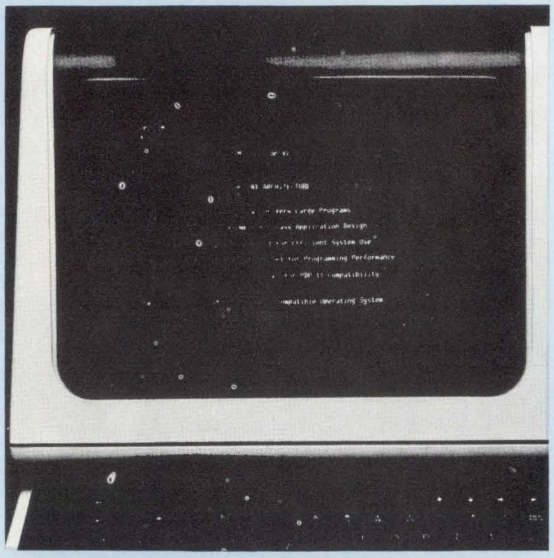
The VAX-SCENE

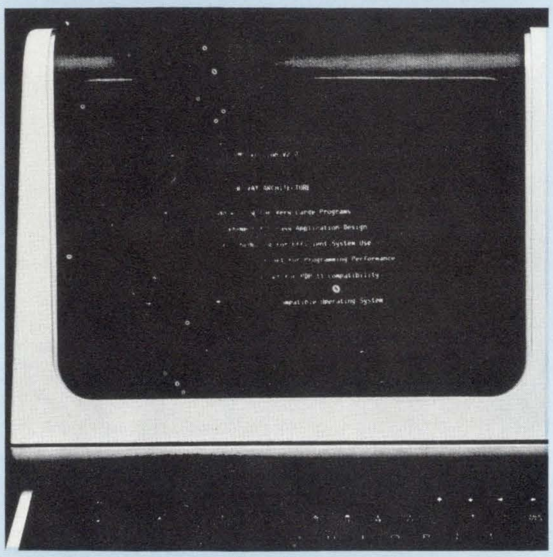
Number 6 (RSTS PROFESSIONAL, Vol. 4, No. 1) February 1982

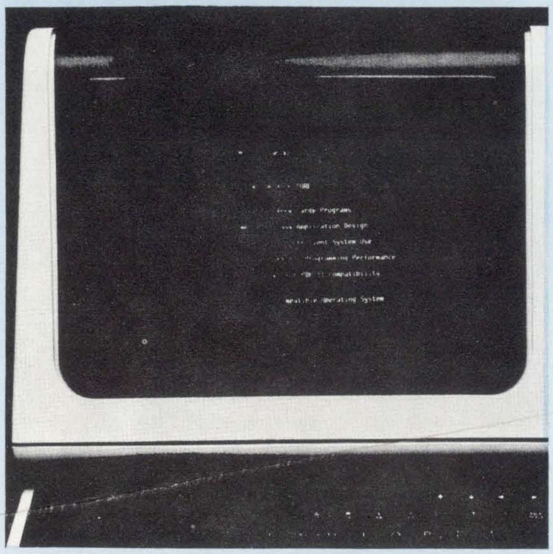


```
Username: SID
Password:

        Welcome to VAX/VMS Version V2.0


                        THE VAX ARCHITECTURE


o  32-Bit Virtual Addressing for Very Large Programs

o  Virtual Memory Management for Easy Application Design

o  Priority-Based Job Scheduling for Efficient System Use

o  Extensive 32-Bit Instruction Set for Programming Performance

o  Built-In 16-Bit Instruction Set for PDP-11 Compatibility

The Result:  VAX/VMS,  A Single, Compatible Operating System
```

INSIDE:







WRITING STRUCTURED PROGRAMS IN VAX-11 BASIC

By Al Cini, Computer Methods Corporation

A FEW THOUGHTS ON SOFTWARE STANDARDS

Despite supervisors' and managers' good intentions, software standards are often ridiculed, resented, and largely ignored by programmers. The legendary rebelliousness of coders notwithstanding, the standards themselves and the way they are introduced are usually to blame.

Most programming standards are shallow, arbitrary collections of narrowly devised rules aimed at uniformity of style rather than good programs. They tend to recognize neatness (indentation, comments, spacing) to the exclusion of quality (organization, structure, modularity). Of course, neatness (**face validity**) is an important first impression of credibility, but neat programs are not necessarily good programs. Many shops have devised a program or two which take messy code and clean it up to conform with local standards. Can we really believe that something as elusive as quality software is so simple to fabricate?

"Standard" paradoxically implies both minimally acceptable mediocrity and an ideal worth working toward. To inspire the latter rather than settle for the former, we need to look for principles which promote quality programming in substance as well as style, and to present those principles in an educational rather than legislative way. When they recognize that such standards will really help them do a better job, programmers will police conformance in themselves and their colleagues, and develop their own unique style of workmanship within them.

LEVELS OF STANDARDIZATION

Programming standards can be devised to guide software engineering at four levels:

- **Documentation.** Documentation standards govern the cosmetic characteristics of programs, including spacing, indentation, and commenting. These standards are sometimes extended to include the selection of variable names and statement labels; in BASIC dialects, some documentation standards to control the use of line numbers are frequently adopted to avoid conflicts between a "main" program segment and source components APPENDed from a library.
- **Implementation.** At the implementation level, programming standards prescribe the organization of language elements. Structured programming, which dictates rules for branching within a program, is an implementation-level standard.
- **Design.** Standards for software design govern the way in which programs and program components are combined to form systems.
- **Analysis.** Specific procedures for the development of software specifications from information provided by users are recommended by analysis-level standards.

This article will present documentation, implementation, and some design standards for VAX-11 BASIC programmers.

VAX-11 BASIC: A BRIEF INTRODUCTION

VAX-11 BASIC is the latest in a series of BASICoid languages developed by DEC for use on its VAX-11 computer family under the VMS operating system. The fact that it isn't called "BASIC-PLUS-3" misleads us to believe that it owes little to its familiar PDP-11-based predecessors; on the contrary, VAX-11 BASIC is what you might get if you crossed the flexibility and "programmer friendliness" of the BASIC-PLUS interpreter with the broadened capabilities and improved performance of the BASIC-PLUS-2 compiler. While VAX-11

BASIC and VMS offer some very unique capabilities of their own, the syntax of the language is much the same and is, for the most part, upward-compatible from the PDP-11.

We can't detail all of the compatibility issues in this article, but to help RSTS readers get a sense of the language, we will discuss some of the major differences which make programming VAX-11 BASIC a special experience. The example program at the end of this article will demonstrate many of these differences and features.

DEF/FNEND BLOCKS

VAX-11 BASIC won't tolerate branches into and out of multiline functions. Any GOTOs or GOSUBs contained inside a function must be to line numbers also contained within that function. Likewise, ON ERROR GO TO statements within functions must name locally contained statements as their targets.

Error handling within a function is local to the function in VAX-11 BASIC. When the function exits, any previously established error handling is reactivated. The following program yields these results in VAX-11 BASIC:

```
LISTNH

900      ZERO=0
910      I=1
1000     ON ERROR GO TO 1090
1010     E=I/ZERO
1020     X=FNERROR
1030     E=I/ZERO
1090     PRINT 'MAIN ERROR';ERR;'HAPPENED AT LINE';ERL
1091     RESUME 1020 IF ERL=1010
1092     RESUME 32767
10000            DEF FNERROR
10010            ON ERROR GO TO 10090
10020            E=I/ZERO
10030            FNEXIT
10090            PRINT 'FUNCTION ERROR';ERR;'HAPPENED AT LINE';ERL
10100            RESUME 10030
10110            FNEND
32767    END

Ready

RUNNH

MAIN ERROR 61 HAPPENED AT LINE 1010
FUNCTION ERROR 61 HAPPENED AT LINE 10020
MAIN ERROR 61 HAPPENED AT LINE 1030
Ready
```

The same program in BASIC-PLUS-2 behaves somewhat differently:

```
RUNNH
MAIN ERROR 61 HAPPENED AT LINE 1010
FUNCTION ERROR 61 HAPPENED AT LINE 10020
FUNCTION ERROR 61 HAPPENED AT LINE 1030
?FNEND without function call at line 10030  in 'ETEST '
```

In VAX-11 BASIC, a multi-line function is established as a separate program unit with its own internal data block contained within a main program. In BASIC-PLUS-2, the organization of a function is less formal, more closely resembling a GOSUB-type subroutine with arguments. This formal block structure can "lose" variables from function to function and drive a conversion programmer a little crazy. Consider this VAX-11 BASIC example:

```
1000     X=FNTEST1(1)
10000            DEF FNTEST1(X)
10010            PRINT 'INSIDE TEST1:';X
10020            X1=FNTEST2
10030            FNEND
11000                    DEF FNTEST2
11010                    PRINT 'INSIDE TEST2:';X
11020                    FNEND
RUNNH
INSIDE TEST1: 1
INSIDE TEST2: 0
```

The same routine in BASIC-PLUS-2:

```
RUNNH
INSIDE TEST1: 1
INSIDE TEST2: 1
```

In the VAX BASIC example, the argument X in FNTEST1, which gets a value of 1 from the main program (line 1000), is local to FNTEST1 and curiously "unknown" to FNTEST2, which is invoked by FNTEST1. To be safe, you must remember to pass any arguments in a function's argument list down through the argument lists of subordinate functions (this is almost as hard to explain as it was to debug). Note that this applies only to argument list variables. Variables outside of DEF argument lists are globally known across function boundaries, as in BASIC-PLUS and BASIC-PLUS-2.

To avoid confusion and survive conversion, the DEF* construction can be used in VAX BASIC to "revert" to BASIC-PLUS and BASIC-PLUS-2 function handling.

DATA TYPES

VAX BASIC introduces a new class of data (EXTERNAL), as well as a new data type (LONG, for longword integer). A DECLARE statement, which allows the definition of symbolic constants and explicit typing of program variables, is also provided.

VAX BASIC integers may be 16- or 32-bit values, depending on whether they are DECLAREd "WORD" or "LONG" within the program. Variables declared INTEGER, and integer variables defined in the traditional fashion by a trailing %, will be compiled as word or longword depending on a COMPILE command switch (/WORD or /LONG). As on the PDP-11, 16-bit integers can represent numbers from -32768 to +32767. The new 32-bit integer, or **longword**, can assume values from -2147483648 to +2147483647. Unfortunately, while short and long integers can be mixed within a single program via DECLARE, it is still not possible to mix both single and double precision floating point variables. The precision of floating point values is still determined on an all-or-nothing basis by a COMPILE switch (/DOUBLE).

EXTERNAL variables and constants are resolved at LINK time (the LINKER is VMS' much faster answer to TKB), and are used to reference global error constants, to define external program functions, and to reference externally defined variables. External constants are used by VMS to define symbolic status values which are returned by operating system services. Using EXTERNAL constants, a program which issues a system service can check the results against the symbol SS$__NORMAL rather than the constant 1%. This makes the program more readable and, should the value for an external constant change, the referencing program can be adapted without editing by simply re-LINKing it.

The DECLARE statement permits the definition of symbolic constants within an application program. "DECLARE INTEGER TRUE = -1" will establish an integer constant TRUE with a boolean value of "true" within the program. Likewise, I/O channel numbers and other program-dependent parameters can be established as symbolic constants using DECLARE. Variables within the program can be "typed" in DECLARE statements, eliminating the requirement that integer names end in a percent sign and string names end in a dollar sign. "DECLARE STRING X" establishes a string variable X, which is not to be confused with the well-known floating point variable of the same name. In practice, using DELCAREd variables with implicitly typed % and $ variables can be very confusing, and will demand a lot of mental adjustment.

An undocumented compiler directive in BASIC-PLUS-2 allows a programmer to establish symbolic constants within their programs. Try this some day:

```
.DEFINE .X% = -1%
.DEFINE .Y$ = "ABC"
X% = .X%
Y$ = Y$
PRINT X%, Y$
```

VAX BASIC offers the .DEFINE as well, but **use these at your own risk**. Memory management violations, inaccurate results, and reserved instruction traps may await you, and the "feature" can be withdrawn from a future release without warning.

INTERFACE TO VMS

BASIC-PLUS and BASIC-PLUS-2 programmers under RSTS/E communicate with the operating system via SYS(). VAX BASIC programmers "CALL" VMS system services as they would call their own externally compiled subroutines and functions. (Did I neglect to mention? VAX BASIC allows the separate compilation of external functions via the FUNCTION/FUNCTIONEND statements.) The interface to VMS system services is the same for most VAX languages

(CORAL and DSM are a little strange), and all VAX languages share a common run-time library of support routines. Thus, under VMS, it is possible for programs written in VAX BASIC to call or be called from COBOL, FORTRAN, or what-have-you.

PROGRAM FORMAT

An undocumented feature of VAX BASIC, described in the VAX BASIC User's Guide (in an undocumented section of the document), allows programming without backslashes and, for the most part, ampersands. This "new" program format is experimental and, to experienced BASIC-PLUS programmers, a little strange. Nonetheless it is already used so widely that there is little hope it will be withdrawn in a future release (without some new version of TRANS to get us from "there" back to "here").

PROGRAM SIZE

The most profound characteristic of VAX BASIC programming is the severely confining VAX/VMS program size restriction: a program under VMS is strictly limited to one billion bytes, and not one byte more (Note that a PDP-11 task is limited to 65K bytes, or roughly one 15000th that size.). In practice, however, a 1GB program image would occupy 4 RM05 disks (about 15 RM03s), so jobs that size are not normally found in nature. Just the same, a PDP-11 programmer who would like to open 10 or 15 indexed files in a single program, or pull a virtual array into real virtual memory for quicker processing, can get very comfortable on a VAX in short order. Subroutine overlays, RMS co-trees, and the plethora of familiar shoe-horning techniques needed to get programs to fit on the PDP-11 can be forgotten on the VAX, unless you are using BASIC-PLUS-2 in compatibility mode.

STRUCTURED VAX BASIC

Two years ago, in the RSTS Professional Vol. 1, No. 1, we adapted structured programming techniques to the BASIC-PLUS and BASIC-PLUS-2 languages. We will do the same here for VAX BASIC, and take this opportunity to add a few options we missed then.

Structured programming recognizes three elementary program "forms:"

Sequence. In a sequence structure, one program statement follows and is executed after another.

Selection. In a selection, one or another separate alternative paths through a program are taken depending on the evaluation of a logical condition.

Iteration. In an iteration, a program section is executed repeatedly while a logical condition is true.

Traditional flowcharts describing these elements are shown in figure 1. Note that each has only one entry point and exit; hence, they can be "plugged" together to make complex composites (programs) which likewise will offer only one path in and one path out. Also, the process squares in the flowcharts can be replaced by other structures in a nesting arrangement. This interchangeability has inspired an alternative representation of these elements, called Chapin charts (figure 2), which highlights the "building block" nature of these elementary structures.

In VAX BASIC, **sequence** consists simply of non-branching executable program statements. For example

```
INPUT X
Y=SQR(X)
PRINT "THE SQUARE ROOT OF";X;"IS";Y
```

A **selection** is managed by an IF-THEN as follows:

```
INPUT X
IF X<0 THEN
        PRINT "CAN'T TAKE THE SQUARE ROOT OF";X
ELSE
        Y=SQR(X)
        PRINT "THE SQUARE ROOT OF";X;"IS";Y
! ENDIF
```

Note the comment ! ENDIF in this example. The present version of VAX BASIC lacks an ENDIF instruction, so we've inserted one here as a comment to preserve visual symmetry. IF blocks are implicitly terminated by a new line number, and the lack of an explicit ENDIF can complicate matters when we include an IF within another IF.

Consider this structured English (sometimes called **pseudocode**) example:

```
IF APPLICANT IS OVER 40 THEN
    IF APPLICANT IS MALE THEN
        ASSIGN TO RISK CLASS 1
    ELSE
        ASSIGN TO RISK CLASS 2
    ENDIF.
    ADD 1 TO OVER FORTY COUNT
ELSE
    ASSIGN APPLICANT TO RISK CLASS 3
    ADD 1 TO UNDER FORTY COUNT
ENDIF.
```

VAX BASIC offers no direct representation of this example. The statement "ADD 1 TO OVER FORTY COUNT" is left dangling in the absence of an ENDIF to end the inner IF block without also ending the outer IF block. We need to re-arrange the logic a little to represent this using VAX BASIC:

```
1000        IF APPLIC.AGE%>40% THEN
                OVER.FORTY% = OVER.FORTY%+1%
                IF APPLIC.SEX$ = "M" THEN
                    APPLIC.RISK% = 1%
                ELSE
                    APPLIC.RISK% = 2%
                !ENDIF.
            ELSE
                UNDER.FORTY% = UNDER.FORTY%+1%
                APPLIC.RISK% = 3%
            !ENDIF.
1005        . . .
```

By moving our dangling statement up after the initial IF, we arrange for the inner and outer IF blocks to terminate in the same place and avoid the problem. Some more complex IF nesting may require that the blocks be separated, and that the same condition be tested more than once to represent all the alternatives.

VAX BASIC offers several iteration mechanisms, all of them essentially variants of the DO-WHILE drawn in figures 1 and 2.

```
WHILE LEN(X$)>0%
        D% = INSTR(1%, X$, ",")
        D% = LEN(X$)+1% &
                IF d% = 0%
        PRINT LEFT(X$, D%-1%)
        X$ = RIGHT(X$, D%+1%)
NEXT
```

This example extracts and prints substrings delimited by commas or by the end of the string from X$.

The VAX BASIC "UNTIL" statement is the same as "WHILE NOT." Unlike the DO-UNTIL in figure 1, VAX BASIC "UNTIL" is a leading decision loop construct. Our iteration example is easily rewritten using the equivalent UNTIL:

```
UNTIL LEN(X$)=0%
        D% = INSTR(1%, X$, ",")
                .
                .
                .
                .
NEXT
```

In either case, the loop contents are not executed at all if the terminating condition is satisfied upon entry (i.e., LEN(X$)=0%).

In a "real" (remember VAX BASIC doesn't have one) DO-UNTIL, the loop contents will always be executed at least once, and conditions established within the loop will determine whether the loop will be repeated. Sequential file processing, in which a READ must be issued to determine whether any records remain in the stream, warrants a DO-UNTIL. Such loops must be built using a DO-WHILE in VAX BASIC:

```
100     END.OF.FILE% = 11%
        DECLARE WORD TRUE = -1%, &
                     FALSE = 0%
1000    X% = FNREAD.RECORD%
        UNTIL NO.MORE.RECORDS%
            PRINT DATA.REC$
            X% = FNREAD.RECORD%
        NEXT
            .
            .
            .
2000    DEF FNREAD.RECORD%
            ON ERROR GO TO 2090
            LINPUT DATA.REC$
            NO.MORE.RECORDS% = FALSE%
            FNEXIT
2090    IF ERR = END.OF.FILE% THEN
            NO.MORE.RECORDS% = TRUE
            RESUME 2099
        ELSE
            ON ERROR GO TO 0
        !ENDIF.
2099    FNEND
```

A function reference with no arguments (FNREAD.RECORD%) is like a named GOSUB. Since the current version of VAX BASIC doesn't allow alphabetic statement labels, the "readability" of a program can sometimes be improved by using functions.

The initial FNREAD.RECORD% call at line 1000 is sometimes referred to as a "priming read," because it primes the DO-UNTIL with an initial value for the logical integer NO.MORE.RECORDS%. We could have avoided two FNREAD.RECORD% calls by setting NO.MORE.RECORDS% = FALSE initially and re-arranging the loop, but then NO.MORE.RECORDS% would be modified in **two** places rather than one, possibly creating future maintenance headaches. For those of you who object to two FNREAD.RECORD% calls, we can re-write the loop using just one by returning the NO.MORE.RECORDS% value as the result of the function:

```
1000    UNTIL FNREAD.RECORD% = END.OF.FILE%
            PRINT DATA.REC$
        NEXT
                —or—
1000    PRINT DATA.REC$ &
            UNTIL FNREAD.RECORD% = END.OF.FILE%
            .
            .
            .
2000    DEF FNREAD.RECORD%
            ON ERROR GO TO 2090
            LINPUT DATA.REC$
            FNREAD.RECORD% = 0%
            FNEXIT
2090    IF ERR = END.OF.FILE% THEN
          FNREAD.RECORD% = ERR
          RESUME 2099
        ELSE
          ON ERROR GO TO 0
        !ENDIF.
2099    FNEND
```

Of course, we could have written this in about four statements (please, no letters). Instead, we established local error trapping within an FNREAD function which houses our LINPUT statement to establish a **functional** module which performs a single service within our program. This module can be tested by itself in immediate mode, converted to an external procedure, or completely re-written without regard for the main program which invokes it. The traditional "ON ERROR GO TO 19000" defeats functional modularity and complicates maintenance. The more complex the function, the more obvious the "local error handling" advantage becomes.

We can use this opportunity to make a few observations about symbolic constants and logical variables. There are two kinds of documentation: **primary** documentation is executed by the computer (source programs, command procedures) while **secondary** documentation (flowcharts, comments within programs) is not. People tend to edit primary documentation when the need arises, but usually ignore the corresponding secondary documentation. This is called **documentation lag**, and implies that the **final say** about why software behaves as it does are the **listings of the procedures themselves**, and nothing else. Our programs are not only more readable when we use symbolic constants and logical variables, but maintenance programmers can rely on our program statements to follow our code without hunting around for comments, which are possibly inaccurate and misleading anyway.

Assigning the value 11% to the variable END.OF.FILE%, or declaring /.DEFINEing a named constant with a value of 11, allows us to test the name rather than the number in our programs. This avoids a distracting look-up during maintenance.

Integer values for all of the VAX BASIC run-time errors have been defined as EXTERNAL CONSTANTs and can be referenced from VAX BASIC, although this is not documented. This technique is a little cumbersome, though, and might not suit every situation:

```
100       EXTERNAL INTEGER CONSTANT &
                    BAS$_ACCDEVUSE &
          ,         BAS$_CANFINFIL &
          ,         BAS$_UNUERR178 &
          ,         BAS$_BADDIRDEV &
!
110       PRINT FNERR%(BAS$_ACCDEVUSE);ERT$(FNERR%(BAS$_ACCDEVUSE)) &
\         PRINT FNERR%(BAS$_CANFINFIL);ERT$(FNERR%(BAS$_CANFINFIL)) &
\         PRINT FNERR%(BAS$_UNUERR178);ERT$(FNERR%(BAS$_UNUERR178)) &
\         PRINT FNERR%(BAS$_BADDIRDEV);ERT$(FNERR%(BAS$_BADDIRDEV)) &
!
120       STOP
10000 &
DEF FNERR%(LONG ERROR_CONSTANT) &
\         FNERR%=(ERROR_CONSTANT AND (NOT X'FFFF8000'))/8% &
\FNEND &
 &
 &

$
```

Likewise, defining I/O channels, filenames, and logical values as constants at the beginning of a program offer single-source edits to these items as well as improved readability.

Another form of iteration mechanism provided by VAX BASIC is **recursion**. DEF functions can call themselves in VAX BASIC as in BASIC-PLUS and BASIC-PLUS-2. In addition, external functions and subprograms can also be used recursively. Frightening as the concept seems to most programmers, recursion is sometimes the most obvious way to represent a procedure:

```
10000     DEF FNFACTORIAL%(N%)
          IF N% < =0% THEN
             FNFACTORIAL% = 1%
          ELSE
             FNFACTORIAL% = FNFACTORIAL%(N%-1%)*N%
          !ENDIF.
10099     FNEND
```

This textbook example computes the factorial of a number by calling itself until its argument equals 1. Because the function is referenced the same way from within itself as from without (sort of like holding a mirror to a mirror), this is sometimes called **symmetrical recursion**.

Asymmetrical recursion, when a function calls itself with different argument values from those it received from the main program, can be applied to less esoteric (and much more common) commercial applications:

```
DEF FNDISPATCH% (OPTION$)
   IF OPTION$ = "ADD" THEN
      X% = FNADD%
   ELSE
   IF OPTION$ = "DEL" THEN
      X% = FNDELETE%
   ELSE
   IF OPTION$ = "SHO" THEN
      X% = FNDISPLAY%
   ELSE
   IF OPTION$ = "ALL" THEN
      X% = FNDISPATCH%("ADD")
      X% = FNDISPATCH%("DEL")
      X% = FNDISPATCH%("SHO")
   !ENDCASE.
      .
      .
      .
FNEND
```

In this example, a "dispatch" function offers three elementary alternatives ("ADD", "DEL", and "SHO") and a composite option ("ALL") which executes all three. The "ALL" option is handled by calling the dispatch routine recursively for all three elementary options.

Of course, for iteration with built-in indexing you can use FOR-NEXT. A VAX BASIC FOR loop is a leading decision (DO WHILE) iteration with a built in indexing feature which increments or decrements a variable each time the loop is executed. The terminal value of the indexing variable can be specified, or an UNTIL/WHILE condition can be used to end loop execution.

PROGRAM FORMATTING

Standard comments such as copyright notices and variable descriptions, line number and variable name conventions, subroutine naming rules — all of these are matters of local discretion and can't be discussed in detail here. You can find a sample program "shell" in your User's Guide to start with and develop your own from there.

I've found that indentation of program statements within loops and selections helps me navigate through my listings, and I include an !ENDIF comment for all my IF blocks to help me avoid accidentally omitting or miswriting clauses. I maintain MAPs as external source modules which can be APPENDed to new programs as needed, and to avoid conflict with internal program variables (unlike COBOL, VAX BASIC doesn't allow qualified names) I prefix my MAP variables with a brief mnemonic:

```
MAP (CUSTOMER__RECORD) &
     CU__ADDRESS$(4)  = 40%, &
     CU__CODE$        = 10%, &
     CU__BAL.DUE, &
     CU__TERRITORY%
```

MAKING STANDARDS (THAT) WORK

You can take the "standards" suggested in this article, add some you've read about elsewhere, and combine them with a few of your own to devise "rules" which will encourage good programming without handcuffing your programmers. Its worth taking the time to understand what a good program is, and encouraging your colleagues and employees to do the same.

SAMPLE PROGRAM

The following sample VAX BASIC program illustrates many of the points discussed in this article. This program (SYSLINK) allows a VAX to communicate with a RSTS/E (or anything else) system via asynchronous line. The terminal from which SYSLINK is invoked is established as a "virtual terminal" to the remote machine, and a rudimentary file transfer feature allows the interchange of source (i.e., sequential ASCII) data between systems. (Note: If you decide to try SYSLINK, remember to SET TERM/FULL DUPLEX on both the terminal from which you run it and the VAX terminal line into which the PDP-11 is connected.)

Al Cini will be conducting a one-day presymposium seminar at DECUS this Spring on structured methods for BASIC-PLUS, BASIC-PLUS-2 and VAX-11 BASIC programmers. For more information, call Computer Methods Corporation at (609) 778-8440, or write DECUS, Presymposium Seminars, MR2-3/E55, 1 Iron Way, Marlborough, MA 01752.

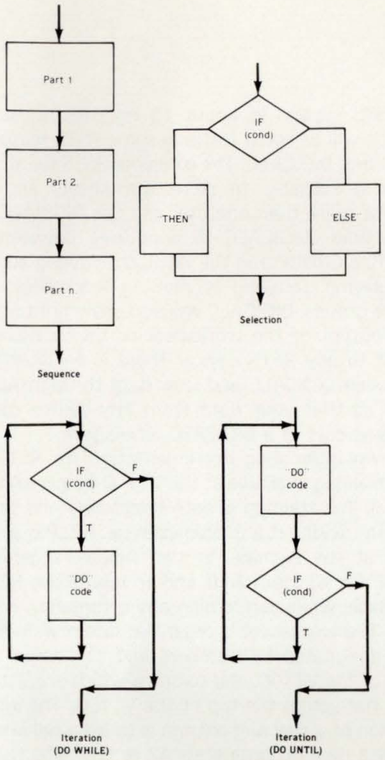


FIGURE 1.

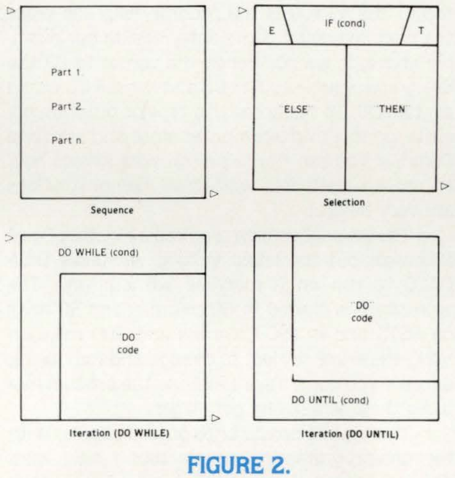


FIGURE 2.

```
TYPE SYSLINK.BAS
1 &
 !      SYSLINK - Intermachine Asynchronous Communication Program
 !
 !      The terminal from which this program is run will become
 !      a 'virtual' terminal to a remote RSTS/E (or whatever)
 !      system.  The communications link is broken by typing
 !      a CTRL/A. Typing CTRL/F at the user terminal will enable
 !      the storage of incoming characters in a VMS file, or
 !      will transmit characters from a VMS file down the line.
 !      This can be used to perform rudimentary ASCII file
 !      transfers.
 !
 !      Inter-machine VAX lines are defined in the DATA statements
 !      at lines 1090-2000.  Lines can be added/removed by adding/
 !      removing entries from this table.
 !
 !      Remember to SET TERM/FULLDUPLEX at BOTH the terminal from
 !      which SYSLINK is invoked AND the VAX terminal lines used
 !      to connect the machines.
 !
 !      Its a good idea to allocate a fairly large TTY_TYPAHDSZ
 !      with SYSGEN.  The VMS default of 78 characters is some-
 !      what lean, and can degrade system performance when SYSLINK
 !      is active.
 !
100     DECLARE INTEGER CONSTANT &
                        YES             =        -1 &
                ,       NO              =         0 &
                ,       REMOTE.TERM.EFN =        32 &
                ,       LOCAL.TERM.EFN  =        33 &
                ,       REMOTE.BUFFER   =         0 &
                ,       LOCAL.BUFFER    =         1 &
                ,       TT.BUFFSIZE     =         1 &
                ,       REMOTE.LOG.CHAN =        12 &
                ,       LOCAL.LOG.CHAN  =        13 &
                ,       END.OF.FILE     =        11 &

200     DECLARE STRING CONSTANT &
                        LOCAL.TERM      =       'TT' &
 ! &
 !
300     EXTERNAL INTEGER FUNCTION &
                        SYS$ASSIGN &
                ,       SYS$CANCEL &
                ,       SYS$CLREF &
                ,       SYS$EXIT &
                ,       SYS$QIO &
                ,       SYS$QIOW &
                ,       SYS$READEF &
                ,       SYS$WFLOR &
 !
400     EXTERNAL LONG CONSTANT &
                        SS$_NORMAL &
                ,       SS$_WASSET &
                ,       IO$_READVBLK &
                ,       IO$_TTYREADALL &
                ,       IO$_WRITEVBLK &
                ,       IO$M_NOECHO &
                ,       IO$M_NOFORMAT &
                ,       IO$M_PURGE &
                ,       IO$M_TRMNOECHO &
                ,       SS$_DEVALLOC &
 !
500     MAP(TT_BUFFAREA) &
                STRING  MSG.BUFFER(1)   =       1 &
                ,       TT.BUFFER       =       200 &
 !
 !
900     CTRL.A$                 =       CHR$(1%)
        CTRL.C$                 =       CHR$(3%)
        CTRL.F$                 =       CHR$(6%)
        CTRL.P$                 =       CHR$(16%)
        CTRL.Q$                 =       CHR$(17%)
        CTRL.S$                 =       CHR$(19%)
        CTRL.Y$                 =       CHR$(25%)
 !
        LOCAL.FUNC.CODE%  =     IO$_TTYREADALL  OR &
                                IO$M_NOECHO     OR &
                                IO$M_TRMNOECHO &
 !
        REMOTE.FUNC.CODE% =     IO$_READVBLK    OR &
                                IO$M_NOECHO     OR &
                                IO$M_TRMNOECHO &
 !
        PRT.FUNC.CODE%    =     IO$_WRITEVBLK   OR &
                                IO$M_NOFORMAT &
 !
1000    PRINT
        PRINT 'SYSLINK Inter-Processor Link Routine'
        PRINT 'Establishing Commline...';
        ERROR%=SYS$ASSIGN(LOCAL.TERM,LOCAL.TERM.CHAN%,,)
        X%=FNERROR%('Local Terminal Assignment Error',ERROR%) &
                IF FNFATAL%(ERROR%)
        RESTORE
        READ LINK.DEVICE$
        DEVICE.ASSIGNED%=NO
        UNTIL LINK.DEVICE$='*END' OR DEVICE.ASSIGNED%
                ERROR%=SYS$ASSIGN(LINK.DEVICE$,REMOTE.TERM.CHAN%,,)
                IF FNFATAL%(ERROR%) THEN
                        IF ERROR% <> SS$_DEVALLOC THEN
                                X%=FNERROR%('Link Assignment Error', &
                                        ERROR%)
                        ELSE
                                READ LINK.DEVICE$
                ELSE
                        DEVICE.ASSIGNED%=YES
                !ENDIF.
 !
1010            NEXT
 !
1020    IF DEVICE.ASSIGNED%=NO THEN
                X%=FNERROR%('Failure to Find Free Line; Try Later',4%)
                X%=FNEXIT%
 !      (ELSE)
        !ENDIF.
 !
1030    PRINT 'Line Established Using ';LINK.DEVICE$;'; Proceed.'
        PRINT
        X%=FNWRITE.MSG%(REMOTE.TERM.CHAN%, CTRL.Q$)
 !      Notify user that link is active;
 !      end any previously active STALLs (CTRL/S) on the asynchronous line.
 !
 !      ****************************************************************
 !      Table of valid asynchronous inter-machine communication lines:
 !
1091    DATA 'TTA7:'
1099    DATA '*END'
 !      ****************************************************************
2000    LOCAL.BUFFADR%  =       LOC(MSG.BUFFER(LOCAL.BUFFER))
        REMOTE.BUFFADR% =       LOC(MSG.BUFFER(REMOTE.BUFFER))
        X%=FNSETUP.READ.QIO%(REMOTE.TERM.EFN, &
                                REMOTE.TERM.CHAN%, &
                                REMOTE.BUFFADR%, &
                                REMOTE.FUNC.CODE%)
        X%=FNSETUP.READ.QIO%(LOCAL.TERM.EFN, &
                                LOCAL.TERM.CHAN%,  &
                                LOCAL.BUFFADR%, &
                                LOCAL.FUNC.CODE%)
        X%=FNPOLL.DEVICES%
        LINK.BROKEN%=(LOCAL.MSG% AND MSG.BUFFER(LOCAL.BUFFER)=CTRL.A$)
        UNTIL LINK.BROKEN%
                IF REMOTE.MSG% THEN
                        MESSAGE$=MSG.BUFFER(REMOTE.BUFFER)
                        X%=FNWRITE.MSG%(LOCAL.TERM.CHAN%, MESSAGE$)
                        X%=FNSETUP.READ.QIO%(REMOTE.TERM.EFN, &
                                        REMOTE.TERM.CHAN%, &
                                        REMOTE.BUFFADR%, &
                                        REMOTE.FUNC.CODE%)
                        IF REMOTE.LOGGING% THEN
                                IF MESSAGE$=CR THEN
                                        PRINT #REMOTE.LOG.CHAN
                                        ! Set up new RMS record.
                                ELSE
                                IF MESSAGE$=HT THEN
                                        PRINT #REMOTE.LOG.CHAN, HT;
                                ELSE
                                        PRINT #REMOTE.LOG.CHAN, MESSAGE$; &
                                                UNLESS ASCII(MESSAGE$) < 31%
                                        ! Don't bother about  control chars
                                        ! other than carriage return and
                                        ! <TAB>.
 !                              ENDIF.
 !              (ELSE)
 !              ENDIF.
 !
2020            IF LOCAL.MSG% THEN
                        MESSAGE$=MSG.BUFFER(LOCAL.BUFFER)
                        IF MESSAGE$=CTRL.C$ OR MESSAGE$=CTRL.F$ THEN &
                                X%=FNEND.LOCAL.LOGGING% &
                                        IF LOCAL.LOGGING%
                                X%=FNEND.REMOTE.LOGGING% &
                                        IF REMOTE.LOGGING%
                                X%=SYS$CANCEL(REMOTE.TERM.CHAN% BY VALUE)
                                X%=FNWRITE.MSG%(REMOTE.TERM.CHAN%, CTRL.C$) &
                                        UNLESS MESSAGE$=CTRL.F$
                                X%=FNSETUP.READ.QIO%(REMOTE.TERM.EFN, &
                                                REMOTE.TERM.CHAN%, &
                                                REMOTE.BUFFADR%, &
                                                REMOTE.FUNC.CODE% OR &
                                                        IO$M_PURGE)
                                X%=FNFILE.CONTROL% &
                                        IF MESSAGE$=CTRL.F$
                                X%=FNSETUP.READ.QIO%(LOCAL.TERM.EFN, &
                                                LOCAL.TERM.CHAN%,  &
                                                LOCAL.BUFFADR%, &
                                                LOCAL.FUNC.CODE%)
                        ELSE
                                X%=FNWRITE.MSG%(REMOTE.TERM.CHAN%, MESSAGE$)
                                X%=FNSETUP.READ.QIO%(LOCAL.TERM.EFN, &
                                                LOCAL.TERM.CHAN%,  &
                                                LOCAL.BUFFADR%, &
                                                LOCAL.FUNC.CODE%)
 !                      ENDIF.
 !              (ELSE)
 !              ENDIF.
 !
2030            IF LOCAL.LOGGING% THEN
                        X%=FNWRITE.MSG%(REMOTE.TERM.CHAN%, LOCAL.LOG.CHAR$)
 !              (ELSE)
 !              ENDIF.
 !
2090            X%=FNPOLL.DEVICES%
                LINK.BROKEN%=(LOCAL.MSG% AND MSG.BUFFER(LOCAL.BUFFER)=CTRL.A$)
        NEXT
        X%=FNEND.LOCAL.LOGGING% &
                IF LOCAL.LOGGING%
        X%=FNEND.REMOTE.LOGGING% &
                IF REMOTE.LOGGING%
        X%=FNEXIT%
 !
10000 &
DEF FNSETUP.READ.QIO%(LONG EFN, CHAN, BUFFADR, FUNCTION.CODE) &
 !
10010   ERROR%=SYS$QIO( EFN                     BY VALUE &
                ,       CHAN                    BY VALUE &
                ,       FUNCTION.CODE           BY VALUE &
                , &
                , &
                , &
                ,       BUFFADR                 BY VALUE &
                ,       TT.BUFFSIZE             BY VALUE &
                , &
                , &
                , &
                , &
                        )

        X%=FNERROR%('Failure to Establish Read QIO on EFN'+NUM1$(EFN), &
                        ERROR%) &
                IF FNFATAL%(ERROR%)

10099 &
FNEND &
 &
 &

11000 &
DEF FNPOLL.DEVICES% &
\       DECLARE LONG    EFN.CLUSTER
 !
11005   IF LOCAL.LOGGING% THEN
                LOCAL.LOG.CHAR$=FNGET.NEXT.LOCAL.LOG.CHAR$
                IF NO.MORE.LOCAL.LOG.CHARS% THEN
                        LOCAL.LOGGING%=NO
                        CLOSE LOCAL.LOG.CHAN
 !              (ELSE)
 !              ENDIF.
 !
11010   ERROR%=SYS$WFLOR(REMOTE.TERM.EFN BY VALUE, &
                                        FNBITMASK%(REMOTE.TERM.EFN) OR &
                                        FNBITMASK%(LOCAL.TERM.EFN) BY VALUE) &
                UNLESS LOCAL.LOGGING%
 !
 !      Wait for activity on either the link or local terminal lines,
 !      unless we have some from the local log file.
 !
        X%=FNERROR%('?Wait for EFN Failure',ERROR%) &
                IF FNFATAL%(ERROR%)
        REMOTE.FLAGSTATUS% =SYS$READEF(REMOTE.TERM.EFN BY VALUE, EFN.CLUSTER)
        LOCAL.FLAGSTATUS%  =SYS$READEF(LOCAL.TERM.EFN  BY VALUE, EFN.CLUSTER)
        X%=FNERROR%('Error Reading EFN',LOCAL.FLAGSTATUS%) &
                IF FNFATAL%(REMOTE.FLAGSTATUS%) OR FNFATAL%(LOCAL.FLAGSTATUS%)
        REMOTE.MSG%     =(REMOTE.FLAGSTATUS%=SS$_WASSET)
        LOCAL.MSG%      =(LOCAL.FLAGSTATUS% =SS$_WASSET)
 !
11099 &
FNEND &
 &
 &

12000 &
DEF FNWRITE.MSG%(PRINT.CHAN%,MESSAGE$) &
 !
12010   TT.BUFFER       -       MESSAGE$
        TT.BUFFSIZE%    -       LEN(MESSAGE$)
        ERROR%=SYS$QIO(                         &
                ,       PRINT.CHAN%     BY VALUE &
                ,       PRT.FUNC.CODE%  BY VALUE &
                , &
                , &
                , &
                ,       TT.BUFFER       BY REF   &
                ,       TT.BUFFSIZE%    BY VALUE &
                , &
                , &
                , &
                , &
                        )
 !
12020   X%=FNERROR%('Terminal QIOW Failure', ERRUR%) &
                IF FNFATAL%(ERROR%)
 !
12099 &
FNEND &
 &
 &

13000 &
DEF FNFATAL(ERR.CODE%)=(ERR.CODE% AND SS$_NORMAL)<>SS$_NORMAL &
 !
15000 &
DEF FNGET.NEXT.LOCAL.LOG.CHAR$ &
 !
15010   ON ERROR GO TO 15090
        IF LEN(LOCAL.LOG.BUFFER$)=0% THEN
                LINPUT #LOCAL.LOG.CHAN, LOCAL.LOG.BUFFER$
                LOCAL.LOG.BUFFER$=LOCAL.LOG.BUFFER$+CR
 !      (ELSE)
 !      ENDIF.
 !
15020   FNGET.NEXT.LOCAL.LOG.CHAR$= LEFT(LOCAL.LOG.BUFFER$,1%)
        LOCAL.LOG.BUFFER$          =RIGHT(LOCAL.LOG.BUFFER$,2%)
        NO.MORE.LOCAL.LOG.CHARS%  =NO
        FNEXIT
 !
15090   IF ERR=END.OF.FILE THEN
                NO.MORE.LOCAL.LOG.CHARS%=YES
                RESUME 15099
 !      (ELSE)
 !      ENDIF.
 !
15099 &
FNEND &
 &
 &

20000 &
DEF FNFILE.CONTROL% &
 !
20030   PRINT
        PRINT 'File Control'
        PRINT
        PRINT 'Select:'
        PRINT
        PRINT '  [RE]ceive and Store Incoming Characters (From RSTS)'
        PRINT '  [TR]ansmit Stored Characters (From VMS)'
        PRINT
        PRINT '   Option';
        LINPUT FILE.OPTION$
        FILE.OPTION$=LEFT(CVT$$(FILE.OPTION$,-1%),2%)
        IF FILE.OPTION$='RE' THEN
                PRINT TAB(5%);'Received Characters Storage File';
                LINPUT REMOTE.LOG.FILE$
                OPEN REMOTE.LOG.FILE$ FOR OUTPUT AS FILE REMOTE.LOG.CHAN
                MARGIN #REMOTE.LOG.CHAN, 200%
                REMOTE.LOGGING%=YES
        ELSE
        IF FILE.OPTION$='TR' THEN
                PRINT TAB(5%);'Transmit Contents of File';
                LINPUT LOCAL.LOG.FILE$
                OPEN LOCAL.LOG.FILE$ FOR INPUT AS FILE LOCAL.LOG.CHAN
                LOCAL.LOGGING%=YES
        ELSE
        PRINT
        PRINT TAB(5%);'? -Invalid File Option-ignored'
        PRINT
 !      ENDCASE.
 !
20099 &
FNEND &
 !
21000 &
DEF FNEND.REMOTE.LOGGING% &
 !
21010   PRINT
        PRINT 'Closing Receiving File ';REMOTE.LOG.FILE$
        PRINT
        CLOSE REMOTE.LOG.CHAN
        REMOTE.LOGGING%=NO
 !
21099 &
FNEND &
 &
 &

22000 &
DEF FNEND.LOCAL.LOGGING% &
 !
22020   PRINT
        PRINT 'Closing Transmission File ';LOCAL.LOG.FILE$
        PRINT
        CLOSE LOCAL.LOG.CHAN
        LOCAL.LOGGING%=NO
 !
22099 &
FNEND &
 &
 &

30000 &
DEF FNEXIT%     =       SYS$EXIT(SS$_NORMAL BY VALUE)
 !
30100 &
DEF FNERROR%(ERR.MSG$, ERR.CODE%) &
 !
30110   PRINT
        PRINT '?';ERR.MSG$;' -- Code:';ERR.CODE%
        PRINT
        FNERROR%=SYS$EXIT(ERR.CODE% BY VALUE)
 !
30199 &
FNEND &
 &
 &

30200 &
DEF FNFATAL%(ERROR%)    -       (ERROR% AND SS$_NORMAL)<>SS$_NORMAL &
 !
31000 &
DEF FNBITMASK%(BIT.TO.SET%) &
 !
31010   BIT.TO.SET%=0% &
                IF BIT.TO.SET% < 0%
        BIT.TO.SET%=BIT.TO.SET%-32% &
                WHILE BIT.TO.SET% >= 32%
        FNBITMASK%=2**BIT.TO.SET%
 !
31099 &
FNEND &
 &
 &

32767   END
$
```

PROGRAM NOTES

By Carl Marbach

SELECT

Ever wanted to know what records in one of your data files matched some special criteria? For example, who in your master file is over 40 years old, college educated and has an income of over $20,000 per year. The standard way of finding this out would be to create a standard program to pass through the file; you would insert the proper code in the middle of this program:

```
IF DATA = WHAT.WE.WANT THEN PRINT
```

If you are a DATATRIEVE user or are using a database query language, then you are using the constructs of that system to configure a report. Report generators are useful tools that allow english-like commands to format and create a report from a file or data base. All of these have a common problem, particularly if the file is a large one: It takes a long time to pass through the entire file.

Many standard programs in any system also pass through these files or data bases picking out certain totals, separating items, and preparing reports. Trial balances, Account distributions, Inventory listings, delinquency reporting and general file listings are examples of some reports printed on a regular basis.

Last year we began using a product from Evans, Griffiths and Hart (EG&H) called SELECT. This product reads input from any file (except RMS Indexed files), selects records based in input parameters, and outputs a file specified by user input. You can scan a 256 Byte/record & 2 per block record I/O file, look to see if bytes 23-24 are equal to 30%, and output a file with as little a 6 bytes per record up to the entire 256 bytes of the input record (more if the input record is larger). SELECT is written in MACRO, and uses its own run-time system (SELECT.SLC) to get some speed. It is FAST. So fast that we thought something was wrong the first time we ran it.

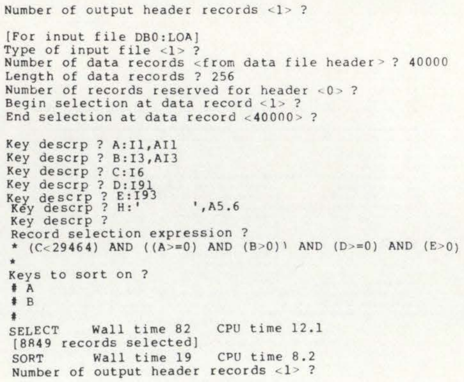
How fast is it? Our 50,000 record, 256 Byte/record file was scanned for one field (1 byte integer) > = 0 AND the second field (2 byte integer) = 0 in 12.8 CPU seconds. Thats right! 12.8! The same file but with simpler or more complex extraction expressions varied from a low of 10.5 seconds to 15.8 for a very complicated selection. All of these times included writing out a file of sub-data from the selected records. One day a bank question occurred that looked like it would be hard to answer: To whose account have we recently applied a 117.07 Payment? The problem required the scanning of our entire history file which has 20 bytes/record and is 1,085,000 records long! Over a million records! The SELECT program was run yielding several 117.07 payments from which we found the correct one. The total time to scan and write out the records which matched was 103 CPU seconds! Under two minutes to scan more than one million records!

We now routinely use SELECT to write index files for processing the main data files; This way we read only those records that we are interested in. We use SELECT to scan through a file and write out only the data fields we need for a report, allowing our programs to manipulate much less data. SELECT has revolutionized the way we think about access to our files; things that were hard to get to are now easy. Times to access data have been reduced 100 fold in many cases.

An option to the SELECT package is FSORT3. FSORT3 is EG&H's Fast Sort program. When attached to the SELECT package, it allows sorting of the output file by any of the selection fields. One Payroll file we have is kept in department order, but we often want things run alphabetically. Simple: use a SELECT procedure to pull out relative record number and name, sort on the name, and use the RRN as an index into the file. Takes 5 CPU seconds to SELECT and 3 seconds to SORT the 700 or so people in the file. Of course we use QUE-11 and the 'DO' command to make this procedure transparent to the operator, but we'll do QUE-11 another time.

SELECT is a fast selection and optional sort package from EG&H. It passes almost any kind of file with unbelievable speed. It has changed the way we access our files and the way we think about file design under RSTS.

Define A as an integer starting at byte 1 and output an integer at byte 1. Define B as an integer starting at byte 3 and output an integer at byte 3. Define C as an integer starting at byte 6 but don't write in the output file . . . write out (H) in the output file 6 blanks beginning at byte 5.



```
Number of output header records <1> ?

[For inout file DB0:LOA]
Type of inout file <1> ?
Number of data records <from data file header> ? 40000
Length of data records ? 256
Number of records reserved for header <0> ?
Begin selection at data record <1> ?
End selection at data record <40000> ?

Key descrp ? A:I1,AI1
Key descrp ? B:I3,AI3
Key descrp ? C:I6
Key descrp ? D:I91
Key descrp ? E:I93
 Key descrp ? H:'      ',A5.6
 Key descrp ?
 Record selection expression ?
 * (C<29464) AND ((A>=0) AND (B>0)) AND (D>=0) AND (E>0)
 *
 Keys to sort on ?
 # A
 # B
 #
 SELECT     Wall time 82    CPU time 12.1
 [8849 records selected]
 SORT       Wall time 19    CPU time 8.2
 Number of output header records <1> ?
```

VISICALC

A couple of years ago a graduate student did a graduate project which consisted of a program to use a micro-computer as an electronic worksheet. The computer displayed the matrix of rows and columns. Unlike a paper and pencil worksheet, this one allowed elements to be either data or equations involving other matrix elements. This program, now called VISICALC (tm), has become the best selling program in computer history. It currently runs on many micros such as APPLE and TRS-80 as well as the CP/M on DEC's VT100 add on (an we suspect DATANODES as well). We often wondered why companies with 11/70's would have to buy an APPLE to run VISICALC; now they don't. VISICALC has been incarnated on RSTS as DIGICALC. There is however no connection between the two companies who produce this competing product, and although they look very much alike; they are different.

DIGICALC is a product from WHY SYSTEMS, and run by Wayne Yarnell. DIGICALC is now available for RSTS , RSX (?), and VAX computers. The"now' may be a few weeks away, but it is definitely here. I tried DIGICALC, or rather had our financial V.P. try it to get his impressions since he uses and APPLE and VISICALC now.

While they are not the same, if you can run VISICALC on the APPLE you can run DIGICALC on a RSTS system in about 15 minutes, most of which will be spent learning some RSTS features built into DIGICALC. The complete RSTS file structure is available to store worksheets and of course more than one user can use DIGICALC at one time. DIGICALC distinguishes between a USER, a GROUP and the WORLD, allowing access for saving, recalling or revising worksheets by these groups. DIGICALC will also allow printing of the output of the worksheet or the worksheet itself to any RSTS device. There is an interface between DIGICALC and user data through ASCII files so that your data from any source could become part of a DIGICALC worksheet.

An interesting implementation for RSTS is the training available at the VT100 (required) terminal. The training is both interactive and table driven making it a unique exercise. HELP is available at the terminal in two modes: a general HELP for all commands and an interactive HELP available while performing any command.

The worksheet is organized into rows which are designated by numbers and run down the side of the VT100, and columns which are letters and run across the top of the VT100. The intersection of a row and column is called a cell and is labeled by its coordinates: A2 or B12, etc. Using the arrows on the keyboard you move the cursor to the cell you want and then simply put in data, and equation or a label. If A1 contains a 10, and B2 contains a 3, and C3 contains A1 * B2; when you calculate (either automatic or on demand) C3 would contain 30. The second line down from the top of the VT100 is the prompt/help line while the next line down is the entry line. In our example above, if we positioned the cursor to C3 the Entry line would say A1 * B2 and the cell C3 would contain 30. To really see this type of program operate, go to a micro-computer store and ask for a Demo; if you can find a person who knows how to use an electronic worksheet demonstrations are very helpful.

The original version required a Floating Point Processor but the latest version will allow DIGICALC to run on a machine without one. The workspace is limited to 60 columns and 50 rows on RSTS and is 150 Columns and 200 rows on VAX, these are subject to change and may be reduced if you don't have a FPP. As the product matures these will surely get larger.

This type of product has proved popular with the non-programming people that I have met. The accountants can relate to a spread sheet they can see, and they seem to adapt to the computerized one more easily than to a more program oriented model system. DIGICALC will do some modelling and a lot of 'fill in the blanks' type of 'what if' questions. There are better packages for real business modelling if this is what you want. The bigger and more featured ones will allow better documentation, more efficient calculations, looping for 'what if' conditions so you can run a model for inflation running from 8% to 14% in .5% increments and see what the results would be; you can even graph the results for better understanding of the output.

For easy use, friendly help, good training, large user group, low price and entry into computerized spread sheet analysis DIGICALC is a neat product. A trial version is available. Now if we only could play space invaders. ♥

HOW DO YOU READ A RSTS/E DISK STRUCTURE?

. . . continued from page 47

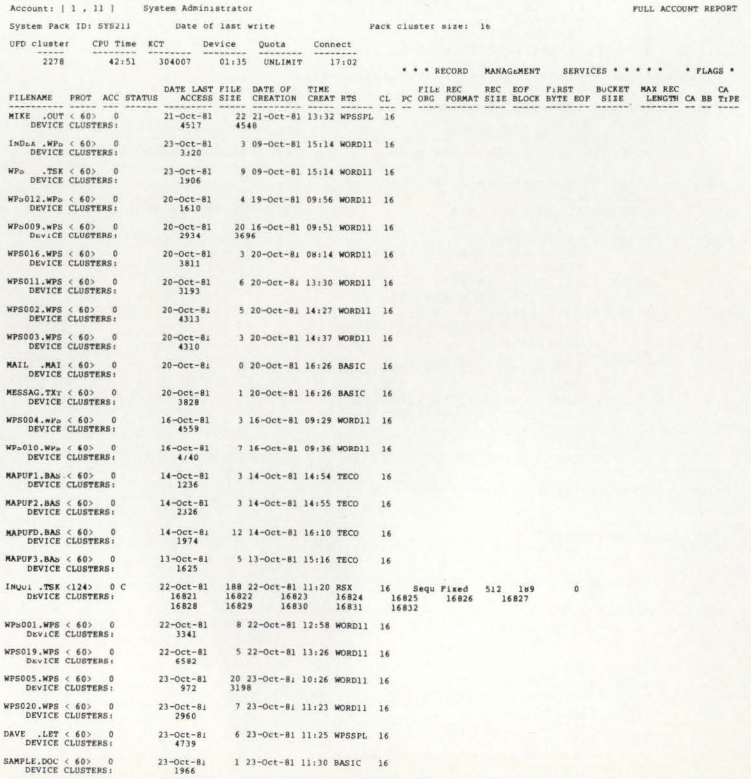
```
\       PRINT #10%,TAB(5%);
\       UFD.CLUSTER = ((UFD.CLUSTER-1)*DCS%)/PCS%
\       PRINT #10%,USING "#####",UFD.CLUSTER;
\       SECONDS = INT(MCPU/10)
\       GOSUB 20000
\       PRINT #10%,TAB(14%);EDIT.TIME$;
\       PRINT #10%,TAB(25%);
\       PRINT #10%,USING "########",MKCT;
\       SECONDS = MDEV
\       GOSUB 20000
\       PRINT #10%,TAB(34%);EDIT.TIME$;
\       PRINT #10%,TAB(45%);
\       PRINT #10%,USING "########",MDPER;
                IF MDPER > 0
\       PRINT #10%," UNLIMIT";
                IF MDPER = 0
\       SECONDS = MCON
\       GOSUB 20000
\       PRINT #10%,TAB(54%);EDIT.TIME$          !PRINT ACCOUNTING DATA.

150     CLOSE #1%
\       UFD.ACCOUNT$ = "["+NUM1$(PROJ%)+","
                + NUM1$(PROG%)+"]"
\       OPEN UFD.ACCOUNT$+DEVICE$
                FOR INPUT AS FILE #2%,
                MODE 8192%                      !OPEN UFD.

160     UFD.LINK% = UFD%

170     GOSUB 25100

200     LINK% = UFD%(UFD.LINK%,0%)
\       GOSUB 15000
\       UFD.LINK% = LINK%
\       GOTO 1000 IF UFD.LINK% = 0%             !GO GET NEXT LINK

210     GOSUB 26000
\       PRINT #10%,RAD$(UFD%(UFD.LINK%,1%));
                RAD$(UFD%(UFD.LINK%,2%));
                ".";
                RAD$(UFD%(UFD.LINK%,3%));       !PRINT FILE NAME

220     PROT% = SWAP%(UFD%(UFD.LINK%,4%)) AND 255%
\       STAT% = UFD%(UFD.LINK%,4%) AND 255%
\       PRINT #10%,TAB(11%);"<";
\       PRINT #10%,USING "###",PROT%;
\       PRINT #10%,">";TAB(17%);
\       PRINT #10%,USING "###",
                UFD%(UFD.LINK%,5%);
\       PRINT #10%,TAB(21%);
\       STAT$ = ""
\       STAT$ = "D"
                IF STAT% AND 128%
\       STAT$ = STAT$ + "Q"
                IF STAT% AND 32%
\       STAT$ = STAT$ + "C"
                IF STAT% AND 16%
\       STAT$ = STAT$ + "O"
                IF STAT% AND 8%
\       STAT$ = STAT$ + "W"
                IF STAT% AND 4%
\       STAT$ = STAT$ + "P"
                IF STAT% AND 2%
\       PRINT #10%,STAT$;                       !PRINT PROT, ACCESS.
\!**********************************************
\!*
\!* STATUS TABLE:
\!*     P  =  FILE IS PLACED
\!*     W  =  WRITE ACCESS NOT GIVEN
\!*     O  =  FILE OPEN UPDATE MODE
\!*     C  =  CONTIGUOUS
\!*     Q  =  NO DELETE OR RENAME
\!*     D  =  FILE MARKED FOR DELETION
\!*
\!**********************************************

250     LINK% = UFD%(UFD.LINK%,6%)
\       GOSUB 15000
\       UAA.UFD.LINK% = LINK%
\       GOTO 450 IF UAA.UFD.LINK% = 0%          !GET THE ACCOUNTING ENTRY

260     LINK% = UFD%(UAA.UFD.LINK%,0%)
\       GOSUB 15000
\       ATTRIBUTE.LINK% = LINK%                 !GET THE ATTRIBUTE LINK

270     PRINT #10%,TAB(28%);
                DATE$(UFD%(UAA.UFD.LINK%,1%));
                TAB(38%);
\       PRINT #10%,USING "#####",
                UFD%(UAA.UFD.LINK%,2%);
\       PRINT #10%,TAB(44%);
                DATE$(UFD%(UAA.UFD.LINK%,3%));
                TAB(54%);
\       PRINT #10%,
                LEFT(TIME$(UFD%(UAA.UFD.LINK%,4%)),5%);
                TAB(60%);
                RAD$(UFD%(UAA.UFD.LINK%,5%));
                RAD$(UFD%(UAA.UFD.LINK%,6%));
                TAB(67%);
\       PRINT #10%,USING
                "###",UFD%(UAA.UFD.LINK%,7%);   !ACCOUNTING DATA OUT

300     GOTO 450 IF ATTRIBUTE.LINK% = 0%


360     PRINT.CONTROL% =
                SWAP%(UFD%(ATTRIBUTE.LINK%,1%))
                        AND 127%
\       FILE.ORG% = UFD%(ATTRIBUTE.LINK%,1%)
                        AND 120%
\       RECORD.FORMAT% = UFD%(ATTRIBUTE.LINK%,1%)
                        AND 9%                  !GET FIRST WORD OF ACCOUNTING
                                                ! ENTRY

370     PRINT #10%,TAB(71%);
\       PRINT #10%,"FO" ;
                IF (PRINT.CONTROL% AND 1%) = 1%
\       PRINT #10%,"CR";
                IF (PRINT.CONTROL% AND 2%) = 2%
\       PRINT #10%,"UN";
                IF (PRINT.CONTROL% AND 4%) = 4%
\       PRINT #10%,"NO";
                IF (PRINT.CONTROL% AND 10%) = 10%
\                                               !INDICATE PRINT CONTROL

380     PRINT #10%;TAB(74%);
\       PRINT #10%;"Sequ";
                IF FILE.ORG% = 0%
\       PRINT #10%,"Rela";
                IF FILE.ORG% = 1%
\       PRINT #10%,"Inde";
                IF FILE.ORG% = 2%               !INDICATE FILE ORGANIZATION

390     PRINT #10%,TAB(79%);
\       PRINT #10%,"Undeti";
                IF RECORD.FORMAT% = 0%
\       PRINT #10%,"Fixed";
                IF RECORD.FORMAT% = 1%
\       PRINT #10%,"Variab";
                IF RECORD.FORMAT% = 2%
\       PRINT #10%,"VFC";
                IF RECORD.FORMAT% = 3%
\       PRINT #10%,"Stream";
                IF RECORD.FORMAT% = 4%          !INDICATE RECORD FORMAT

400     PRINT #10%,TAB(86%);
\       PRINT #10%,USING "####",
                UFD%(ATTRIBUTE.LINK%,2%);
\       PRINT #10%,TAB(91%);
\       PRINT #10%,USING "#####",
                UFD%(ATTRIBUTE.LINK%,6%);
\       PRINT #10%,TAB(97%);
\       PRINT #10%,USING "#######",
                UFD%(ATTRIBUTE.LINK%,7%);       !REST OF ATTRIBUTE STUFF

410     LINK% = UFD%(ATTRIBUTE.LINK%,0%)
\       GOSUB 15000
\       SECOND.A.LINK% = LINK%
\       GOTO 450 IF SECOND.A.LINK% = 0%         !GET SECOND ATTRIBUTE ENTRY

420     PRINT #10%,TAB(106%);
\       PRINT #10%,USING
                "######",UFD%(SECOND.A.LINK%,1%);
\       PRINT #10%,TAB(114%);
\       PRINT #10%,USING "#######",
                UFD%(SECOND.A.LINK%,2%);        !PRINT OUT SOME RMS STUFF

450     PRINT #10%,TAB(122%);"ON";
                IF UFD%(UFD.LINK%,0%) AND 4%
\       PRINT #10%,TAB(125%);"YE";
                IF UFD%(UFD.LINK%,0%) AND 2%
\       PRINT #10%,TAB(128%);"SEQ";
                IF UFD%(UFD.LINK%,6%) AND 4%
                AND UFD%(UFD.LINK%,0%) AND 4%
\       PRINT #10%,TAB(128%);"RAN";
                IF (UFD%(UFD.LINK%,6%) AND 4%) = 0%
                AND UFD%(UFD.LINK%,0%) AND 4%
\                                               !PRINT OUT FLAGS

500     PRINT #10%
\       PRINT #10%,"    DEVICE CLUSTERS:";      !PRINT BANNER

510     LINK% = UFD%(UFD.LINK%,7%)
515     GOSUB 15000
\       UAR.LINK% = LINK%
\       GOTO 900 IF UAR.LINK% = 0%              !GET RETRIEVAL ENTRIES.

520     GOSUB 26000
\       FOR X% = 1% TO 7%
\               GOTO 900
                  IF UFD%(UAR.LINK%,X%) = 0%
\               PRINT #10%,TAB(20%+X%*10%);
\               UNSIGNED.TEST =
                        UFD%(UAR.LINK%,X%)
\               GOSUB 600
\               DISK.CLUSTER =
                  ((UNSIGNED.TEST-1)*DCS%)/PCS%
\               PRINT #10%,USING "#####",
                        DISK.CLUSTER;
\       NEXT X%                                 !PRINT OUT RETRIEVAL ENTRIES,
                                                ! DISK.CLUSTER = ((DCN-1)*DCS)
                                                !                /PCS

530     PRINT #10%
\       LINK% = UFD%(UAR.LINK%,0%)
\       GOTO 515                                !GET MORE UNTIL ZERO IN LINK%


600     GOTO 610 IF UNSIGNED.TEST > 0
\       UNSIGNED.TEST = 32768 + UNSIGNED.TEST
                        + 32767                 !CONVERT NEGATIVE TO INTEGER

610     RETURN                                  !ALL DONE


900     PRINT #10%
\       GOSUB 26000
\       PRINT #10%
\       GOTO 200

1000    GOTO 32767

15000 !*************************************************************
\     !*
\     !* THIS SECTION CALCULATES A LINK%
\     !*
\     !*************************************************************

15010   CLO.MASK% = 7% * 512%
\       ENO.MASK% = 31% * 16%
\       UL.BLO% = (SWAP%(LINK%) AND 240%)*2%
\       UL.CLO% = ((LINK% AND CLO.MASK%)/16%)*CS%
\       UL.ENO% = (LINK% AND ENO.MASK%)/16%

15050   LINK% = UL.BLO% + UL.CLO% + UL.ENO%
\       RETURN

15100 !************************************************************
\     !*
\     !* THIS FUNCTION CONVERTS UNSIGNED INTEGER TO FLOATING PT.
\     !*
\     !************************************************************

15110   DEF FNUSI(USINT%)
\       USTMP = USINT%
\       USTMP = USTMP + 65535 IF USINT% < 0%
\       FNUSI = USTMP
\       FNEND

\       PRINT #10%,TAB(15%);MCPU;TAB(25%);
                MKCT;TAB(35%);MDEV;
                TAB(45%);MDPER;TAB(55%);
                MCON                            !PRINT ACCOUNTING DATA.
```

```
20000 !***************************************************************
\      !*
\      !* CONVERT NUMBER TO HOURS, MINUTES, SECONDS.
\      !*
\      !***************************************************************

20010   HOURS% = INT(SECONDS/3600%)
\       TEMP.SECONDS% = SECONDS - 3600% * HOURS%
\       MINUTES% = INT(TEMP.SECONDS%/60%)
\       TEMP.SECONDS2% =
                INT(TEMP.SECONDS% - 60% * MINUTES%)
\                                                 !CALCULATE HRS, MN, SECS

20020   HOURS$ = NUM1$(HOURS%)
\       HOURS$ = SPACE$(3%-LEN(HOURS$)) + HOURS$
\       MINUTES$ = NUM1$(MINUTES%)
\       MINUTES$ = STRING$(2% - LEN(MINUTES$),48%)
                     + MINUTES$
\       SECONDS$ = NUM1$(TEMP.SECONDS2%)
\       SECONDS$ = STRING$(2% - LEN(SECONDS$),48%)
                     + SECONDS$               !CONVERT TO STRINGS

20030   EDIT.TIME$ = HOURS$
                IF HOURS% > 0%
\       EDIT.TIME$ = "   "
                IF HOURS% = 0%
\       EDIT.TIME$ = EDIT.TIME$ + ":"
                IF HOURS% > 0%
\       EDIT.TIME$ = EDIT.TIME$ + " "
                IF HOURS% = 0%
\       EDIT.TIME$ = EDIT.TIME$ + MINUTES$
                IF HOURS% > 0%
                     OR MINUTES% > 0%
\       EDIT.TIME$ = EDIT.TIME$ + "  "
                IF MINUTES% = 0%
                     AND HOURS% = 0%
\       EDIT.TIME$ = EDIT.TIME$ + ":"
                IF MINUTES% > 0%
                     OR HOURS% > 0%
\       EDIT.TIME$ = EDIT.TIME$ + " "
                IF MINUTES% = 0%
                     AND HOURS% = 0%
\       EDIT.TIME$ = EDIT.TIME$ + SECONDS$      !RETURN FORMATED TIME

20040   RETURN

25000 !*******************************************
\      !*
\      !* THIS SECTION PRINTS OUT NEW PAGE
\      !*       HEADINGS.
\      !*
\      !*******************************************

25005   PRINT #10%,CHR$(12%)
\       P1% = P1% + 1%
\       PRINT #10%,"UFD  "DATE$(0);"  ";TIME$(0%);
             TAB(40%);"* * * ALLIS-CHALMERS HTD * * *";
             TAB(120);
             "PAGE ";P1%
\       PRINT #10%
\       PRINT #10%,TAB(45%);"FULL ACCOUNT REPORT"
\       PRINT #10%,"Account: [";PROJ%;",";PROG%;
             "]      ";A.NAME$
\       PRINT #10%
\       LO% = 5%
\       RETURN                                  !PRINT HEADINGS.

25100 !*******************************************
\      !*
\      !* PRINTS COLUMN HEADINGS.
\      !*
\      !*******************************************

25115   PRINT #10%,TAB(71%);"* * * RECORD   MANAGEMENT";
              "    SERVICES * * * * *";
              TAB(122%);"* FLAGS *"
\       PRINT #10%
\       PRINT #10%,TAB(28%);"DATE LAST";TAB(38%);
              "FILE";TAB(44%);"DATE OF";
              TAB(54%);"TIME";TAB(74%);"FILE";
              TAB(79%);"REC";TAB(86%);"REC";
              TAB(91%);"EOF";TAB(97%);"FIRST";
              TAB(106%);"BUCKET";TAB(114%);"MAX REC";
              TAB(128%);"CA"
\       PRINT #10%,"FILENAME";TAB(11%);"PROT";
              TAB(17%);"ACC";TAB(21%);"STATUS";
              TAB(31%);"ACCESS";TAB(38%);"SIZE";
              TAB(44%);"CREATION";TAB(54%);"CREAT";
              TAB(60%);"RTS";TAB(67%);"CL";
              TAB(71%);"PC";
              TAB(74%);"ORG";TAB(79%);"FORMAT";
              TAB(86%);"SIZE";TAB(91%);"BLOCK";
              TAB(97%);"BYTE EOF";TAB(107%);
              "SIZE";TAB(115%);"LENGTH";
              TAB(122%);"CA";TAB(125%);"BB";
              TAB(128%);"TYPE"
\       PRINT #10%,STRING$(10%,45%);TAB(11%);"-----";
              TAB(17%);"---";TAB(21%);"------";
              TAB(28%);STRING$(9%,45%);TAB(38%);
              "-----";TAB(44%);STRING$(9%,45%);
              TAB(54%);"-----";TAB(60%);"------";
              TAB(67%);"---";TAB(71%);"--";
              TAB(74%);"----";TAB(79%);"------";
              TAB(86%);"----";TAB(91%);"-----";
              TAB(9/%);STRING$(8%,45%);TAB(106%);
              STRING$(6%,45%);TAB(114%);STRING$(7%,45%);
              TAB(122%);"--";TAB(125%);"--";
              TAB(128%);"----"
\       LO% = LO% + 5%
\       RETURN                                          !MORE HEADINGS

26000 !*************************************************
\      !*
\      !* COUNTS LINES ON PAGE AND DETERMINES PAGING.
\      !*
\      !*************************************************

26005   LO% = LO% + 1%
\       GOTO 26100 IF LO% < 60%
\       GOSUB 25000
\       GOSUB 25100                                     !NEW PAGE

26100   RETURN                                          !ALL DONE

32000 !*************************************************
\      !*
\      !* ERROR SECTION.
\      !*
\      !*************************************************

32005   IF ERR = 28 THEN 32767

32010   ON ERROR GOTO 0

32766 !*************************************************
\      !*
\      !* EOJ
\      !*
\      !*************************************************

32767   CLOSE #1%
\       CLOSE #2%
\       END
```

LAWS OF PROJECT MANAGEMENT

. . . from a friend at DECUS

- No major project is ever installed on time, within budget, with the same staff that started it.
- Projects progress quickly until they become 90% complete; then remain at 90% complete forever.
- If project content is allowed to change freely, the rate of change will exceed the rate of progress.
- No system is ever completely debugged; attempts to debug a system inevitably introduce new bugs that are even harder to find.
- Project teams detest progress reporting because it vividly demonstrates their lack of progress.



```
Account: [ 1 , 11 ]     System Administrator                                                FULL ACCOUNT REPORT

System Pack ID: SYS211        Date of last write                 Pack cluster size:  16

UFD cluster    CPU Time  KCT        Device    Quota      Connect
   -----       --------  --------   --------  --------   --------
    2278         42:51    304007      01:35   UNLIMIT      17:02
                                                                  * * * RECORD   MANAGEMENT    SERVICES * * * * *    * FLAGS *

                                    DATE LAST FILE  DATE OF   TIME                      FILE REC     REC  EOF   FIRST     BUCKET  MAX REC         CA
FILENAME   PROT  ACC STATUS         ACCESS SIZE  CREATION  CREAT RTS        CL  PC ORG  FORMAT SIZE BLOCK BYTE EOF  SIZE    LENGTH CA BB TYPE
---------- ----- --- ------      --------- ----- --------- ----- ------    --- -- ---- ------ ---- ----- --------- ------  ------- -- -- ----
MIKE  .OUT < 60>   0             21-Oct-81    22 21-Oct-81 13:32 WPSSPL  16
    DEVICE CLUSTERS:                4517     4548

INDEX .WPS < 60>   0             23-Oct-81     3 09-Oct-81 15:14 WORD11  16
    DEVICE CLUSTERS:                3320

WPS   .TSK < 60>   0             23-Oct-81     9 09-Oct-81 15:14 WORD11  16
    DEVICE CLUSTERS:                1906

WPS012.WPS < 60>   0             20-Oct-81     4 19-Oct-81 09:56 WORD11  16
    DEVICE CLUSTERS:                1610

WPS009.WPS < 60>   0             20-Oct-81    20 16-Oct-81 09:51 WORD11  16
    DEVICE CLUSTERS:                2934     3696

WPS016.WPS < 60>   0             20-Oct-81     3 20-Oct-81 08:14 WORD11  16
    DEVICE CLUSTERS:                3811

WPS011.WPS < 60>   0             20-Oct-81     6 20-Oct-81 13:30 WORD11  16
    DEVICE CLUSTERS:                3193

WPS002.WPS < 60>   0             20-Oct-81     5 20-Oct-81 14:27 WORD11  16
    DEVICE CLUSTERS:                4313

WPS003.WPS < 60>   0             20-Oct-81     3 20-Oct-81 14:37 WORD11  16
    DEVICE CLUSTERS:                4310

MAIL  .MAI < 60>   0             20-Oct-81     0 20-Oct-81 16:26 BASIC   16
    DEVICE CLUSTERS:

MESSAG.TXT < 60>   0             20-Oct-81     1 20-Oct-81 16:26 BASIC   16
    DEVICE CLUSTERS:                3828

WPS004.WPS < 60>   0             16-Oct-81     3 16-Oct-81 09:29 WORD11  16
    DEVICE CLUSTERS:                4559

WPS010.WPS < 60>   0             16-Oct-81     7 16-Oct-81 09:36 WORD11  16
    DEVICE CLUSTERS:                4/40

MAPUF1.BAS < 60>   0             14-Oct-81     3 14-Oct-81 14:54 TECO    16
    DEVICE CLUSTERS:                1236

MAPUF2.BAS < 60>   0             14-Oct-81     3 14-Oct-81 14:55 TECO    16
    DEVICE CLUSTERS:                2326

MAPUFD.BAS < 60>   0             14-Oct-81    12 14-Oct-81 16:10 TECO    16
    DEVICE CLUSTERS:                1974

MAPUF3.BAS < 60>   0             13-Oct-81     5 13-Oct-81 15:16 TECO    16
    DEVICE CLUSTERS:                1625

INQUI .TSK <124>   0 C           22-Oct-81   188 22-Oct-81 11:20 RSX     16    Sequ Fixed   512   189         0
    DEVICE CLUSTERS:               16821    16822      16823     16824     16825      16826      16827
                                   16828    16829      16830     16831     16832

WPS001.WPS < 60>   0             22-Oct-81     8 22-Oct-81 12:58 WORD11  16
    DEVICE CLUSTERS:                3341

WPS019.WPS < 60>   0             22-Oct-81     5 22-Oct-81 13:26 WORD11  16
    DEVICE CLUSTERS:                6582

WPS005.WPS < 60>   0             23-Oct-81    20 23-Oct-81 10:26 WORD11  16
    DEVICE CLUSTERS:                 972     3198

WPS020.WPS < 60>   0             23-Oct-81     7 23-Oct-81 11:23 WORD11  16
    DEVICE CLUSTERS:                2960

DAVE  .LET < 60>   0             23-Oct-81     6 23-Oct-81 11:25 WPSSPL  16
    DEVICE CLUSTERS:                4739

SAMPLE.DOC < 60>   0             23-Oct-81     1 23-Oct-81 11:30 BASIC   16
    DEVICE CLUSTERS:                1966
```

BIO.BAS

By Rob Frazer, Nationwide Data Dialog, Inc., Southampton, PA

```
1         EXTEND
10        ! BIO.BAS                                                   &
          !                                                           &
          !       BIORHYTHM GRAPH AND/OR COMPATABILITY PROGRAM        &
          !
20        !       The biorhythm theory is that from the day of one's  &
          !       birth, one's intellectual, emotional, and physical  &
          !       highs and lows fall into a 33-, 28-, and 23-day cycle &
          !       respectively.  A more positive reading indicates that &
          !       powers or sensitivities are at a peak, and a negative &
          !       reading implies the opposite; however, on days when the &
          !       neutral or zero level is intersecting, the powers are &
          !       unpredictable and can be suprisingly high or low.   &
          !
30        !       The compatibility reading is simply a measure of how &
          !       in-phase two individual's cycles are.               &
          !
40        !       In examining my own biorhythm after-the-fact when I &
          !       have had an extremely good or bad day, I have come to &
          !       the conclusion that, for me, the whole concept is one &
          !       big pile of FIRQB.  (But it was fun to code.)       &
          !
44        !       PROPRIETARY NOTICE:                                 &
          !       by:               Rob Frazer, Applications for      &
          !                         Nationwide Data Dialog, Inc.      &
          !                         70 James Way                      &
          !                         Southampton, PA 18966             &
          !                                                           &
          !       This software is furnished free (just what it's worth) &
          !       to all subscribers of the RSTS PROFESSIONAL.        &
          !       The author assumes no responsibility for any emotional &
          !       insecurities, bad-karmic loops, suicides, mass-murders, &
          !       Picasso's "Guernica", earthquakes, the Sex Pistols, or &
          !       any bad luck which may blah, blah, blah...          &
          !                                                           &
          !
60          Z = .1E39                       ! DEMAND SCALE FACTOR ZERO
90          GOSUB 19600                     ! NORMAL PGM SETUP        &
          !
92          DIM G.POS%(101%)                ! HORIZONTAL POSITION     &
          !
94          KB.BDT$ = '22-JUN-51'                                     &
          \ DF.DAYS% = 15%                                            &
          \ F.OUT$ = 'KB:'                                            &
          !
100         GOSUB 11000                             ! GET BIRTHDATE   &
          !
200         PRINT #KB%, 'Enter <G>raph, or <C>ompatibility.....';     &
          \ INPUT #KB%, Z$                                            &
          \ Z% = ASCII(CVT$$(Z$,-1%))                                 &
          \ GOTO 1000       IF Z% = 71%             ! G               &
          \ GOTO 2000       IF Z% = 67%             ! C               &
          \ PRINT #KB%, 'G or C, please...'                           &
          \ GOTO 200                                                  &
          !
1000      !       G R A P H                                           &
          !                                                           &
            GOSUB 11100                             ! GET GRAPH DAYS  &
          !
1020        GOSUB 11200                             ! GET O/P DEV     &
          !
1100        GOSUB 12000                             ! GRAPH HEADING   &
          !                                                           &
          \ FOR I.OLD% = D1.OLD% TO D2.OLD%                           &
          \  GOSUB 11300                            ! PRINT DAY OR DATE &
          \  GOSUB 11400                            ! COMPUTE, PRINT  &
          !
1140        NEXT I.OLD%                                               &
          !
1160        GOSUB 12200                             ! GRAPH TRAILER   &
          !                                                           &
          \ GOTO 100                                                  &
          !                                                           &
          !                                                           &
          !
2000      !       C O M P A T I B I L I T Y                           &
          !                                                           &
            GOSUB 11500                             ! OTHER'S BIRTHDAY &
          !
2020        GOSUB 11600                             ! FIND % OUT OF PHASE &
          !
2080        GOSUB 11700                             ! PRINT VALUES    &
          !
2090        PRINT #KB%, 'Anyone else ?';                              &
          \ INPUT #KB%, Z$                                            &
          \ GOTO 2000     IF ASCII(CVT$$(Z$,-1%)) = 89%   ! 'Y'       &
          \ GOTO 32767                                                &
          !                                                           &
          !                                                           &
          !
11000     !       GET OPERATOR'S BIRTHDATE                            &
          !                                                           &
            PRINT #KB%, 'Enter your birthdate in format DD-Mon-YY'    &
          \ PRINT #KB%, '<';KB.BDT$;'>.......................';      &
          \ INPUT #KB%, Z$                                            &
          \ KB.BDT$ = CVT$$(Z$,4%)          IF LEN(Z$)                &
          \ BDT.JUL$ = FNJUL$(KB.BDT$)              ! JULIAN FORMAT   &
          \ GOTO 11000     IF E%                                      &
          \ RETURN                                                    &
          !
11100     !       GET GRAPH DATES                                     &
          !                                                           &
            KB.D1$ = DATE$(0%)                                        &
          \ PRINT #KB%, 'Enter 1st day of graph <';KB.D1$;'> ?';      &
          \ INPUT #KB%, Z$                                            &
          \ KB.D1$ = CVT$$(Z$,4%) IF LEN(Z$)                          &
          \ D1.JUL$ = FNJUL$(KB.D1$)                                  &
          !                                                           &
          \ D1.OLD% = FNDAY.DIFF%(BDT.JUL$,D1.JUL$)       ! FIND DAYS OLD &
          \ PRINT #KB%, 'You are '; D1.OLD%; ' days old as of '; KB.D1$ &
          !                                                           &
          \ PRINT #KB%, 'Enter number of days to graph <';DF.DAYS%;'>?'; &
          \ INPUT #KB%, Z%                                            &
          \ Z% = DF.DAYS% UNLESS Z% > 0%                              &
          \ D2.OLD% = D1.OLD% + Z% - 1%                               &
          \ D1.D% = VAL(RIGHT(D1.JUL$,3%))          ! DAY#            &
          \ D1.Y% = VAL(LEFT(D1.JUL$,2%))                             &
          \ MAX.Y% = FND.Q%(D1.Y%)          ! DAYS IN THIS YEAR       &
          \ RETURN                                                    &
```

PROVERBS . . . from a friend at DECUS

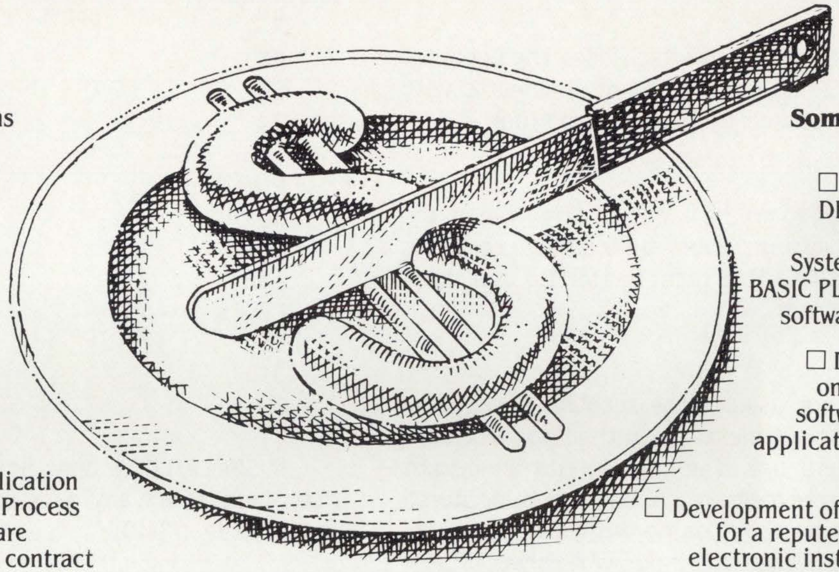
1. You cannot produce a baby in one month by impregnating nine women.
2. The same work under the same conditions will be estimated differently by ten different estimators or by one estimator at ten different times.
3. The most valuable and least used word in a project manager's vocabulary is "NO."
4. You can con a sucker into committing an unreasonable deadline, but you can't bully him into meeting it.
5. The more ridiculous the deadline, the more it costs to try to meet it.
6. The more desperate the situation, the more optimistic the situatee.
7. Too few people on a project can't solve the problems—too many create more problems than they solve.
8. You can freeze the user's specs but he won't stop expecting.
9. Frozen specs and the abominable snowman are alike: They are both myths and they both melt when sufficient heat is applied.
10. The conditions attached to a promise are forgotten and the promise is remembered.
11. What you don't know hurts you.
12. A user will tell you anything you ask about—nothing more.
13. Of several possible interpretations of a communication, the least convenient one is the only correct one.
14. What is not on paper has not been said.
15. Parkinson and Murphy are alive and well—in your project.

```
        !
11200   !       GET OPTIONAL OUTPUT DEVICE                                  &
        !                                                                   &
          PRINT #KB%, 'Output to <'; F.OUT$; '> ? ';                        &
        \ INPUT LINE #KB%, Z$                                               &
        \ Z$ = CVT$$(Z$,-1%)                                                &
        \ F.OUT$ = Z$    IF LEN(Z$)                                         &
        \ IF     F.OUT$ = 'KB:'                                             &
          THEN   F.OUT% = KB%                                               &
        \        GOTO 11280                                                 &
        !
11220     F.OUT% = 2%                                                       &
        \ OPEN F.OUT$ FOR OUTPUT AS FILE F.OUT%                             &
        !
11280     RETURN                                                            &
        !
11300   !       PRINT THE DAY, OR THE WHOLE DATE IF CRITICAL                &
        !                                                                   &
           D% = D1.D% + I.OLD% - D1.OLD%                                    &
        \  D% = 1000% + D% - MAX.Y%      IF D% > MAX.Y%  ! YEAR ROLLOVER &
        \  D$ = DATE$((D1.Y%-70%)*1000% + D%)                               &
        \  IF    I.OLD% = D1.OLD%        OR                                 &
                 LEFT(D$,2%) = '01'              ! PRINT THE WHOLE DATE  &
           THEN PRINT #F.OUT%, D$;                       ! ONLY IF IMPORTANT &
           ELSE PRINT #F.OUT%, LEFT(D$,2%);                                 &
        !
11390     RETURN                                                            &
        !
11400   !       COMPUTE FUNCTION VALUES                                     &
        !                                                                   &
          I.P = FNSIN(I.OLD%*2.*PI/23.)          ! PHYSICAL                 &
        \ I.E = FNSIN(I.OLD%*2.*PI/28.)          ! EMOTIONAL                &
        \ I.I = FNSIN(I.OLD%*2.*PI/33.)          ! INTELLECTUAL             &
        !                                                                   &
        \ G.POS%(I%) = 32%       FOR I% = 1% TO 51%                         &
        \ G.POS%(26%) = ASCII(' ')                                          &
        \ G.POS%( 0%) = 51%                                                 &
        \ G.POS%(26.5+25%*I.P) = ASCII('P')                                 &
        \ G.POS%(26.5+25%*I.E) = ASCII('E')                                 &
        \ G.POS%(26.5+25%*I.I) = ASCII('I')                                 &
        \ CHANGE G.POS% TO D$                                               &
        \ PRINT #F.OUT%, TAB(12%); D$            ! PRINT ONE LINE           &
        \ RETURN                                                            &
        !
11500   !       FIND THE OTHER BIRTHDATE                                    &
        !                                                                   &
          PRINT #KB%, "Enter your potential partner's birthdate"            &
        \ PRINT #KB%, ' in format DD-Mon-YY..................';            &
        \ INPUT #KB%, Z$                                                    &
        \ PT.BDT$ = CVT$$(Z$,4%)                                            &
        \ PT.JUL$ = FNJUL$(PT.BDT$)                                         &
        \ GOTO 11520     IF PT.JUL$ < BDT.JUL$                              &
        !                                                                   &
        \ DAY.DIF% = FNDAY.DIFF%(BDT.JUL$,PT.JUL$)                          &
        \ PRINT #KB%, 'You are';DAY.DIF%;'days older';                      &
        \ GOTO 11590                                                        &
        !
11520     DAY.DIF% = FNDAY.DIFF%(PT.JUL$,BDT.JUL$)                          &
        \ PRINT #KB%, 'You are';DAY.DIF%;'days younger';                    &
```

```
        !
11590     RETURN                                                            &
        !
11600   !       FIND THE THREE PHASE PERCENTAGES                            &
        !                                                                   &
          X = DAY.DIF%                                                      &
        \ C.P = 23. * (X/23. - FIX(X/23.))       ! HOW FAR OUT OF PHASE     &
        \ C.E = 28. * (X/28. - FIX(X/28.))                                  &
        \ C.I = 33. * (X/33. - FIX(X/33.))                                  &
        !                                                                   &
        \ C.P = 23. - C.P        IF C.P > 23./2. ! ROUND TO FORWARD         &
        \ C.E = 28. - C.E        IF C.E > 28./2. ! OUT OF PHASE             &
        \ C.I = 33. - C.I        IF C.I > 33./2.                            &
        !                                                                   &
        \ P.P = 100. * (1. - C.P/(23./2.))       ! PERCENTAGE OF            &
        \ P.E = 100. * (1. - C.E/(28./2.))       ! HALF-CYCLE OUT           &
        \ P.I = 100. * (1. - C.I/(33./2.))                                  &
        \ RETURN                                                            &
        !
11700   !       DISPLAY CALCULATED PERCENTAGES                              &
        !                                                                   &
          PRINT #KB%, 'The compatibility percentages are:'                  &
        \ PRINT #KB%, 'Physical          ';                                 &
        \ PRINT #KB% USING '###.##', P.P;                                   &
        \ PRINT #KB%, '%'                                                   &
        \ PRINT #KB%, 'Emotional         ';                                 &
        \ PRINT #KB% USING '###.##', P.E;                                   &
        \ PRINT #KB%, '%'                                                   &
        \ PRINT #KB%, 'Intellectual      ';                                 &
        \ PRINT #KB% USING '###.##', P.I;                                   &
        \ PRINT #KB%, '%'                                                   &
        \ PRINT #KB%                                                        &
        \ PRINT #KB%, 'Average           ';                                 &
        \ PRINT #KB% USING '###.##', (P.P+P.F+P.I)/3.;                      &
        \ PRINT #KB%, '%'                                                   &
        \ PRINT #KB%     FOR I% = 1% TO 3%                                  &
        \ RETURN                                                            &
        !
12000   !       HEADING FOR BIOGRAPH                                        &
        !                                                                   &
          PRINT #F.OUT% FOR I% = 1% TO 4%                                   &
        \ PRINT #F.OUT%, TAB(30%); 'Biorhytnm for ';KB.BDT$                 &
        \ PRINT #F.OUT%                                                     &
        \ PRINT #F.OUT%, '  Date '; TAB(12%);                               &
                '-....:....:....:....:....0....:....:....:....:....+'   &
        \ RETURN                                                            &
        !
12200   !       GRAPH TRAILER                                               &
        !                                                                   &
          PRINT #F.OUT%, TAB(12%);                                          &
                '-....:....:....:....:....0....:....:....:....:....+'   &
        \ PRINT #F.OUT% FOR I% = 1% TO 6%                                   &
        \ RETURN                                                            &
        !
13000   !·      FNDAY.DIFF%     DERIVE DIFFERENCE IN DAYS                   &
        !                                                                   &
        DEF FNDAY.DIFF%(X$,Y$)                                              &
        \ Y% = VAL(LEFT(X$,2%))                  ! 1ST YEAR                 &
        \ S% = FND.Q%(Y%) - VAL(RIGHT(X$,3%))    ! PORTION                  &
        !
13004     Y% = Y% + 1%                                                      &
        \ GOTO 13008     IF Y% > VAL(LEFT(Y$,2%))            ! MORE TO COME  &
        \ S% = S% + FND.Q%(Y%)                               ! DAYS PER YEAR &
        \ GOTO 13004                                                        &
        !
13008     S% = S% - FND.Q%(Y%-1%)                            ! NOT WHOLE YEAR &
        \ S% = S% + VAL(RIGHT(Y$,3%))                        ! PORTION       &
        \ FNDAY.DIFF% = S%                                                  &
        \ FNEND                                                             &
        !
13100   !       FND.Q% = NUMBER OF DAYS IN YEAR Y%                          &
        !                                                                   &
        DEF FND.Q%(Y%) = 365% - (Y% > 0%                     ! NON-CENTURY   &
                         AND Y% = 4% * (Y%/4%))              ! LEAP YEAR     &
        !
13400   !>      FNJUL$          DERIVE JULIAN DATE "YYDDD"                  &
        !                                                                   &
        DEF FNJUL$(D$)                                                      &
        \ M$ = CVT$$(MID(D$,4%,3%),32%)          ! UPPER CASE               &
        \ M% = INSTR(1%,                                                    &
                'XXJANFEBMARAPRMAYJUNJULAUGSEPOCTNOVDEC',M$) / 3%           &
        \ E% = (M% = 0%)                                                    &
        \ D% = VAL(MID('XX000031059090120151181212243273304334',            &
                         3%*M%,3%))              ! ODD DAYS FROM PRIOR MONTH &
        \ Y% = VAL(RIGHT(D$,8%))                                            &
        \ D% = D% + 1%  IF Y% = 4% * (Y%/4%)                 ! LEAP YEAR     &
                        IF M% > 2%                           ! AFTER FEB.    &
        \ Z$ = NUM1$(1000% + VAL(LEFT(D$,2%)) + D%)                         &
        \ FNJUL$ = RIGHT(D$,8%) + RIGHT(Z$,2%)                              &
        \ FNEND                                                             &
        !
19600   !> ------ S T A N D A R D   I N I T I A L I Z A T I O N ------     &
          ON ERROR GOTO 19800                                               &
        \ KB%=12%                                                           &
        \ OPEN 'KB:' AS FILE KB%, MODE 4%                                   &
        \ RETURN                                                            &
        !
19800   !> ------ L O C A L   E.R R O R   T R A P S -----------------      &
          E%=ERR                                                            &
        !
19990     GOTO 32000                                                        &
        !
20000   !> ------ S T A N D A R D   R O U T I N E S -----------------      &
        !
27950   DEF FNSIN(X)                                                        &
        \ X = X - (2 * PI * FIX(X/(2*PI)))       ! SUBTRACT 2PI'S           &
        \ SUM, TRM = X                                                      &
        \ FOR N% = 2% TO 999%                                               &
        \       GOTO 27954       IF ABS(TRM) < 0.00001   ! PRECISION        &
        \       Y% = 2%*N% - 1%                                             &
        \       Y = -(X*X)/(Y%*(Y%-1%)) ! TERM MULTIPLIER                   &
        \       TRM = TRM * Y                                               &
        \       SUM = SUM + TRM                                             &
        \ NEXT N%                                                           &
        !
27954     SUM = SGN(SUM)         IF ABS(SUM) > 1.                           &
        \ FNSIN = SUM                                                       &
        \ FNEND                                                             &
        !
32000   !> ------ S T A N D A R D   E R R O R   T R A P S -----------      &
        !
32500     ON ERROR GOTO 0                                                   &
        \ STOP                                                              &
        !
32560     DEF FNE$(E%) = CVT$$(RIGHT(SYS(CHR$(6%)                           &
                          +CHR$(9%)+CHR$(E%)),3%),5%)                       &
        !
32767     END
```

♥

Hinditron Offshore Software Capability. It Costs About 50% Less.



Hinditron Computer Systems and Consultants has the offshore software facility to help you get more from your computer for less.

Hinditron's software development centre in Bombay, India, is geared to meet your specific requirements, using a wide range of computer and microprocessor systems.

Hinditron offers software services for developing Application Software, System Software, Process Control Software and Software Conversions on turnkey and contract basis.

Hinditron offers all this for almost half the prevailing rate.

The Company

Hinditron has exclusive representation rights for DEC, Tektronix, Applicon, FPS and several other internationally reputed companies in India. The company provides TOTAL technical support, training and consultancy services to its clients.

Hinditron has a multi-disciplinary team of over 200 professionals including software engineers, business applications specialists and hardware engineers.

Hinditron's software development centre is equipped with a PDP-11/34 System and will shortly acquire a VAX 11/780 System.

Hinditron has developed sophisticated software products and carried out software projects in U.S.A.; U.K.: Australia. New Zealand and the Middle-East, South-East Asian and African regions. DEC and a number of DEC OEMS have availed of these services.

Some of Our Current Projects:

□ Development of a DBMS based on-line Accounts Payable System under RSTS/E - BASIC PLUS II for a leading software house in U.S.A.

□ Development of an on-line DBMS-based software for insurance applications for a leading DEC OEM in USA.

□ Development of PASCAL Compiler for a reputed manufacturer of electronic instruments in U.S.A.

□ Development of an I/O Handler in MACRO-11.

□ Conversion of software packages in FORTRAN from RSX-11M to IAS Operating System for a leading DEC user.

Our Clients Know The System Works

Our national and international clients have put Hinditron's offshore capabilities to work in everything from batch to on-line DBMS, from business applications to engineering design and graphics to hardware design.

And they all know IT WORKS.

For more information on how you can join us offshore, return the coupon below or call Red Carlson at (206) 486-6555.

HINDITRON

Hinditron Computer Systems and Consultants
6450 NE 183, Seattle, WA 98155

□ Please send me information on how I can save up to 50% on my computer software.

□ Please have a representative call.

Name ——————————

Title —————— Phone ——————

Company ——————————

Address ——————————

——————————

City ——————————

State —————— Zip ——————

HINDITRON COMPUTER SYSTEMS &
CONSULTANTS PVT. LTD.
Eros Building,
42, Maharshi Karve Road,
Bombay 400 020. India
Telex : 011-2326 India.

DEC, RSTS/E, RSX-IIM, IAS are the registered trademarks of Digital Equipment Corporation, U.S.A.

CIRCLE 98 ON READER CARD

THE LOW SPEED SPOOLING PACKAGE

By M. H. Koplitz,

1.0 INTRODUCTION

The low speed spooling package will enable the user to spool out reports to terminals without the need of the RSTS spooling package. There are three programs involved in the package. They are SPOLER.BAS, SPOL1.BAS and CHAR.BAS.

1.1 SPOLER.BAS

SPOLER.BAS is the program which does the spooling. The program can either be run directly or chained to with the critical data being passed to it by core common.

1.2 SPOL1.BAS

SPOL1.BAS is used to create a virtual array file of the accounts on the RSTS system. Every time a new account is added to the system SPOL1.BAS must be run. No user input is needed for this program.

1.3 CHAR.BAS

This program is not used in the spooling package. It allows the user to print out messages in the large block letters that RSTS spoolers use. The user inputs an output device for the block letter messages to be printed, or stored (on disk). Then he inputs the lines he wishes to have converted to block letters. When the input is finished <CR> will end the program. The output can be queued to a printer if so desired.

INSTALLATION OF THE PACKAGE

2.0 How to Install

Log into a privileged account. If the package is on magnetic tape medium, copy it to disk. Three files exist in the package. Make sure that you are in the BASIC-PLUS runtime system. Then old each program and compile them in [1,0] with a protection code of 232. The three files are SPOLER.BAS, SPOL1.BAS, CHAR.BAS. Then run [1,0]SPOL1.

SPOL1.BAS will create one of the data files needed by SPOLER.BAS. A second file is optional and can be created by any editor. This file is SPOLER.DAT and MUST reside in [1,0].

2.1 Contents of SPOLER.DAT

SPOLER.DAT contains information about alias names that you have set up for terminals. If you wish to have KB0: known to SPOLER as "CONSOLE" then an entry in SPOLER.DAT for "CONSOLE" must be set up. For each alias SPOLER.DAT will contain the following attributes:

1) Alias name
2) Number of lines per form
3) KB number of alias
4) Width of form.

Each entry is followed by a carriage return.

NOTE: If a terminal is busy for an alias the requested spooling can be sent to the RSTS spooling system with the form name equal to the alias. Therefore it is good policy to give alias names to terminals which will at some time of the day have a RSTS spooler running on that form name.

The first value of SPOLER.DAT is a number informing SPOLER how many entries there are in the file. Example of a SPOLER.DAT:

```
3
PC
68
24
132
PURCH
68
25
132
PINES
50
31
80
```

If SPOLER.DAT does not exist then SPOLER assumes that there aren't any alias names. Note that SPOLER.DAT must reside in [1,0]!

SPOLER.BAS

3.0 OVERVIEW

SPOLER.BAS can be run in two ways:

1) Run directly (RUN [1,0]SPOLER),
2) Or chained to (CHAIN "[1,0]SPOLER" 500) with commands in core common.

NOTE: When run directly SPOLER will ask for all the information it needs to do its job.

SPOLER does the following functions (listed in chronological order):

1) Gets information about terminal name, form sizes, width, etc.
2) Determine account using SPOLER.BAS.
3) Determine whether job is on pseudo keyboard or not.
4) If alias give translate it to terminal number.
5) Analyze file name for account number.
6) Analyze file name for device number.
7) Analyze file name for body and extension.
8) Do a directory lookup of file name.
9) Convert terminal name to terminal number.
10) Assign terminal.
11) Do printing.

3.1 RUNNING SPOLER DIRECT

Use the RUN command to invoke SPOLER.BAC out of account [1,0]. The program banner will be displayed followed by several questions.

All the financial questions recorded in the Minutes, answered in seconds.

If you ever wished you had inside information on what would happen under certain circumstances, you are a prime candidate for the FPS–80 service from RTZ Timesharing.

With FPS–80 you can have any information about financial planning in seconds – as an instant display on your own VDU or as a permanent printed record.

When you subscribe to FPS–80, you are able to compile a store of information on your company on our computers. You will be linked directly to our service, and by dialling a special number on your own telephone, you are immediately connected and free to key in any questions you might have that could affect your company's financial stability.

On the other hand, if you should wish to work from your own computer, no matter what size, we may still be able to help – just ask for further details.

The FPS–80 system is highly flexible and easy to use – even for novices. It's backed by the technical knowledge, financial expertise and excellent service for which RTZCS is renowned and what's more, is surprisingly inexpensive.

If you have any questions about how to ask questions about the future, fill in the coupon today.

Please send me further details on FPS-80

Name ____________________

Company ____________________

Address ____________________

____

____

Tel No ____________________

RTZ
Timesharing

Whenever it's a question of money.

Post to: David Holroyd, RTZ Computer Services Ltd.,
103 Jermyn Street, London SW1Y 6EB. Tel: 01-930 4163.

M

CIRCLE 99 ON READER CARD

```
RUN [1,0]SPOLER
SPOLER V1.0 Installation name Spool files to terminals

File name? XXXXXX

Terminal to spool to? XXXXX or KB??:

Form width? XX (only if KB??: answered above)

Form size? XX(only if KB??: answered above)

Number of copies? XX
```

There are two switches that are available when answering the above questions.

They are "/D" on the filename indicating that the file is to be deleted after it is spooled, and "/DET" on the terminal name (or KB??:) which will cause SPOLER to detach and print "SPOLER is detaching in job slot ??".

If SPOLER runs detached it will log itself off when done. NOTE: Filename must be in the following format:

```
[xxx,yyy]Device:Filename.ext
If the account is left out the current account is assumed.
If the Device: is left out then SY0: is used
```

3.2 CHAINING TO SPOLER

The information needed for SPOLER must be placed in core common in the following order and in the following format. (Filename);(Terminal name);(Program to chain to);(line number to goto in chained program);(number of copies)

The semicolons are used as delimiters, the parens are not used. An example of the command is:

```
SPOLER.BAS;PURCH;[1,51]ZZM;1;5
```

SPOLER will print five copies of SPOLER.BAS to the PURCH terminal then chain to [1,51]ZZM at line 1. To delete a file after spooling place the "/D" switch described in 3.1 on the filename option.

KB??: can be substituted in place of the terminal name. If this is done the questions,

```
Form size of KB? XX
Form width? XX
```

will appear. Note that the "/DET" switch does not work when chaining to SPOLER.3-2 Chapter 3 SPOLER.BAS

SPOLER must be chained to at line 500.

3.3 DETACHED ERRORS

If the terminal specified is not available during spooling in the detached state the filename given is queued to the lp: with the form name equal to the terminal name. If a terminal number is given and it is not available then an error is generated to OPSER and the SPOLER terminates.

If any other error occurs OPSER is sent a message about the error and SPOLER terminates.

3.4 ATTACHED ERROR ON TERMINAL AVAILABILITY

The only recoverable (non-fatal) error is when the terminal requested for is not available. SPOLER then gives the following options:

```
%Terminal not available for printing
```

1. Wait (10 times then queued up)
2. Queue to spooler (report available ????????????)
3. Queue to high speed printer

Option 2 is not given when the "hard" terminal number was given. The ???? are replaced by the following depending on the time of day:

a. after a few minutes - between 16:00 and 10:00, also between 12:00 and 13:00

b. after 14:00 - between 13:00 and 14:00

c. after 12:00 - between 10:00 and 12:00

These values can be changed by reviewing line 32510 of SPOLER.BAS. If the choice is option 1 then SPOLER will attempt up to ten times to get the terminal. SPOLER will print a message after every attempt. After ten attempts it prints out "?Doing queing" and queues up the request.

3.5 ERRORS THAT CAN OCCUR

?Core common error	The core common string was missing some element or delimiter.
?Illegal account specs	The account is not in [xxx,yyy] format.
?Protection violation	A non-privileged account attempted to print a file not in his account.
?Illegal device specs	The device is illegal.
?No file ext. given	A file extension was not specified.
?Can not find terminal name	The terminal name given is not in SPOLER.DAT.
?Bad line number passed	The line number field in the core common is not an integer.
?Illegal account number	The account number is not a legal RSTS account.
?Illegal terminal # specified in KB??	The ?? is not a legal number.
?Can not find file or account	The file specified to print can not be found.
?Error to output terminal	A error occurred when attempting to send a line out output to the terminal. Could be a hung device?
?↑C trap	A ↑C was typed.
?Chaining to prog. not found	The program to chain to was not found.
?Illegal switch on filename	A switch other than /DET was given.
?Deletion of file in error	For some reason the file deletion could not occur.
?Unknown error at line ??	Some error occurred that is not taken care error: ?? of by a standard error.

SPOL1.BAS and CHAR.BAS

4.0 SPOL1.BAS

SPOL1 creates a virtual array file in account [1,0] that is needed by SPOLER to print the names associated with the account numbers. There is a limit in SPOL1 of 150 accounts. This can be increased by adjusting the DIM statements in both SPOL1 and SPOLER. When new accounts are added run SPOL1 to add the new entry to the virtual array. Note that SPOL1 uses $ACCT.SYS to create the virtual array so $ACCT.SYS must be kept up to date.

4.1 CHAR.BAS

CHAR is a small utility which will print any message in the block letters to any output device. It also resides in [1,0].

```
                            [11,11] SPOLER.BAS
001     EXTEND
010 !*****************************************************************
\     !*
\     !* THIS PROGRAM WAS WRITTEN BY M H KOPLITZ, SYSTEM ADMINISTRATOR,
\     !*
\     !*  05-AUG-81.
\     !*
\     !* THIS PROGRAM WILL SPOOL FILES TO LOW SPEED PRINTERS(TERMINALS).
\     !*  A TABLE EXISTS IN THE PROGRAM WHICH WILL IDENTIFY THE
\     !*  DIFFERENT PRINTERS. THIS TABLE FILL BE FILLED IN BY THE
\     !*  SYSTEMS MANAGER AND IS UPDATED AFTER EVERY HARD COPY TERMINAL
\     !*  CHANGE.
\     !*
\     !* THE PROGRAM CAN BE ENTERED IN TWO WAYS. 1) IS BY CHAINING TO
\     !*  LINE 500, IN CORE COMMON MUST BE THE FOLLOWING INFO;
\     !*  A) FILE TO SPOOL WITH ACCOUNT (IF NO ACCOUNT ONE IN ASSUMED),
\     !*  B) TERMINAL TO SPOOL TO.
\     !*  C) PROGRAM TO CHAIN TO.
\     !*  D) LINE NUMBER TO CHAIN TO.
\     !*  ALL DATA WILL BE SEPARATED BY SEMICOLONS.
\     !*
\     !*  2) RUNNING THE PROGRAM DIRECTLY WILL CAUSE PROMPTS TO COME
\     !*          UP FOR THE ABOVE INFORMATION
\     !*
\     !****************************************************************
\     !*
\     !*  REVISION #1 - M H KOPLITZ    OCTOBER 2, 1981
\     !*
\     !*      PURPOSE: WHEN KB: IS ENTERED AS THE TERMINAL TO SPOOL TO
\     !*               KB:, THE REPORTS GOES TO KB0:. CORRECT THIS
\     !*               PROBLEM BY LINE 1025 AND CHANGES TO 1030.
\     !*
\     !****************************************************************

100     PRINT "SPOLER   V1.1                    ";
              "    Spool files to terminals"
\       ON ERROR GOTO 25000
\       X$ = SYS(CHR$(6%)+CHR$(-7%))
\       ATTACHED% = 1%
\       PRINT                                               !PRINT BANNER, SET
                                                            ! FLAGGING AND ^C
                                                            ! TRAP.

110     PRINT "File name";
\       INPUT LINE FILE.NAME$
\       FILE.NAME$ = CVT$$(FILE.NAME$,4%)
\       GOTO 32767 IF FILE.NAME$ = "\"
\       INPUT "Terminal to spool to";TERMINAL.NAME$
\       GOTO 110 IF TERMINAL.NAME$ = "\"
\       GOTO 120 IF LEFT(TERMINAL.NAME$,2%) <> "KB"
\       INPUT "Form width";KB.WIDTH%
\       INPUT "Form size";TERMINAL.SIZE%                    !TERMINAL NAME AND
                                                            ! FILE NAME TO USE.
\!*******************************************************
\!*
\!* FILE NAME MUST BE IN FOLLOWING ORDER:
\!*
\!*      [XXX,YYY]DEVICE:FILENAME.EXTENSION
\!*
\!* ACCOUNT AND/OR DEVICE CAN BE LEFT OUT!
\!*
\!* ACCOUNT DEFAULTS TO SY0:, ACCOUNT TO CURRENT.
\!*******************************************************

120     INPUT "Number of copies";Q$
\       GOTO 110 IF Q$ = "\"
\       C9% = VAL(Q$)
\       T% = INSTR(1%,TERMINAL.NAME$,"/")
\       GOTO 140 IF T% = 0%
\       X$ = TERMINAL.NAME$
\       TERMINAL.NAME$ = LEFT(TERMINAL.NAME$,T%-1%)
\       INSTRUCTION$ = MID(X$,T% + 1%,LEN(X$) - T%)
\       GOTO 130 IF ASCII(INSTRUCTION$) = ASCII("D")
\       PRINT "?Illegal switch"
\       GOTO 110                                            !SEE IF THE DETACH
                                                            ! SWITCH WAS PUT ON
                                                            ! THE TERMINAL SPEC.
                                                            ! ALSO GET NUMBER
                                                            ! COPIES.
                                                            ! IF SO CHECK LEGALITY
                                                            ! ELSE ERROR OUT.

130     PRINT "SPOLER is detaching in job slot ";
              PEEK(518%)/2%
\       X$ = SYS(CHR$(6%)+CHR$(7%))                         !DO THE DETACH HERE


140     GOTO 600                                            !GOTO PROCESSING
                                                            ! SECTION.


500     ON ERROR GOTO 25000
\       X$ = SYS(CHR$(6%)+CHR$(-7%))
\       X$ = SYS(CHR$(7%)) + ";"
\       CHAINED.TO% = 1%
\       ATTACHED% = 1%                                      !GET CORE COMMON
                                                            ! STRING, SINCE WE
                                                            ! ENTERED VIA CHAIN
                                                            ! STATEMENT.
\!*******************************************************
\!*
\!* AT LINE 500 IS WHERE ONE WOULD CHAIN TO FROM ANY
\!*      PROGRAM. CORE COMMON NEEDS THE FOLLOWING
\!*      SPECS:
\!*
\!*      FILE NAME DELIMITED BY ;
\!*      TERMINAL NAME (COULD BE KB??) DELIMETED BY ;
\!*      PROGRAM TO CHAIN TO DELIMETED BY ;
\!*      LINE NUMBER IN PROGRAM TO CHAIN TO DELIMITED
\!*             BY ;
\!*      NUMBER COPIES
\!*
\!*******************************************************
510     T% = INSTR(1%,X$,";")
\       GOTO 550 IF T% = 0%
\       FILE.NAME$ = LEFT(X$,T% - 1%)
\       T1% = INSTR(T% + 1%,X$,";")
\       GOTO 550 IF T1% = 0%
\       TERMINAL.NAME$ = MID(X$,T% + 1%,T1% - T%-1%)
\       T2% = INSTR(T1% + 1%,X$,";")
\       GOTO 550 IF T2% = 0%
\       PROGRAM.TO$ = MID(X$,T1% + 1%,T2% - T1%-1%)
\       T3% = INSTR(T2% + 1%,X$,";")
\       GOTO 550 IF T3% = 0%
\       LINE.NUMBER% = VAL(MID(X$,T2% + 1%,T3% - T2%-1%))
\       T4% = INSTR(T3% + 1%,X$,";")
\       GOTO 550 IF T4% = 0%
\       C9% = VAL(MID(X$,T3% + 1%, T4% - T3%-1%))
```

```
\       GOTO 575                                            !TAKE THE CORE COMMON
                                                            ! STRING AND BREAK
                                                            ! IT UP INTO THE 4
                                                            ! COMPONENTS. GOTO
                                                            ! 550 IF ERROR
                                                            ! OCCURRS.

550     ERROR.FLAG% = 1%                                    !SIMPLY SET ERROR
                                                            ! FLAG TO 1% THIS
                                                            ! MEANS THAT SOME
                                                            ! COMPONENT OF THE
                                                            ! CORE COMMON IS
                                                            ! MISSING.


575     INPUT "Form size of KB";TERMINAL.SIZE%
                IF LEFT(TERMINAL.NAME$,2%) = "KB"
\       INPUT "Form width";KB.WIDTH%
                IF LEFT(TERMINAL.NAME$,2%) = "KB"           !SINCE KB SPECS
                                                            ! ON CHAIN GET THE
                                                            ! FORM SIZE.


600     A% = PEEK(PEEK(PEEK(520%) + 8%) + 24%)
\       PROJ.NUMBER% = SWAP%(A%) AND 255%
\       PROG.NUMBER% = A% AND 255%                          !GET THE PROGRAMMER
                                                            ! AND PROJECT NUMBER.

610     IF ((PEEK(PEEK(PEEK(PEEK(520%)))+2%)
                AND 255%) = (PEEK(518%) AND 255%)
          AND
                (PEEK(PEEK(PEEK(PEEK(520%)))+6%)
                AND 8192%) = 8192%)
          THEN
                ATTACHED% = 1%
          ELSE
                ATTACHED% = 0%                              !DETERMINE WHETHER
                                                            ! WE ARE ATTACHED OR
                                                            ! DETACHED. PG 7-150
                                                            ! PROGRAMMERS MAN.

620     X$ = SYS(CHR$(6%)+CHR$(26%)+CHR$(0%)+CHR$(0%))
\       Y$ = MID(X$,5%,1%)
\       PSEUDO.NUMBER% = ASCII(Y$)                          !ARE WE ON A PSEUDO
                                                            ! KEYBOARD? PSEUDO
                                                            ! .NUMBER% = 0%
                                                            ! MEANS WE AREN'T

630     GOTO 32000 IF ERROR.FLAG%                           !GOTO ERROR HANDLING
                                                            ! SECTION IF ERROR
                                                            ! .FLAG% =1%

640     OPEN "[1,0]SPOLER.DAT" FOR INPUT AS FILE #10%
\       INPUT #10%,KB.TABLE%(0%,0%)
\       FOR X% = 1% TO KB.TABLE%(0%,0%)
\               INPUT #10%,KB.TABLE$(X%)
\               INPUT #10%,KB.TABLE%(X%,1%)
\               INPUT #10%,KB.TABLE%(X%,2%)
\               INPUT #10%,KB.TABLE%(X%,3%)
\       NEXT X%
\       CLOSE #10%
                                                            !TERMINAL TABLE,
                                                            ! ZERO ELEMENT OF
                                                            ! INTEGER TABLE IS #
                                                            ! ENTRIES. INT. TABLE
                                                            ! IS NUMBER LINE PER
                                                            ! PAGE ON PRINTER. THE
                                                            ! SECOND VALUE IS THE
                                                            ! TERMINAL NUMBER.
\!*******************************************************
\!*
\!* THIS FILE IS UNDER THE NAME [1,0]SPOLER.DAT.
\!*      IT CONTAINS THE NAMES OF THE KBs AND SPECS ABOUT
\!*      EACH. THE ORDER IS:
\!*             ITEM ONE MUST BE NUMBER TERMINALS IN
\!*                             FILE
\!*             NAME OF TERMINAL(ALIAS)
\!*             # LINES/FORM
\!*             KB NUMBER
\!*             WIDTH OF PRINTER
\!*
\!*      FOR EACH TERMINAL.
\!*
\!*******************************************************


700 !*******************************************************
\     !*
\     !* INITIALIZATION COMPLETE, WE HAVE ALSO DETERMINED
\     !*  OUR STATUS ON THE COMPUTER, TO WHETHER WE ARE
\     !*  DETACHED OR ATTACHED OR WHATEVER.
\     !*  NOW LET US CHECK TO SEE IF WE HAVE ACCESS TO THE
\     !*  FILE WE WISH TO PRINT.
\     !*
\     !*****************************************************

710     T$ = CVT$$(FILE.NAME$,64%)
\       T% = INSTR(1%,T$,")")
\       GOTO 800 IF T% = 0%                                 !CONVERT ALL [] TO
                                                            ! (). THEN SEE IF
                                                            ! ANY EXIST. IF NONE
                                                            ! THEN ASSUME CURRENT
                                                            ! ACCOUNT. ALLOW PRIV
                                                            ! ACCOUNT ACCESS TO
                                                            ! ALL FILES.

720     FILE.NAME$ = RIGHT(FILE.NAME$,T% + 1%)
\       ACCOUNT$ = LEFT(T$,T% - 1%)
\       T% = INSTR(1%,ACCOUNT$,",")
\       GOTO 780 IF T% = 0%
                OR ASCII(ACCOUNT$) <> ASCII("(")
\       FILE.PROJ% = VAL(MID(ACCOUNT$,2%,T% - 2%))
\       FILE.PROG% =
            VAL(MID(ACCOUNT$,T%+1%,LEN(ACCOUNT$) - T%))
\       ACCOUNT$ = RIGHT(ACCOUNT$,2%)                       !GET THE PROGRAMMER
                                                            ! AND PROJECT NUMBER
                                                            ! OF THE FILE TO SPOOL
                                                            ! ALSO STRIP THE
                                                            ! ACCOUNT NUMBER FROM
                                                            ! THE FILE NAME
730     GOTO 810 IF PROJ.NUMBER% = 1%
\       GOTO 790 IF FILE.PROJ% <> PROJ.NUMBER%
\       GOTO 790 IF FILE.PROG% <> PROG.NUMBER%              !ERROR OUT IF THE ACCT
                                                            ! NUMBERS DO NOT
                                                            ! MATCH.

740     GOTO 810                                            !FILE OK GO TO NEXT
                                                            ! PART

780     ERROR.FLAG% = 2%
```

```
\       GOTO 32000                                      !ERROR DUE TO ILLEGAL
                                                        ! ACCOUNT SPECS. GOTO
                                                        ! ERROR SECTION.

790     ERROR.FLAG% = 3%
\       GOTO 32000                                      !ERROR DUE TO ACCT
                                                        ! ATTEMPTS TO GET
                                                        ! FILE IT CAN'T HAVE.
                                                        ! GOTO ERROR SECTION.

800     FILE.PROJ% = PROJ.NUMBER%
\       FILE.PROG% = PROG.NUMBER%
\       ACCOUNT$ = NUM1$(FILE.PROJ%) + ","
                + NUM1$(FILE.PROG%)                     !SINCE NO ACCOUNT WAS
                                                        ! GIVEN, USE THE
                                                        ! CURRENT ACCOUNT.

810 !****************************************************
\   !*
\   !* NOW THAT WE HAVE THE ACCOUNT, GET THE DEVICE
\   !*  WHERE THE FILE IS SITTING.
\   !*
\   !****************************************************

820     T% = INSTR(1%,FILE.NAME$,":")
\       GOTO 880 IF T% = 0%
\       DEVICE$ = LEFT(FILE.NAME$,T%-1%)
\       LATER.DEVICE$ = DEVICE$
\       FILE.NAME$ = RIGHT(FILE.NAME$,T% + 1%)
\       GOTO 850 IF LEFT(DEVICE$,2%) = "DB"
\       X$ = SYS(CHR$(6%)+CHR$(-10%)+DEVICE$+":")
\       DEVICE$ = MID(X$,23%,3%)
\       GOTO 900                                        !GET THE DEVICE NAME
                                                        ! AND STRIP IT FROM
                                                        ! THE FILENAME.

850     DEVICE$ = LEFT(DEVICE$,2%)
                + CHR$(VAL(RIGHT(DEVICE$,3%)))
\       GOTO 900                                        !WHEN ENTERING DB??:
                                                        ! CHANGE TO WORK IN
                                                        ! SYS CALL 6 + 17


880     DEVICE$ = "SY"+CHR$(0%)
\       LATER.DEVICE$ = "SY"
\       GOTO 900                                        !SINCE NO DEVICE GIVEN
                                                        ! ASSUME IT TO BE SY0:

890     ERROR.FLAG% = 4%
\       GOTO 32000                                      !ILLEGAL DEVICE ERROR
                                                        ! GOTO ERROR SECTION



900 !****************************************************
\   !*
\   !* NOW THAT WE HAVE DETERMINED THAT WE CAN HAVE
\   !*  ACCESS TO THE FILE, LET USE SEE IF THE FILE
\   !*  INDEED EXISTS!
\   !*
\   !****************************************************

910     T% = INSTR(1%,FILE.NAME$,".")
\       GOTO 990 IF T% = 0%
\       X$ = FILE.NAME$
\       FILE.NAME$ = LEFT(FILE.NAME$,T% - 1%)
\       RAD50.FILENAME$ = MID(SYS(CHR$(6%)+CHR$(-10%)
                        + FILE.NAME$),7%,4%)
\       EXTENSION$ = RIGHT(X$,T% + 1%)
\       T% = INSTR(1%,EXTENSION$,"/")
\       GOSUB 950 IF T%
\       RAD50.EXTENSION$ = MID(SYS(CHR$(6%) + CHR$(-10%)
                        + EXTENSION$),7%,2%)
\                                                       !TAKE THE FILE NAME
                                                        ! AND BREAK IT UP INTO
                                                        ! THE FILE NAME AND
                                                        ! ITS EXTENSION THE
                                                        ! RADIX-50 IT.

920     X$ = SYS(CHR$(6%)+CHR$(17%)+CHR$(255%)+CHR$(255%)
                + CHR$(FILE.PROG%) + CHR$(FILE.PROJ%)
                + RAD50.FILENAME$ + RAD50.EXTENSION$
                + STRING$(10%,0%) + DEVICE$
                + CHR$(255%))                           !SYS CALL TO SEE
                                                        ! IF THE FILE IS ON
                                                        ! THE DISK. ERROR SECT
                                                        ! AT 25000 WILL PICK
                                                        ! UP ERROR IF FILE NOT
                                                        ! THERE.

930     GOTO 1000                                       !KEEP GOING

950 !****************************************************
\   !*
\   !* LINES 950 THRU 980 ARE LITTLE SUBROUTINE IN
\   !*  MIDDLE OF FILE.NAME SECTION TO DETERMINE WHETHER
\  .!*  A /D WAS ENTERED, THIS INDICATES TO DELETE FILE.
\   !*
\   !****************************************************

960     SWITCH$ = RIGHT(EXTENSION$,T% + 1%)
\       EXTENSION$ = LEFT(EXTENSION$, T% - 1%)
\       ERROR.FLAG% = 17%
                IF SWITCH$ <> "D"
\       GOTO 32000 IF SWITCH$ <> "D"                    !GET EXTENSION BY
                                                        ! ITSELF, THEN CHECK
                                                        ! FOR A "D".

980     RETURN                                          !END FUNCTION


990     ERROR.FLAG% = 5%
\       GOTO 32000                                      !NO EXTENSION GIVEN
                                                        ! SO ERROR OUT.

1000 !***************************************************
\    !*
\    !* NOW THAT WE KNOW THAT THE FILE IS THERE AND
\    !*   THAT WE CAN GET IT. SEE IF TERMINAL IS
\    !*   AVAILABLE.
\    !*
\    !***************************************************

1010    GOTO 1025 IF LEFT(TERMINAL.NAME$,2%) = "KB"
\       FOR X% = 1% TO KB.TABLE%(0%,0%)
\               GOTO 1040
                  IF KB.TABLE$(X%) = TERMINAL.NAME$
\       NEXT X%                                         !FIND TERMINAL.NAME$
                                                        ! IN TERMINAL TABLE.

1020    ERROR.FLAG% = 6%
\       GOTO 32000                                      !SORRY DID NOT FIND
                                                        ! TERMINAL NAME.

1025    X$ = SYS(CHR$(6%)+CHR$(26%)+CHR$(0%)+CHR$(0%))
\       CURRENT.TERMINAL% = ASCII(MID(X$,4%,1%))        !GET JOB'S CONSOLE.

1030    T% = INSTR(1%,TERMINAL.NAME$,":")
\       TERMINAL.NAME$ = LEFT(TERMINAL.NAME$,T%-1%)
                IF T% <> 0%
\       TERMINAL.NUMBER% =
                VAL(RIGHT(TERMINAL.NAME$,3%))
                IF LEN(TERMINAL.NAME$) > 2%
\       TERMINAL.NUMBER% = CURRENT.TERMINAL%
                IF LEN(TERMINAL.NAME$) = 2%
\       GOTO 1050                                      _!SINCE ONE ENTERED A
                                                       ‾! TERMINAL TO THE
                                                        ! TERMINAL NAME QUEST
                                                        ! USE SAID NUMBER.

1040    KB.WIDTH% = KB.TABLE%(X%,3%)
\       TERMINAL.NUMBER% = KB.TABLE%(X%,2%)
\       TERMINAL.SIZE% = KB.TABLE%(X%,1%)               !GET THE TERMINAL
                                                        ! NUMBER AND PAGE
                                                        ! SIZE.

1050    X$ = SYS(CHR$(6%) + CHR$(10%) + STRING$(20%,0%)
                + "KB" + CHR$(TERMINAL.NUMBER%)
                + CHR$(255%))                           !ASSIGN KB??: TO
                                                        ! THE CURRENT JOB.

1100 !***************************************************
\    !*
\    !* ALL RIGHT PEOPLE WE NOW HAVE THE TERMINAL WE
\    !*  WANT, THE FILE, AND ALL OTHER JAZZ. LET'S
\    !*  PRINT OUT THE STUFF.
\    !*
\    !***************************************************

1110    GOTO 1200 IF ATTACHED% = 0%
                OR PSEUDO.NUMBER%
\       PRINT
\       PRINT "  ***Entering print phase***"
                IF CHAINED.TO%
\       PRINT IF CHAINED.TO%                            !SKIP IF DETACHED
                                                        ! OR ON PSEUDO

1120    INPUT "Is paper at top of form (Y/N)";
                Y$
\       Y$ = LEFT(CVT$$(Y$,-1%),1%)
\       GOTO 1120 IF Y$ <> "Y"
\       PRINT
\       PRINT "        ....PROCESS IN PROGRESS...."     !ASK USER IF PAPER
                                                        ! AT TOP OF FORM.

1200 !***************************************************
\    !*
\    !* NOW OPEN THE FILE, IT SHOULD BE THERE SINCE WE
\    !*   DID A SYS CALL ON IT.
\    !*
\    !***************************************************

1205    FOR C8% = 1% TO C9%
\       INPUT.COUNTER% = 0%                             !LOOP THRU NUMBER
                                                        ! COPIES. PRINT HEADER
                                                        ! PAGE FIRST.

1210    OPEN "[" + ACCOUNT$ + "]" + LATER.DEVICE$ + ":"
                + FILE.NAME$ + "." + EXTENSION$
                FOR INPUT AS FILE #1%                   !ERROR SECTION WILL
                                                        ! TAKE CARE OF
                                                        ! ANY ERROR.

1220    OPEN "KB"+NUM1$(TERMINAL.NUMBER%)+":"
                FOR OUTPUT AS FILE #2%
\       GOSUB 1600 IF C8% = 1%                          !ERROR SECTION WILL
                                                        ! TAKE CARE OF ERROR.
                                                        ! PRINT HEADINGS IF
                                                        ! FIRST COPY.

1230    INPUT LINE #1%,INPUT.LINE$
\       INPUT.COUNTER% = INPUT.COUNTER% + 1%
\       NEW.PAGE%= INSTR(1%,INPUT.LINE$,CHR$(12%))
\       INPUT.LINE$ = CVT$$(INPUT.LINE$,4%)             !READ IN A LINE, SEE
                                                        ! IF NEW PAGE IS
                                                        ! NEEDED.

1240    PRINT #2%,LEFT(INPUT.LINE$,KB.WIDTH%)
                IF NEW.PAGE% = 0% OR INPUT.LINE$ <> ""
\       LINE.COUNTER% = LINE.COUNTER% + 1%
                IF NEW.PAGE% = 0% OR INPUT.LINE$ <> ""
\       GOSUB 1500 IF NEW.PAGE% AND INPUT.COUNTER% > 1%
\       GOSUB 1500 IF NEW.PAGE% AND INPUT.COUNTER% = 1%
                AND INPUT.LINE$ <> ""
\                                                       !PRINT OUT THE LINE
                                                        ! DO NEW PAGE SECTION
                                                        ! IF NEW PAGE. SKIP
                                                        ! NEW PAGE IF ^L ONLY
                                                        ! CHARACTER ON FIRST
                                                        ! LINE.

1250    GOTO 1230                                       !GET MORE STUFF TO
                                                        ! PRINT.

1260    GOSUB 1500
\       NEXT C8%
\       ERROR.FLAG% = 13%
\       GOTO 32000                                      !PRINT ALL COPIES THEN
                                                        ! FLAG ERROR AS END
                                                        ! OF FILE TO STOP
                                                        ! PRINTING

1500 !***************************************************
\    !*
\    !* SIMULATE A NEW PAGE.
\    !*
\    !***************************************************

1510    GOTO 1520 IF TERMINAL.SIZE% = 0%
\       T% = INT(LINE.COUNTER%/TERMINAL.SIZE%)
\       T1% = LINE.COUNTER% - T% * TERMINAL.SIZE%       !REDUCE LINE.COUNTER%
                                                        ! DOWN TO KNOW HOW
                                                        ! MANY LINES ON
                                                        ! PAGE WE PRINTED.
```

```
1520     PRINT #2%  FOR X% = T1% + 1% TO TERMINAL.SIZE%
                 IF T1% > 0%
\        LINE.COUNTER% = 0%
\        RETURN                                          !PUMP OUT THE LINES
                                                         ! NEEDED TO GET TO TOP
                                                         ! OF FORM THEN RETURN.

1600 !***************************************************
\    !*
\    !* PRINT HEADINGS BEFORE DOING REPORT
\    !*
\    !***************************************************

1610     GOSUB 2000
\        PASS.NAME$ = NUM1$(PROJ.NUMBER%) + "  "
                    + NUM1$(PROG.NUMBER%) + " "
                    + THE.NAME$
\        GOSUB 10000
\        PRINT #2%,LEFT(PASS.BACK$(X%),KB.WIDTH%)
                FOR X% = 0% TO 6%
\        PRINT #2%
\        PRINT #2%,LEFT("[[[??????==========================="
               + "==00000000000000000000000*********]]]"
                       ,KB.WIDTH%)
\        PRINT #2%,LEFT("[[[??????==========================="
               + "==00000000000000000000000*********]]]"
                       ,KB.WIDTH%)
\        PRINT #2%
\        PRINT #2%,DATE$(0);"   ";TIME$(0%);"   ";C9%;
               " cop";
\        PRINT #2%,"y"; IF C9% = 1%
\        PRINT #2%,"ies"; IF C9% > 1%
\        PRINT #2%,"   Nodelete" IF SWITCH$ = ""
\        PRINT #2%,"   Delete" IF SWITCH$ = "D"
\        PRINT #2%,"Requested printing of:"
\        PRINT #2%
\        PASS.NAME$ = NUM1$(FILE.PROJ%) + "  "
                    + NUM1$(FILE.PROG%) + "  "
                    + FILE.NAME$ + "."
                    + EXTENSION$

\        GOSUB 10000                                     !PRINT USER HEADER
                                                         ! GET FILE DATA.

1620     PRINT #2%,LEFT(PASS.BACK$(X%),KB.WIDTH%)
                FOR X% = 0% TO 6%
\        PRINT #2% FOR X% = 1% TO 2%
\        PRINT #2%,LEFT("[[[??????==========================="
               + "==00000000000000000000000*********]]]"
                       ,KB.WIDTH%)
\        PRINT #2%,LEFT("[[[??????==========================="
               + "==00000000000000000000000*********]]]"
                       ,KB.WIDTH%)
\        LINE.COUNTER% = 25%
\        GOSUB 1500
\        RETURN                                          !PRINT OUT FILE NAME
                                                         ! AND ACCOUNT.
2000 !***************************************************
\    !*
\    !* THIS SECTION WILL LOOK UP THE NAME IN THE
\    !*  VIRTUAL ARRAY SPOLER.DAT IN [1,0].
\    !*
\    !***************************************************

2010     OPEN "[1,0]SPOLER.VIR" AS FILE #9%,
                RECORDSIZE 2048%
\        DIM #9%,NAME.ACCT$(150%) = 15%
                ,ACCT.PROJ%(150%)
                ,ACCT.PROG%(150%)                        !OPEN AND DEFINE FILE

2020     FOR X% = 1% TO ACCT.PROJ%(0%)
                GOTO 2040
                  IF ACCT.PROJ%(X%) = PROJ.NUMBER%
                  AND ACCT.PROG%(X%) = PROG.NUMBER%
\        NEXT X%                                         !LOOP THREW ALL NAMES
                                                         ! AND ACCOUNT NUMBERS

2030     THE.NAME$ = "??????"
\        GOTO 2050                                       !ACCOUNT NOT FOUND

2040     IF CVT$$(NAME.ACCT$(X%),2%) = ""
                THEN THE.NAME$ = "??????"
                ELSE THE.NAME$ = NAME.ACCT$(X%)          !IF NAME IS BLANK
                                                         ! INDICATE AS ERROR
                                                         ! ELSE SEND NAME BACK.

2050     RETURN                                          !FINISH UP.

10000 !********************************************************
\     !*
\     !* THIS LITTLE ROUTINE WILL PRINT THE LARGE LETTER USED
\     !*        BY SPLRUN (SPOOLING PACKAGE) FOR ANY FILE SPEC
\     !*        OR ACCOUNT NUMBER PASSED TO IT IN PASS.NAME$.
\     !*        ACCOUNTS AND FILE NAMES ARE TREATED THE SAME.
\     !*
\     !********************************************************

10010    DIM #11%,C%(9%,39%)
\        OPEN "SPOOL:CHARS.QUE" AS FILE #11%             !DIM AND OPEN THE
                                                         ! CHARS.QUE.
10020    PASS.BACK$(X%) = "" FOR X% = 0% TO 6%
\        PASS.NAME$ = CVT$$(PASS.NAME$,32%)
\        FOR U8% = 0% TO 6%                              !REMOVE ALL EDIT
                                                         ! STUFF

10025      FOR U9% = 1% TO LEN(PASS.NAME$)
\               LETTER$ = MID(PASS.NAME$,U9%,1%)
\               LETTER$ = " " IF LETTER$ = ","
\               INDEX.IN% = 0%
\               INDEX.IN% = ASCII(LETTER$) - 64%
                       IF ASCII(LETTER$) > 64%
                         AND ASCII(LETTER$) < 91%
\               INDEX.IN% = 27% IF ASCII(LETTER$) = 36%
\               INDEX.IN% = 28% IF ASCII(LETTER$) = 46%
\               INDEX.IN% = 29% IF ASCII(LETTER$) = 63%
\               INDEX.IN% = ASCII(LETTER$) - 18%
                       IF ASCII(LETTER$) > 47%
                         AND ASCII(LETTER$) < 58%
\               INDEX.IN% = 29% IF INDEX.IN% = 0%
\                                                        !START LOOP THRU THE
                                                         ! PASSED ARGUMENT.
                                                         ! GET THE INDEX TO
                                                         ! VIRTUAL ARRAY.

10030           FOR U7% = 0% TO 4%
\                   IF C%(U8%,INDEX.IN%) AND 2**U7%
                     THEN
                      PASS.BACK$(U8%) = PASS.BACK$(U8%)
                      + " "
                    ELSE
                      PASS.BACK$(U8%) = PASS.BACK$(U8%)
                      + LETTER$                         !DETERMINE WHETHER WE
                                                        ! SHOULD BUILD A
                                                        ! CHARACTER OR SPACE.

10040           NEXT U7%                                !LOOP THREW ALL FIVE
                                                        ! BITS.

10050           PASS.BACK$(U8%) = PASS.BACK$(U8%) + " "
\           NEXT U9%
\       NEXT U8%                                        !PAD A SPACE BETWEEN
                                                        ! THINGS THEN CONT.
10060   RETURN
\!*********************************************************
\!*
\!* THE DATA IS RETURNED IN PASS.BACK$() ELEMENTS 0 TO
\!*     6.
\!*
\!*********************************************************

19100 !*********************************************************
\     !*
\     !* THIs SECTION SENDS INFO TO OPSER.
\     !*
\     !*********************************************************

19105   DEF FNO9$(C$)                                   !SEND MESSAGE TO OPSER
19110   S1$=CHR$(6%)+CHR$(-10%)
        \ S2$=CHR$(6%)+CHR$(22%)+CHR$(-1%)+CHR$(0%)
                +"OPSER "+STRING$(10%,0%)
        \ S$=SYS(CHR$(6%)+CHR$(22%)+CHR$(0%)+CHR$(0%))
        \ S$=SYS(CHR$(6%)+CHR$(-21%))
                ! SYS CALL STRINGS :     S1$=FILENAME STRING SCAN.
                !                        S2$=SEND MSG. TO 'OPSER'.
                ! SYS CALLS :   REMOVE THIS JOB AS A RECEIVER.
                !               DROP TEMP. PRIVILEGES.

19120   S$=SYS(CHR$(6%)+CHR$(22%)+CHR$(0%)+CHR$(0%))
                ! REMOVE THIS JOB AS A RECEIVER.

19140   C$=CVT$$(C$,189%)
        \ GOTO 19160 IF LEN(C$)=0%
                ! TRIM PARITY BIT, DISCARD EXCESS CHARS.AND
                !       LEADING/TRAILING SPACES  TABS, CONVERT
                !       LOWER CASE TO UPPER CASE,REPLACE EMBEDDED
                !       SPACES  TABS WITH ONE SPACE.
                ! IF NOTHING LEFT NO PROCESSING TO BE DONE.

19150   Ir LEN(C$) <= 19% THEN
                S$=SYS(S2$+CHR$(LEN(C$)+1%)+C$)
                ELSE
                S$=SYS(S2$+CHR$(255%)+LEFT(C$,19%))
        \       C$=RIGHT(C$,20%)
        \       GOTO 19150
                ! SEND THE DATA TO 'OPSER'.

19160   FNEND

25000 !*********************************************************
\     !*
\     !* ERROR SECTION FROM ON ERROR GOTO.
\     !*
\     !*********************************************************

25005   Ir ERR = 28% THEN ERROR.FLAG% = 15%
\               RESUME 32000                            !^C TRAP

25010   Ir ERL = 110% THEN ERROR.FLAG% = 7%
\               RESUME 32000                            !^Z IS FLAGGED AS A
                                                        ! 7 THEN GO TO ERROR
                                                        ! PROCESSING SECTION.

25015   IF ERL = 640% THEN RESUME 700                   !FILL IN KB NAME TABLE
                                                        ! FROM FILE.

25020   IF ERL = 510% THEN ERROR.FLAG% = 8%
\               RESUME 32000                            !BAD LINE NUMBER
                                                        ! PASSED TO PROGRAM.

25030   IF ERL = 720% THEN ERROR.FLAG% = 9%
\               RESUME 32000                            !BAD ACCOUNT VALUE.

25040   IF ERL = 850% THEN ERROR.FLAG% = 4%
\               RESUME 32000                            !ILLEGAL DEVICE NUMBER

25050   IF ERL = 1030% THEN ERROR.FLAG% = 10%
\               RESUME 32000                            !ILLEGAL TERMINAL #.

25060   Ir ERL = 1050% OR ERL = 1220
                THEN ERROR.FLAG% = 11%
\               RESUME 32000                            !KB CAN'T BE ASSIGNED.
25070   IF ERL = 920% OR ERL = 1210
                THEN ERROR.FLAG% = 12%
\               RESUME 32000                            !CAN'T FIND FILE OR
                                                        ! ACCOUNT.

25080   Ir ERL = 1230% THEN RESUME 1260                 !END OF INPUT FILE.

25090   Ir ERL = 1240% OR ERL = 1520
                THEN ERROR.FLAG% = 14%
\               RESUME 32000                            !ERROR TO OUTPUT FILE.

25110   Ir ERL = 32715 THEN PROGRAM.TO$ = ""
\               ERROR.FLAG% = 16%
\               RESUME 32000                            !PROGRAM TO CHAIN TO
                                                        ! DOES NOT EXIST.

25120   IF ERL = 2010% OR ERL = 2020%
                THEN THE.NAME$ = "??????"
\               RESUME 2050                             !CAN'T FIND ACCOUNT OR
                                                        ! FILE OR SUBCRIPT
                                                        ! OVERFLOW.

25130   IF ERL = 32712%
                THEN ERROR.FLAG% = 18%
\               RESUME 32000                            !KILL DID NOT WORK


25500   ERROR.FLAG% = 99%                               !UNKNOWN ERROR


32000 !*********************************************************
\     !*
\     !* THIs Is THE ERROR.FLAG PROCESSING SECTION.
\     !*
\     !*********************************************************
```

```
32010   IF ERROR.FLAG% = 1%
                THEN ERROR.OUT$ = "?Core common error"
        ELSE IF ERROR.FLAG% = 2%
                THEN ERROR.OUT$ = "?Illegal account specs"
        ELSE IF ERROR.FLAG% = 3%
                THEN ERROR.OUT$ = "?Protection violation"
        ELSE IF ERROR.FLAG% = 4%
                THEN ERROR.OUT$ = "?Illegal device specs"
        ELSE IF ERROR.FLAG% = 5%
                THEN ERROR.OUT$ = "?No file ext. given"
        ELSE IF ERROR.FLAG% = 6%
                THEN ERROR.OUT$ = "?Can not find terminal name"
        ELSE IF ERROR.FLAG% = 7%
                THEN 110
        ELSE IF ERROR.FLAG% = 8%
                THEN ERROR.OUT$ = "?Bad line number passed"
        ELSE IF ERROR.FLAG% = 9%
                THEN ERROR.OUT$ = "?Illegal account number"
        ELSE IF ERROR.FLAG% = 10%
                THEN ERROR.OUT$ = "?Illegal terminal # specified"
        ELSE IF ERROR.FLAG% = 11%
                THEN 32500
        ELSE IF ERROR.FLAG% = 12%
                THEN ERROR.OUT$ = "?Can not find file or account"
        ELSE IF ERROR.FLAG% = 13%
                THEN 32700
        ELSE IF ERROR.FLAG% = 14%
                THEN ERROR.OUT$ = "?Error to output terminal"
        ELSE IF ERROR.FLAG% = 15%
                THEN ERROR.OUT$ = "?^C trap"
        ELSE Ir ERROR.FLAG% = 16%
                THEN ERROR.OUT$ = "?Chaining to prog. not found"
        ELSE Ir ERROR.FLAG% = 17%
                THEN ERROR.OUT$ = "?Illegal switch on filename"
        ELSE IF ERROR.FLAG% = 18%
                THEN ERROR.OUT$ = "?Deletion of file in error"
        ELSE ERROR.OUT$ = "?Unknown error AT LINE "
                +NUM1$(ERL) + " error: "+NUM1$(ERR)
\                                                       !MAKE UP OUTPUT STRING
                                                        ! WITH CORRECT ERROR.
                                                        ! IF 7 OR 13 GOTO
                                                        ! 32700, IF 11 GOTO
                                                        ! 32500.

32050   GOTO 32090 IF ATTACHED% = 0%
\       PRINT
\       PRINT ERROR.OUT$
\       GOTO 32060 IF PSEUDO.NUMBER%
\       GOTO 32700                                      !IF ATTACHED PRINT
                                                        ! OUT ERROR MESSAGE
                                                        ! IF NOT PSEUDO THEN
                                                        ! GOTO END.

32060   C$ = ERROR.OUT$ + "--------" + "ON ACCOUNT"
                + "[" + NUM1$(PROJ.NUMBER%) + ","
                + NUM1$(PROG.NUMBER%) + "]"
                + "  WHILE IN SPOLER ON PSEUDO KB"
\       X$ = FNO9$(C$)
\       GOTO 32700                                      !SEND ERROR MESSAGE
                                                        ! TO OPSER SINCE ON
                                                        ! PSEUDO KB#. THEN END
                                                        ! PROGRAM

32090   C$ = ERROR.OUT$ + CHR$(13%) + "ON ACCOUNT"
                + "[" + NUM1$(PROJ.NUMBER%) + ","
                + NUM1$(PROG.NUMBER%) + "]"
                + "  WHILE IN SPOLER IN DETACH MODE"
\       X$ = FNO9$(C$)
\       GOTO 32700                                      !SEND ERROR MESSAGE
                                                        ! TO OPSER SINCE WE
                                                        ! ARE IN DETACHED
                                                        ! MODE.
```

```
32500 !****************************************************
\      !*
\      !* THIS SECTION GIVES USER CHOICE OF OPTION TO
\      !*    PICK WHEN THE TERMINAL IS NOT AVAILABLE.
\      !*
\      !****************************************************

32505   GOTO 32530 IF (ATTACHED% = 0% OR PSEUDO.NUMBER%)
                AND LEFT(TERMINAL.NAME$,2%) <> "KB"
\       GOTO 32540 IF (ATTACHED% = 0% OR PSEUDO.NUMBER%)
                AND LEFT(TERMINAL.NAME$,2%) = "KB"      !IF WE ARE IN A DET
                                                        ! STATE OR ON PSEUDO
                                                        ! GOTO QUE SECTION.

32510   ATTEMPT.KB% = ATTEMPT.KB% + 1%
\       GOTO 32580 IF ATTEMPT.KB% = 11%
\       SLEEP 5% IF ATTEMPT.KB% > 1%
\       PRINT "%Attempt ";(ATTEMPT.KB%);" failed"
                IF ATTEMPT.KB% > 1%
\       GOTO 1050 IF ATTEMPT.KB% > 1%
\       PRINT
\       PRINT "%Terminal not available for printing"
\       PRINT
\       TIME.NEW$ = "12:00" IF LEFT(TIME$(0),2%) < "12"
                AND LEFT(TIME$(0),2%) > "10"
\       TIME.NEW$ = "16:00" IF LEFT(TIME$(0),2%) > "13"
                AND LEFT(TIME$(0),2%) < "16"
\       TIME.NEW$ = "a few min." IF TIME.NEW$ = ""
\       PRINT "Options:
\       PRINT "         1. Wait (10 times then queued up)"
\       PRINT "         2. Queue to spooler (report available"
                         " after ";TIME.NEW$;")"
              IF LEFT(TERMINAL.NAME$,2%) <> "KB"
\       PRINT "         3. Queue to high speed printer"
\                                                       !ADD ONE TO ATTEMPTS
                                                        ! THEN IF FIRST TRY
                                                        ! AND FAILED THEN AS
                                                        ! OPTIONS. OPTION 2
                                                        ! ONLY IF A NAMED
                                                        ! TERMINAL.

32520   INPUT "Answer";ANSWER$
\       QUE.DELETE$ = ""
\       QUE.DELETE$ = "/DE" IF SWITCH$ = "D"
\       GOTO 1050 IF ANSWER$ = "1"
\       GOTO 32530 IF ANSWER$ = "2"
              AND LEFT(TERMINAL.NAME$,2%) <> "KB"
\       GOTO 32540 IF ANSWER$ = "3"
\       GOTO 32520                                      !GET THE ANSWER AND
                                                        ! BRANCH TO WHERE WE
                                                        ! MUST BE.

32530   X$ = SYS(CHR$(8%)+"[1,0]SPOLER"+CHR$(13%)
                + CVT%$(32600) + "Q LP:/FO:"
                + TERMINAL.NAME$ + "="
                + LATER.DEVICE$ + ":"
                + "[" + ACCOUNT$ + "]"
                + FILE.NAME$ + "." + EXTENSION$
                + QUE.DELETE$
                + CHR$(13%)
                + PROGRAM.TO$ + "~"
                + CVT%$(LINE.NUMBER%) + "~"
                + CVT%$(ATTACHED%) + "~"
                + CVT%$(PSEUDO.NUMBER%) + "~")
\       CHAIN "$QUE" 31000                              !QUE UP FILE TO
                                                        ! PRINT WITH FORM
                                                        ! EQUAL TO TERMINAL
                                                        ! NAME.

32540   X$ = SYS(CHR$(8%)+"[1,0]SPOLER"+CHR$(13%)
                + CVT%$(32600) + "Q LP:/FO:NORMAL = "
                + LATER.DEVICE$ + ":"
                + "[" + ACCOUNT$ + "]"
                + FILE.NAME$ + "." + EXTENSION$
                + QUE.DELETE$
                + CHR$(13%)
                + PROGRAM.TO$ + "~"
                + CVT%$(LINE.NUMBER%) + "~"
                + CVT%$(ATTACHED%) + "~"
                + CVT%$(PSEUDO.NUMBER%) + "~")
\       CHAIN "$QUE" 31000                              !QUE UP FILE TO
                                                        ! PRINT WITH FORM
                                                        ! EQUAL TO NORMAL
                                                        ! SO GOES TO HIGH
                                                        ! SPEED PRINTER.

32580   PRINT "?Doing queing"
\       GOTO 32530
                IF LEFT(TERMINAL.NAME$,2%) <> "KB"
\       GOTO 32540
                IF LEFT(TERMINAL.NAME$,2%) = "KB"       !10 ATTEMPTS FAILED
                                                        ! SO DO QUEING.

32600 !****************************************************
\      !*
\      !* REENTER FROM QUE TO THIS POINT
\      !*
\      !****************************************************

32610   Z9$ = SYS(CHR$(7%))
\       Z9$ = RIGHT(Z9$,2%)
\       T% = INSTR(1%,Z9$,CHR$(13%))
\       T1% = INSTR(T%,Z9$,"~")
\       PROGRAM.TO$ = MID(Z9$,T% + 1%,T1%-T%-1%)
\       LINE.NUMBER% = CVT$%(MID(Z9$,T1% + 1%, 2%))
\       T2% = INSTR(T1%+1%,Z9$,"~")
\       ATTACHED% = CVT$%(MID(Z9$,T2% + 1%,2%))
\       T3% = INSTR(T2%+1%,Z9$,"~")
\       PSEUDO.NUMBER% = CVT$%(MID(Z9$,T3% + 1%,2%))    !RETURN FROM QUE AND
                                                        ! RETORE ALL VALUES
                                                        ! NEEDED TO FINISH UP

32700 !****************************************************
\      !*
\      !* THIS SECTION ENDS THE PROGRAM, AND THEN DOES
\      !*   ANY NECESSARY CHAINS, IF IT IS
\      !*   DETACHED AND NOT CHAINING OUT THEN KILL
\      !*   THE JOB.
\      !*
\      !****************************************************

32710   CLOSE #1%,#2%,#11%
\       X$ = SYS(CHR$(6%) + CHR$(11%) + STRING$(20%,0%)
                + "KB" + CHR$(TERMINAL.NUMBER%)
                + CHR$(255%))                           !DEASSIGN TERMINAL.

32712   GOTO 32715 IF ERROR.FLAG% <> 13%
                OR SWITCH$ <> "D"
\       KILL "[" + ACCOUNT$ + "]" + LATER.DEVICE$ + ":"
                + FILE.NAME$ + "." + EXTENSION$         !KILL THE FILE IF
                                                        ! /D ON FILE NAME

32715   GOTO 32720 IF ASCII(PROGRAM.TO$) = 0%
\       CHAIN PROGRAM.TO$   LINE.NUMBER%                ! IF PROGRAM EXISTS
                                                        ! TO CHAIN DO SO.

32720   GOTO 32767 IF ATTACHED%
\       X$ = SYS(CHR$(6%) + CHR$(8%)
                + CHR$(PEEK(518%)/2%) + STRING$(23%,0%)
                + CHR$(0%) + CHR$(255%))                !SINCE WE ARE DETACHED
                                                        ! KILL OFF THE JOB.

32730   GOTO 32767 IF ERROR.FLAG% <> 13%
                OR SWITCH$ <> "D"
\       KILL "[" + ACCOUNT$ + "]" + LATER.DEVICE$ + ":"
                + FILE.NAME$ + "." + EXTENSION$         !KILL THE FILE IF
                                                        ! /D ON FILE NAME
                                                        ! REQUESTED

32767   END
```

[11,11] SPOL1.BAS

```
001     EXTEND
\ !**********************************************
\ !*
\ !*    THIS PROGRAM WAS WRITTEN BY M H KOPLITZ
\ !*     SYSTEMS ADMINISTRATOR
\ !*     , ALLIS-CHALMERS HTD, YORK
\ !*     PA 17404.
\ !*
\ !*    THIS PROGRAM MAKES A VIRTUAL FILE FROM
\ !*    ACCT.SYS FOR THE SPOLER.BAS PROGRAM.
\ !*
\ !**********************************************

010     OPEN "$ACCT.SYS" FOR INPUT AS FILE #1%
\       OPEN "#SPOLER.VIR" FOR OUTPUT AS FILE #2%
\       DIM #2%,NAME.ACCT$(150%) = 15%
                ,ACCT.PROJ%(150%)
                ,ACCT.PROG%(150%)

015     ON ERROR GOTO 100

020     INPUT #1%,PROJ%,PROG%,PASS.WORD$,QUOTA,
                UFD%,THE.NAME$                          !INPUT LINE

030     COUNTER% = COUNTER% + 1%
\       NAME.ACCT$(COUNTER%) = LEFT(THE.NAME$,15%)
\       ACCT.PROJ%(COUNTER%) = PROJ%
\       ACCT.PROG%(COUNTER%) = PROG%
\       GOTO 020                                        !PLACE ACCT.SYS DATA INTO .VIR
```

```
100     IF ERL = 020 THEN RESUME 200
110     ON ERROR GOTO 0
120     GOTO 32767

200     ACCT.PROJ%(0%) = COUNTER%
\       GOTO 32767

32767   CLOSE #1%,#2%
\       END
```

[11,11] CHAR.BAS

```
001      EXTEND
010 !**********************************************************
\   !*
\   !* THIS PROGRAM WAS WRITTEN BY M H KOPLITZ, 10-AUG-81,
\   !*  SYSTEMS ADIMINISTRATOR, ALLIS-CHALMERS HTD.
\   !*
\   !* THIS PROGRAM WILL PRODUCE A LINE IN RSTS SPOOLER
\   !*  FORMAT AND SEND IT TO ANY OUTPUT DEVICE.
\   !*
\   !*********************************************************

020      PRINT "CHAR  V1.0  Allis-Chalmers HTD   ";
                "       Make RSTS spooling letters"
\        PRINT                                         !PRINT BANNER

030      INPUT "Output device";OUTPUT.DEVICE$
\        OPEN OUTPUT.DEVICE$
                 FOR OUTPUT AS FILE #1%                !OPEN OUTPUT

040      PRINT "Line to convert (return = done)";
\        INPUT LINE PASS.NAME$
\        PASS.NAME$ = CVT$$(PASS.NAME$,4%)
\        GOTO 32767 IF PASS.NAME$ = ""                 !INPUT LINE

050      GOSUB 10000
\        PRINT #1%,PASS.BACK$(X%) FOR X% = 0% TO 6%
\        PRINT #1%  FOR X% = 1% TO 3%
\        GOTO 040                                      !CONVERT, PRINT OUT
                                                       ! THEN GET MORE.


10000 !**********************************************************
\      !*
\      !* THIS LITTLE ROUTINE WILL PRINT THE LARGE LETTER USED
\      !*       BY SPLRUN (SPOOLING PACKAGE) FOR ANY FILE SPEC
\      !*       OR ACCOUNT NUMBER PASSED TO IT IN PASS.NAME$.
\      !*       ACCOUNTS AND FILE NAMES ARE TREATED THE SAME.
\      !*
\      !*********************************************************

10010   DIM #11%,C%(9%,39%)
\       OPEN "SPOOL:CHARS.QUE" AS FILE #11%           !DIM AND OPEN THE
                                                       ! CHARS.QUE.
10020   PASS.BACK$(X%) = "" FOR X% = 0% TO 6%
\       PASS.NAME$ = CVT$$(PASS.NAME$,32%)
\       FOR U8% = 0% TO 6%                             !REMOVE ALL EDIT
                                                       ! STUFF

10025     FOR U9% = 1% TO LEN(PASS.NAME$)
\               LETTER$ = MID(PASS.NAME$,U9%,1%)
\               LETTER$ = " " IF LETTER$ = ","
\               INDEX.IN% = 0%
\               INDEX.IN% = ASCII(LETTER$) - 64%
                        IF ASCII(LETTER$) > 64%
                          AND ASCII(LETTER$) < 91%
\               INDEX.IN% = 27% IF ASCII(LETTER$) = 36%
\               INDEX.IN% = 28% IF ASCII(LETTER$) = 46%
\               INDEX.IN% = 29% IF ASCII(LETTER$) = 63%
\               INDEX.IN% = ASCII(LETTER$) - 18%
                        IF ASCII(LETTER$) > 47%
                          AND ASCII(LETTER$) < 58%
\               INDEX.IN% = 29% IF INDEX.IN% = 0%
\                                                      !START LOOP THRU THE
                                                       ! PASSED ARGUMENT.
                                                       ! GET THE INDEX TO
                                                       ! VIRTUAL ARRAY.

10030           FOR U7% = 0% TO 4%
\                   IF C%(U8%,INDEX.IN%) AND 2**U7%
                     THEN
                      PASS.BACK$(U8%) = PASS.BACK$(U8%)
                      + " "
                    ELSE
                      PASS.BACK$(U8%) = PASS.BACK$(U8%)
                      + LETTER$                        !DETERMINE WHETHER WE
                                                       ! SHOULD BUILD A
                                                       ! CHARACTER OR SPACE.

10040           NEXT U7%                               !LOOP THREW ALL FIVE
                                                       ! BITS.

10050           PASS.BACK$(U8%) = PASS.BACK$(U8%) + " "
\           NEXT U9%
\       NEXT U8%                                       !PAD A SPACE BETWEEN
                                                       ! THINGS THEN CONT.
10060   RETURN
\!**********************************************************
\!*
\!* THE DATA IS RETURNED IN PASS.BACK$() ELEMENTS 0 TO
\!*      6.
\!*
\!**********************************************************

32767   CLOSE #1%
\       END
```

♥

PROPOSED STANDARD EDT 2.0 INITIALIZER

By David Spencer, Infinity Software Corporation

1.0 INTRODUCTION

This is the first of two articles describing techniques to get more out of EDT version two. This installment deals with an EDTINI.EDT initializer file. It was created in a joint effort of Steven Edwards of Software Techniques and myself.

The second article will deal with EDT hints and kinks. This includes a "wish-list" of additional commands, work-arounds to problems, and complaints.

Credit is due to Steve and members of the Software Techniques staff, which have contributed lots of useful information that has been incorporated into these articles.

2.0 REASONS FOR EXTENDED COMMANDS

Everyone agrees that EDT version 2 is a wonderful editor. What we can't agree on, though, is a standard set of redefined keys. It seems everybody has their own idea on the best functions. Hopefully this argument can be cooled somewhat with the introduction of a standard EDT initializer file.

The distributed EDT comes with assignments to a number of keys for keypad editing (figure 1). Although these keys do an excellent job, more sophisticated users outgrow the set supplied by DEC.

For example, EDT provides a number of simultaneous buffers. The range of commands available for buffer manipulation is fantastic. But there is no way to access buffers through editing keys. Any buffer changes must be executed with line-mode commands either in line-mode (unsatisfactory) or with the keypad "GOLD 7" command (cumbersome). Either way, keypad editing is reduced to something only slightly better than line-mode editing.

What I present with this article is a standard set of EDT extensions. These extensions allow users to access buffers, do file input/output, and other things available only in line-mode. This set (figure 2) does not interfere with any keys pre-assigned by DEC. However, these extra definitions increase EDT's usability many-fold.

3.0 DEFINITION OF TERMS

A number of terms are used throughout these two articles to describe commands. To prevent confusion, definitions of these terms follow. I assume that the reader already has some knowledge of EDT, and has at least glanced at the manual.

- GOLD
  "GOLD" is a synonym for the blue key on VT52's and the PF1 key on VT100's.

- CONT x
  "CONT x" is used to symbolize the typing of the control character "x".

- GOLD x
  "GOLD x" is the striking of the GOLD key followed by key "x", where "x" is a letter, control character, or keypad key.

- Keystroke
  A keystroke is one or more keyboard keys typed to complete an editing sequence. For example, the combination "GOLD CONT Z" is thought of as one keystroke.

- Internal Key Number
  EDT uses an internal numbering scheme to uniquely identify keystrokes. The second article will include a table of all the internal EDT key numbers and keystrokes.

- Iteration
  All EDT editing keys can be prefixed with a repetition count. This is entered by striking the GOLD key, a number, and a keystroke.

4.0 NEW KEY DESCRIPTIONS

Here is a detailed list of the additional EDT key definitions. Nearly all of these commands accept an iteration count.

In those commands which ask for input (like buffer name), the input must be terminated with the keypad ENTER key. If for some reason you wish to abort the command asking for input, type "CONT U" and EDT will terminate the command without any action taking place.

NEW COMMANDS

GOLD ARROW-UP

Move the editing window upward 22 lines. The vertical orientation of the cursor will be maintained from the line it left.

GOLD ARROW-DOWN

Move the editing window downward 22 lines. Once again, the vertical orientation of the cursor will be maintained from the line previous to the GOLD ARROW-DOWN.

CONT B

Move backward a word. Identical to striking the keypad "5" key followed by the keypad "1" key. "CONT B" works to move the cursor backward regardless of the motion flags.

CONT F

Move forward a word. This key is identical to striking the keypad "4" key and then the keypad "1" key. "CONT F" works the same at all times.

CONT P

Move to next paragraph. Using this key moves the cursor to the next paragraph. (The default definition for a paragraph delimiter is two consecutive carriage-returns.)

CONT G

Paste contents of named buffer. Typing CONT G will ask for the buffer name on the twenty-third line on the screen. Enter the buffer name, and that buffer's contents will be inserted at the current position.

CONT X

Cut region to named buffer. Select a region of text and type CONT X. Enter the buffer name. The selected region will be removed from the current editing buffer and placed into the buffer name entered.

GOLD CONT D

Toggle word delimiter sets. This allows the use of two completely different word delimiter sets. The implementation in this initializer is one word delimiter set for programming, and one for word processing.

GOLD CONT G

Replace region with named buffer. This command works very much like the keypad "GOLD 9" command. Select a region of text, and type ".GOLD CONT G". Enter the buffer name. The selected region will be replaced with the contents of the entered buffer.

GOLD CONT H

Transpose previous two characters. By striking the GOLD key and backspace, the previous two characters will be transposed. This is helpful for typists that frequently type "teh" instead of "the".

GOLD CONT W

Toggle 80/132 screen width. A very useful command to switch the screen between 80 and 132 column mode. (Just for VT100's though.)

GOLD CONT X

Copy region to named buffer. Select a region and type GOLD CONT X. EDT will accept a buffer name. The selected region will then be copied to the buffer name entered.

GOLD CONT Z

Complete edit and leave EDT to the monitor. It is the equivalent of invoking the EDT "EXIT" command.

GOLD .

Insert special file marker. This keystroke inserts the "~~/\~~" special position marker. The mark can be found later with the "GOLD /" command. This command is useful when you leave EDT, and come back later and want to find your place.

GOLD /

Locate position file marker. This command locates and deletes the special position marker from the text buffer. Leaves the cursor in the position formerly occupied by the marker.

GOLD B

Switch to a named buffer. Enter the buffer name to switch into followed by the ENTER key. The screen will then refresh and you'll be editing your buffer.

GOLD C

Change case, but retain first letter. This command will advance a character, and change case on the remainder of the word. Very useful for changing totally uppercase words to capitalized lowercase, or capitalized words to all uppercase.

GOLD F

Fill a paragraph. Entering this command will cause EDT to set the mark, move to next paragraph, and fill the selected region.

GOLD I

Read file into buffer. Enter the file to open followed by ENTER, and the buffer to copy the file into. EDT will leave you in the buffer selected after the read is completed. VERY useful in extracting needed text from other files. Why write something twice?

GOLD L

Return to last position. This executes the EDT "FIND LAST" command. It is most useful in bouncing between two buffers without entering the buffer names with the "GOLD B" command. EDT does, however, seem to lose what the other buffer name was when a cut or paste is performed.

GOLD M

Switch to buffer MAIN. Identical to entering "GOLD B" and "MAIN" as the buffer to switch to. Returns to the spot where the cursor was when "MAIN" was left.

GOLD O

Output named buffer to a file. EDT will ask for the output filename, and then the buffer name to write to that file. EDT is smart enough to rename any existing file by that name given to ".BAK" before writing out the file. You will be left in the buffer name selected after the output is complete.

GOLD Q

Abort edit. This command aborts the current edit, saves the journal file, and leaves EDT. This is the equivalent of invoking the EDT "QUIT/SAVE" command.

GOLD S

Show all named buffers. Same as the line-mode command "SHOW BUFFER". This command helps when you've forgotten what buffers were in use.

5.0 OUTLINE OF INITIALIZER FILE

The initializer file (figure 3) is set up in stages of definition. Macro definitions come first, followed by key definitions, with terminal characteristics finishing up. Separating the various types of commands makes it easier to add new commands in the future.

The first thing done is defining the word delimiter toggle and screen width toggle macros. The macros are accessed later by keystrokes, and by the initializer itself.

Following the macro definitions comes the key definitions. They go from control characters, to GOLD control characters, and then to GOLD with letters. This loosely follows the ASCII character set and EDT's internal table.

The next step is to set terminal characteristics. The word wrap is defined as the seventy-ninth column. Truncate mode is set (this allows EDT to use the VT100 scrolling region for screen changes). EDT is then told to "SET KEYPAD" and "SET MODE CHANGE". These commands insure EDT will be started in keypad mode screen editing. The word delimiters are then set to programming.

The last line in the initializer insures that the user will start editing in the MAIN buffer.

By looking at this initializer file and reading the EDT manual, many useful things can be learned about EDT.

6.0 NOTES ON USING BUFFERS

EDT will allow buffer names at least sixty characters long. The valid characters for any buffer name must be alpha (A-Z) followed by any combination of alphanumeric characters and the underline ("__"). Of course, remember not to get carried away with names. Sooner or later you'll have to type that name again!

There is a very useful thing to know where pulling in files with EDT. There are times when you simply want to place the entire contents of a file directly into the current position in the editing buffer. This is done quite simply. Type the sequence "GOLD I" and enter the file name. When EDT asks for the buffer, enter the current buffer name followed with a period. (Such as "MAIN.")

This will copy the file directly into the current position of the buffer named. By supplying a buffer name other than the one currently being edited, the text will be inserted at the last cursor position.

If the period is not supplied, EDT will insert the contents of the opened file at the top of the specified buffer. And, of course, if the entered buffer name doesn't exist, EDT will create it.

7.0 USE FILES LIKE A LIBRARY

By opening files and using named buffers, productivity can be increased substantially. When writing programs or creating documents, its a cinch to reference an existing file. Some useful function, subroutine, or lump of text can then be extracted and pasted into the file being edited. It becomes very easy to have one file somewhere on the system that contains all common source code.

8.0 USE EDT FOR MULTIPLE EDITS

Many files can be edited in a single EDT session. By using the "GOLD I" input and "GOLD O" output commands, files may be read into buffers, modified, and written back out. So, if you want, you could stay in EDT all day.

9.0 CLOSING WORDS

I recently had the chance to try EDT on VAX. The initializer file works perfectly, with one exception. Apparently "CONT X" is a key that VMS intercepts. The work-around I used was "CONT V" to duplicate the "CONT X" functions.

I also found VMS rather impolite in dealing with a "CONT Y". It doesn't seem to matter what you're doing, "CONT Y" stops it immediately. This of course includes TECO and EDT.

That does it for this installment. In the next issue, I will provide several figures and lists of some internals, neat ideas, and problem areas in EDT. I have quite a lot of interesting information. Hope everybody can wait until the next issue. Until then, enjoy!

```
        DELETE              Delete previous character
        CONT A, GOLD A      Compute tab level
        CONT D, GOLD D      Decrease tab level
        CONT E, GOLD E      Increase tab level
        CONT H              Go to start of line
        CONT J              Delete to start of word
        CONT K              Define key
        CONT R              Refresh screen
        CONT T, GOLD T      Adjust tabs
        CONT U, GOLD U      Delete to start of line
        CONT W, GOLD W      Refresh screen
        CONT Z, GOLD Z      Return to line mode

(The keypad keys are not shown  here  since  their  function
should be well known to all.)
```

FIGURE 1. Standard Keystroke Assignments

```
Text editing keys

     GOLD ARROW-UP    Go backward 22 lines
     GOLD ARROW-DOWN  Advance 22 lines

     CONT B           Move backward a word
     CONT F           Advance a word
     GOLD CONT H      Transpose previous two characters
     CONT P           Move to next paragraph

     GOLD C           Advance character, change case on
                      remainder of word
     GOLD F           Set mark, move paragraph, fill region

Utility keys

     CONT G           Get contents of named buffer and
                      insert in text at position
     GOLD CONT G      Substitute contents of named buffer
                      with selected region.
     CONT X           Transfer region to named buffer
                      and cut from text
     GOLD CONT X      Transfer region to named buffer
                      and retain in text

     GOLD .           Insert special marker
     GOLD /           Locate and delete special marker
     GOLD I           Open file and copy to named buffer
     GOLD O           Output named buffer to a file
     GOLD S           Show named buffers

Context changing keys

     GOLD CONT D      Toggle word delimiter sets
     GOLD CONT W      Toggle 80/132 width
     GOLD CONT Z      Complete edit and leave EDT

     GOLD B           Switch to a named buffer
     GOLD L           Return to last position
     GOLD M           Switch to buffer MAIN
     GOLD Q           Abort edit, save journal, leave EDT
```

FIGURE 2. Proposed Additions to Key Assignments

```
!        Standard Initializer File        V01.01  15-Sep-81
!+
!        Word Delimiter Macros
!-
DEF M DELIMITERS_PROGRAMMING
F=DELIMITERS_PROGRAMMING
I
DEF K GOLD CONT D AS "EXT DELIMITERS_WORD_PROCESSING."
^Z
C; ISE EN WO '^Z 9ASC 10ASC 11ASC 12ASC 13ASC 27ASC I ()[],-+*/='^Z EX
^Z
!
DEF M DELIMITERS_WORD_PROCESSING
F=DELIMITERS_WORD_PROCESSING
I
DEF K GOLD CONT D AS "EXT DELIMITERS_PROGRAMMING."
^Z
C; ISE EN WO '^Z 9ASC 10ASC 11ASC 12ASC 13ASC 27ASC I ,'^Z EX
^Z
!+
```

FIGURE 3. New Initializer File Listing (1 of 2)

```
!       Screen Width Macros
!-
DEF M WIDTH_132
I=WIDTH_132
DEF K GOLD CONT W AS "EXT WIDTH_80."
SE SC 132
^Z
!
DEF M WIDTH_80
I=WIDTH_80
DEF K GOLD CONT W AS "EXT WIDTH_132."
SE SC 80
^Z
!+.
!       Keys Definitions
!-
DEF K CONT B AS "-W."
DEF K CONT F AS "+W."
DEF K CONT G AS "PASTE=?'Put buffer: '."
DEF K CONT P AS "PAR."
DEF K CONT X AS "CUTSR=?'Cut buffer: '."
!
DEF K GOLD 12     AS "(-22V)."
DEF K GOLD 13     AS "(+22V)."
DEF K GOLD CONT D AS "EXT DELIMITERS_WORD_PROCESSING."
DEF K GOLD CONT G AS "CUTSR=DELETE PASTE=?'Rep buffer: '."
DEF K GOLD CONT H AS "(-C D-C C UNDC)."
DEF K GOLD CONT W AS "EXT WIDTH_132."
DEF K GOLD CONT X AS "EXT CO SELECT TO=?'Cop buffer: ' ; F L."
DEF K GOLD CONT Z AS "EXT EX."
!
DEF K GOLD . AS "I~~/\~~~Z -6C."
DEF K GOLD / AS "S%~~/\~~%%."
DEF K GOLD B AS "EXT F=?'Buffer: '.."
DEF K GOLD C AS "(C SEL W CHGCSR)."
DEF K GOLD F AS "(SEL PAR FILLSR)."
DEF K GOLD I AS "EXT INC ?'Input file: ' =?' Buffer: '."
DEF K GOLD L AS "EXT F L."
DEF K GOLD M AS "EXT F=MAIN.."
DEF K GOLD O AS "EXT WR ?'Output file: ' =?' Buffer: '."
DEF K GOLD Q AS "EXT QUIT/SAVE."
DEF K GOLD S AS "EXT SH BU."
!+
!       Set Terminal Characteristics
!-
SE WR 79
SE TR
SE K
SE M C
DELIMITERS_PROGRAMMING
!+
!       Set Buffer to MAIN, and Start
!-
F=MAIN
```

FIGURE 3. New Initializer File Listing (2 of 2)

[1,11] FASINI.EDT

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

LETTERS to the RSTS Pro . . .

. . . continued from page 6

I left Volume 3, No. 2, of the *RSTS Professional* on an airplane last week. Would it be possible for you to send me another copy of that issue?

D. Ross Porter
Pioneer Hi-Bred Internat'l, Inc.
Des Moines, Iowa

It's already sent, Ross. We appreciate your getting ***RSTS Professional*** *airborne. And we're sure the RSTS pro that picks up your copy will find it more enjoyable than the reading fare that is currently offered on planes. Thanks.*

—

After reading your December, 1981 issue, I was disturbed by the article "Benchmark Dibol vs. BASIC +2" by Frank Metcalf [p. 82]. Although I am sure Mr. Metcalf meant well, he has painted a grossly inaccurate picture. He seems to be comparing Dibol v4C with Basic +2. Dibol v4C does not support RMS file structures, as most languages did not at that time.

A more accurate comparison would have been Dibol v4D vs. Basic +2, or better yet, the new Dibol v4.5 and BP2 v1.6 using resident libraries. I believe Mr. Metcalf would be mildly surprised, to say the least. (I have seen some DEC internal documents showing that Dibol is the fastest and most compact language after Macro!?). Although we use Basic +, BP2, Dibol and Datatrieve in our shop, we are primarily a Dibol shop.

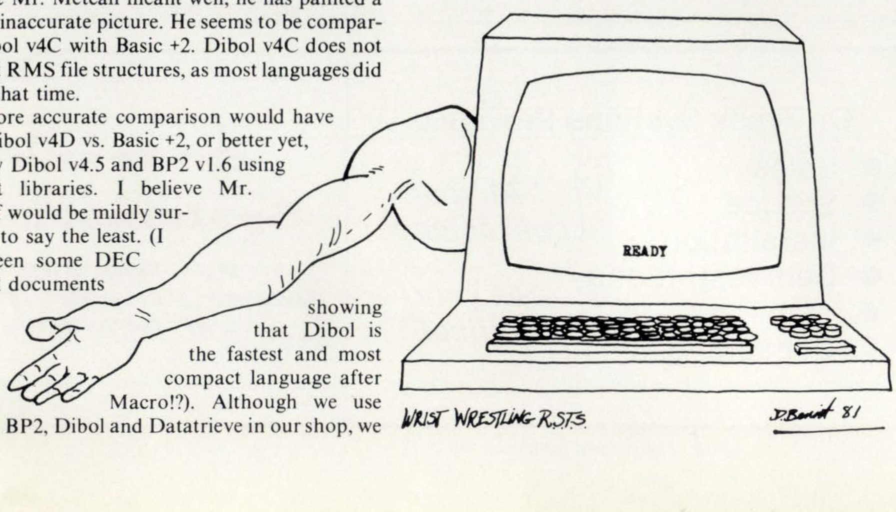
In short, I feel that the comparison done was inaccurate, all be it with good intentions. Dibol does have its deficiencies (i.e. no supported system call functions). Overall, it is a viable language that should be considered during application development.

Mike Milner
Director of Data Processing
City of Largo, Florida

P.S. The magazine is excellent; keep up the good work!

Thank you for your concerned comments, Mike. We like to view all sides.

—



I certainly appreciated Mr. George May's article *$REORDR - Sorting Alphabetically, (RSTS Professional, Vol. 3, No. 4).* I have his version of REORDR running on my machine now; however, I found a problem with line 1210 that must be corrected before installation.

Line 1210 should read:

```
1210    GOTO 1220 IF LEN(C$) = 0%                                     &
\       C$ = LEFT(CVT$$(C$,32%),3%)                                   &
\       GOTO 1220 IF C$ = 'CRE'                                       &
\       GOTO 1205 IF C$ <> 'ACC' AND C$ <> 'ALP'                      &
\       C3% = -1% IF C$ = 'ACC'                                       &
\       C3% = 1% IF C$ = 'ALP'                                        &
!       PRESET TO DEFAULT VALUE.  DONE IF DEFAULT INPUT, ELSE         &
!       GET FIRST THREE CHARACTERS IN UPPER-CASE ONLY.  DONE IF       &
!       'CRE'.  GET AGAIN IF NOT 'ACC' OR 'ALP'.  OTHERWISE,          &
!       SET TO ACCESS DATE SETTING                                    &
!
```

Thank you for a great publication!

Steve Young
Missouri Pacific Railroad Co.
St. Louis, MO

Thank you, Steve. We're glad we could help each other.

—

I enjoy your publication very much. Keep up the good work. Please provide more information concerning hardware failures, potential problems, etc. As a starter, an article on RSTS Error Logging (ERRDIS) would be great. DEC's System Manager Documentation is very inadequate along this line.

L. Dawson, Sr. Systems Specialist
E.I. du Pont de Nemours & Co.
Kinston, NC

O.K. L., we'll try to get an article on Error Logging into one of the next few issues. Keep watching.

—

Enclosed is payment for another fabulous year of RSTS Pro. I can't express the joy of finding a magazine of this type and caliber.

Thank you.
Mark Ruggiero

Thank YOU, Mark. Our readers make it possible.

SHOW

By Lawrence Fisher
Los Angeles Unified School District, Los Angeles, CA

SHOW is a program designed to display files on any video terminal that has TECO support on this system. It supports wildcard filespecs given in the normal format.

To run show, simply type SHOW <filespec>. No filespec will assume the argument of *.* (note: When displaying files that use Basic + lines, be certain to include a "/" at the end of the filespec so that TECO can convert the line structure to normal ASCII stream).

Show has one switch upon entry, that being "/s:<text>" where <text> is any phrase you might want the first screen to include.

The following is a summary of the commands that are available from within the program:

Command	Function
<SP>	Go forward one screen (note: this won't go forward an entire screen if there isn't enough text remaining on that page to complete an entire screen).
↑H	Go back one screen. Same as above.
↑L	Append next page of text. This reads in the next page of text after the end of the current page.
↑Y	Yank next line of text. This clears the text buffer before reading in the next page.
↑N	Go forward to next file. If you are using a wildcard filespec this will go to the next file in the sequence. If you are at the last file, it will exit the program. If you are not reading a wildcard file, you will return to the front of your file.
*	This prints out the name of the current file you are reading in the upper left hand corner of your screen.
<num>	Go to the line specified.
+<num>	Go forward the specified number of lines.
-<nurn>	Go backward the specified number of lines.
↑Z	Exit the program.

```
!   !!SHOW.TES!! !!V01.00!!      !          !  B$      Wildcard Filespec   !
!  Last edit 11-11-81 by LWF     !          !  D$      First half of spec  !
!                                !          !  E$      Second "    "    "  !
!    Copyright (C) 1981          !          !  N$      Get number Macro    !
!     Lawrence Fisher            !          !  S$      Search Argument     !
!                                !          !                              !
!  Options:                      !          !  1%      2/3 the screen      !
!                                !          !              Height          !
!    /S:          Search         !          !  3%      Last key typed      !
!                                !          !  4%      EOL for Term        !
!  Q-regs  used:                 !          !  9%      Screen Height       !
!                                !          !  N%      Numeric Arg var.    !
!  A$      Current Filename      !          !  S%      Search Arg Entered  !
! Enter the Get Number Macro !
@^UN%UN.UPJQNi$<^TUNQN"DQNI$|0;'>I $J\UN.+1,0KQPJQN%

! Get the Command line and proccess it out. !
Z"E G_$ '
J
:@S% %"S B,.D | :@FS%SHOW%% '
:@S%/S:%"L .-3,.D .,ZXS .,ZD -1US '
! strip all spaces & tabs and post fix extension (if none found) !
! I Borrowed part of this code from TYPE.TES !
HXA HXB 0,0XE 1U2 J :@S%"%"U 0,0XA 0,0XB J < @FS/^ES//; >
J < @S%:%; 0,.:XA 0,.K >
J :@S%[%"S .-1U9 @S%]% Q9,.:XA Q9,.K '
J :@S%(%"S .-1U9 @S%)% Q9,.:XA Q9,.K '
J :@S%<%"S .-1U9 @S%>% -1EJ-4"E Q9,.:XB | Q9,.:XA ' Q9,.K '
J :@S%/%"S R .,Z:XE .,ZK '
^"%U8 -1EJ-4"E ^^?U8
 J :@S%$%"S ^^$:@^UA%% -D '
 J :@S%&%"S ^"&:@^UA%% -D '
 J :@S%!%"S ^^!:@^UA%% -D '
 J :@S/%/"S ^^%:@^UA%% -D '
 J :@S%#%"S ^"#:@^UA%% -D '
 J :@S%@%"S ^^@:@^UA%% -D ' '
Z"E -1EJ-5"E @I%*.*% | @I%*.*% ' |
 J :@S%.%"F HXD ZJ @I%.*% -1EJ-5"E :QA"N :@ER%^EQD^EQE%"S -2D  ' ' ' |
  R ."E @I%*% ' ' '
J GA ZJ GB HXA
J :@S%^EU8%"S 0U2 %1^[ ' :@S%*%"S 0U2 %1^[ ' HXB Q2"E @EN/^EQB/ ' '

! Test for Scope support !
399ET$
ET&512"E ^A%No interactive scope available.
^A @O!STOP! '
```

```
! Attempt to open the file !
Q2"N @O!NOLOOK! ' :@EN//+1"G @^A%?Can't find file %:GA @^A%.
%  @O!STOP! ' HK G* HXB
 !NOLOOK! HK :@ER/^EQB^EQE/"U @^A%?Can't find file %:GA @^A%.
%  @O!STOP! '

! Set up the scope for support unique to it (EOL and Height) !
1:W-64"E 9U1 16U9 | 15U1 24U9 '
0:W-4"E 155^U4$ 91:^U4$48:^U4$ 75:^U4$
        | 0:W-10"E 155^U4$ 15:^U4$ | 155^U4$ 75:^U4$ ''

! Clear the screen, read in a page, and position the cursor either 2/3 !
! of the way down the screen, or at the item searched for.             !
-1W$
Y$
QS"L@_%^EQS% 0L |  Q1L$ '
-1W$

! Begin the actual command loop !
<^TU3$ ! Get a character !
Q3"D J Q3MNL -1W F< | ! Test for a numeric , if so, go to that line !
Q3-8"E -Q9L -1W$ F< | ! Test for a backspace, if so, go back a screen !
Q3-12"E A -1W$ F< | ! Test for a ^L (FF) if so, append a page !
Q3-14"E Q2"N @O!NOLOK! ' :@EN//; HK G* HXB !NOLOK! HK :@ER/^EQB^EQE/"U
 @O!END! ' -1W Y Q1L -1W | ! Test for a ^N, if so, go to next file !
Q3-23"E 0W$ -1W$ F< | ! Test for a ^W, if so, repaint screen !
Q3-25"E Y Q1L -1W$ F< | ! Test for a ^Y, if so, Yank in the next page !
Q3-26"E @O!END! | ! Test for ^Z to exit !
Q3-32"E Q9L -1W$ F< | ! Test for <SP> to go forward a screen !
Q3-42"E -1,5:W 6:W-1J -1W :G4 :G* $  ^T$ -2W Q1L 32768W$ -1W 0,5:W F< |
! Test for "*" - if so, print filespec in upper right hand corner, and !
! await the typing of any character !
Q3-43"E Q3MNL -1W F< | ! Test for + sign, for advancing x lines !
Q3-45"E Q3MNL -1W  ! Test for - sign for going back x lines !
'.''''''''' >

! Go to the bottom of the screen !
!END!
-1,5:W
6:WJ
23L
-1W
^A
^A

! And terminate all operations. !
!STOP!
^C^C
$$
```

♥

"INPUT LOOP" PROGRAMMING TECHNIQUE

By James F. Shaughnessy, Jr., Mirfanta Corp., New York

Synopsis: The "Input Loop" is a set of BASIC code to handle screen-oriented data entry applications. The same loop is executed for each entry field. The particular path through the loop is controlled by a mode variable and a set of parameters for each field.

Interactive entry and maintenance programs for data files are, obviously, one of the more common types of applications written for minicomputers. When I started to write these types of programs, I looked at examples (in BASIC) that had been supplied with the mini my firm had purchased and used them as a model for what I wanted to do. This model consisted of:

— a creation program to put a new record into the file.
— an inquiry program to look at records already in the file.
— a maintenance program to allow records in the file to be modified.
— a delete program to remove records from the file.

The create and modify programs were essentially the same, with the exception of code for identifying and retrieving the record from the file, and code for identifying the particular field(s) within the record to be changed. Another standard feature was that for each input field there was a section of code of perhaps 10 statements that was almost identical to sections of code for the other fields handled by the program.

I soon found the process of writing programs patterned after this model to be time-consuming, tedious, and exasperating. The euphoria of sucessfully debugging the "create" program quickly dissipated in the realization that I couldn't look at or change the records I had just created until I finished debugging the "inquiry" and "modify" programs! Another problem was that by being ambitious and trying to create a record with a lot of data fields, all handled by one program, I ran into memory overflows.

I started to find a solution to this set of frustrations when I realized that I could write a program with a single "INPUT" statement that would handle every field on the screen if I could come up with the proper set of parameters on which to branch around that input statement.

The technique which developed out of this centers on what I call the "Input Loop". The program flows continually through the same loop, branching on a "mode" variable, a field counter, and a set of parameters for each field. There are four primary "modes" of operation:

Create
Inquire
Modify
Delete

In addition, there are two secondary modes of operation:

change a single field while in primary Create mode
change a single field while in primary Modify mode

Each field is uniquely identified by an integer number (I%) and has associated with it an array of integer parameters (Z9%(I%,X%)). These parameters are used to identify:

0% — Screen row, if <0%, convert input to uppercase
1% — Starting column; if <0%, clear line before input
2% — Maximum input length, if <0%, RSET the input with spaces (unless overridden, the input will be automatically LSET with spaces to the maximum input length)
3% — Input Restrictor (absolute value):
 1% — accept anything up to the maximum input length
 2% — do not accept null
 3% — accept only maximum input length
 4% — accept only null or maximum input length
 5% — accept only "Y" or "N" (if <0%, do not pad the input)
4% — Pointer for edit routine
5% — Pointer to assign routine
6% — Pointer to element within array for assign and display routines (see below)
7% — Pointer to format and display routine
8% — Display length
9% — Help message number

Primary branching is to an Edit module, an Assignment module, a Format (& Display) module, and an Accept, Modify, Cancel module.

Let's look at the steps required for creating a new record:

The field counter is set to a starting value, and we start through the input loop.
Parameters are set according to the current field.
Data is input for that field.
Perform the edit function for that field. If the input doesn't pass, go back to the input statement.
Assign the value input to the proper file variable.
Format and display the file variable. (More on this later.)
Increment the field counter. If it's not greater than the ending value, go back to the top of the input loop.
Ask the user to Accept, Modify, Cancel (A/M/C) the record.
On Accept, put the record to the file and go back to the top of the input loop and reset the field counter for the next record.
On Cancel, just go back to the top of the input loop and reset the field counter.
On Modify, change the mode to 5%, ask the user to identify the field to modify, set the parameters for

that field, and go through the input loop once. Return here to prompt for another field to change.

If there are no more fields to change, go back to the Accept/Modify/Cancel prompt.

Now, for the variations, in the Inquiry mode:

Have the user identify the record by key and retrieve it from the file.

Display it by going through the input loop for each field, only executing the Format module.

In the Modify mode:

Identify, retrieve, and display the record as in the inquiry mode.

Change the mode to 6%, ask the user to identify the field to change, etc.

On Accept, update the record.

In the Delete mode:

Identify, retrieve, and display the record as in the inquiry mode.

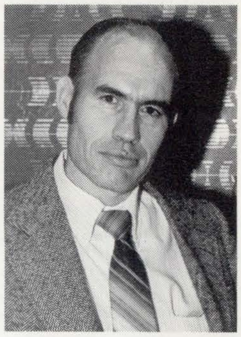
Ask the user to verify his/her intention to delete the record.

Delete it, if verified.

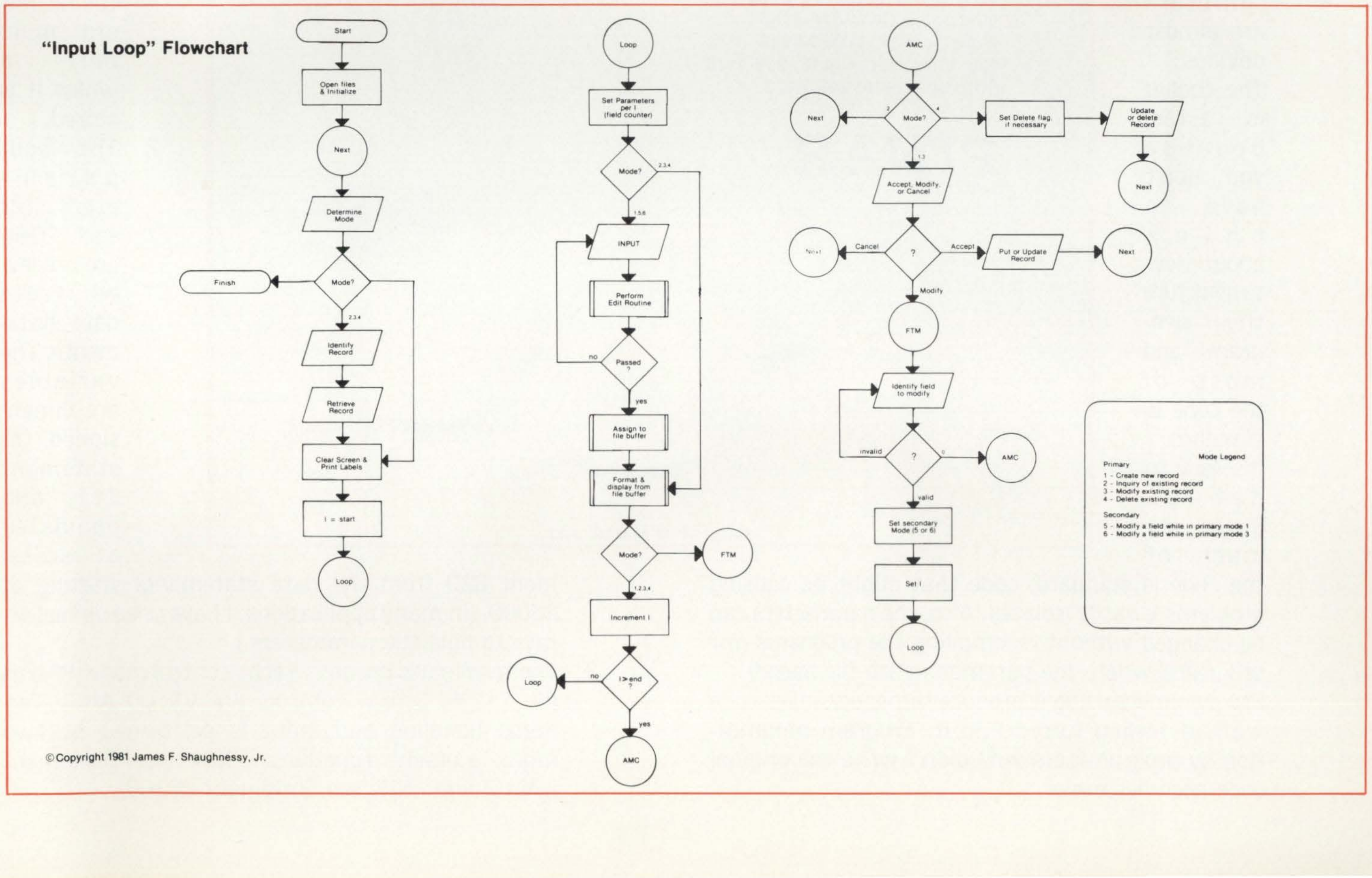
That's the essence of the technique. One enhancement used in some applications is to automatically record modifications to existing records in a log on a field-by-field, parameter-driven basis. Others include displaying records from different files on the screen at the same time, spreading a single record over multiple screens, and zero-balancing multiple records against a control figure before allowing any of them to be accepted.

Some of the "user-oriented" features include:

— back up a field by entering a "/"
— start over by entering a "*"
— reuse the value in the buffer by entering a "-"



"Input Loop" Flowchart



© Copyright 1981 James F. Shaughnessy, Jr.

- get a help message for any field by entering a "?"
- interprets a "4" as a "Y" and a "0" as an "N" in situations where only a yes/no response is acceptable. (This is a feature for operators who are primarily using the numeric keypad for data entry.)
- automatically uppercase the first letter each word (This is especially useful in mailing list applications, as data entry personnel don't have to use the shift key. Records can be upper/lower case for use in conjunction with word processing software.)

The technique leads to increased programmer productivity in several ways:

- Establish the record layout and the screen layout and the necessary parameters are almost obvious.
- The coding is faster, because you don't have to think about how to structure the program and most of the code is standard.
- It's easy to debug. Since so much of the code is standard, code that might be causing problems is easily isolated. Also, the parameters can be changed without recompiling the programs (for programs where the parameters are file-based).
- Programmers using it produce rather standard code without feeling forced into it. Program modification by programmers who didn't write the original code becomes easier.

The flowchart on page 81 summarizes the main points of the technique. The following program listing is a straightforward name and address application written in Basic-Plus, Extend. The first implemention of the technique in a DEC BASIC was in BASIC + 2, so this is in some respects a "downgraded" conversion.

Significant points about the program:

1. The output file is a Record I/O file. The first block contains a pointer to the next available block number for adding a new record (deleting a record does not free the block in which it was stored.). The user ID number for a record is one less than the file record number in which it is stored.
2. The input parameters for each field are retrieved from data statements. The variables are dimensioned at statement 711, and populated at statement 820 from the data statements starting at 25000. (In many applications, I have used virtual arrays to hold the parameters.)
3. The terminal is opened in echo control mode (8%) as file #12%. Screen controls are VT100 ANSI. Terminal handling and input is performed by two multistatement functions, FNS$ (see Statement 20010) and FNI$ (see Statement 20100).

In order to adapt the program for a different application, the following changes would have to be made:

Statement

711, 820 & 25000 The input parameters and help messages for each field must be changed according to the new application.

751 Field the output according to the new application.

810 & 5201 A new table string for the look-up of the field to be modified has to be devised.

821 Include new title for the screen.

851 Paint the screen with labels appropriate for the new application.

1011 Clear the screen for each new record in the most efficient manner for the application.

1102 & 1291 New start and end values for the input loop.

Subroutines

11000 Insert edit and verification statements appropriate for the new application.

12000 Insert assignment statements according to the new field statement.

13000 Insert format statements according to the new field statement. Formatted numeric output can be accomplished with a "PRINT #12% USING" statement instead of "PRINT #12%, X$;". (If you are working in BASIC-PLUS-2, it is very helpful to have FORMAT$(A,B$) in your repertory of frequently-used string functions.)

Author's Notes: The author makes no claims as to the suitability of the accompanying program listing for any purpose other than the demonstration of the programming technique described above. This article originally appeared in the August, 1981 edition of Computers-R-Digital.

[1,4] INPUT.LST

```
500     EXTEND
        !SA02.BAS
        !-----------------------------------------------------------!
        !                                                           !
        !       Demonstation of 'Input Loop' technique              !
        !                                                           !
        !       Copyright 1981 by James F. Shaughnessy, Jr.         !
        !                         New York, NY                      !
        !                                                           !
        !       No warranty is expressed or implied about the       !
        !       suitability of this program for any application     !
        !                                                           !
        !-----------------------------------------------------------!
        !                                                           !
        !       M.ODE% = 1%, Create new record                      !
        !              = 2%, Inquire on an existing record          !
        !              = 3%, Modify existing record                 !
        !              = 4%, Delete existing record                 !
        !                                                           !
        !              = 5%, Modify field while in M.ODE% = 1%      !
        !              = 6%, Modify field while in M.ODE% = 3%      !
        !                                                           !
        !-----------------------------------------------------------!
```

```
601     ON ERROR GOTO 23000

610     PRINT CHR$(155%) + "<";          ! Set VT100 to ANSI mode

700     !-----------------------------------------------------------!
        !                  DIMENSION STATEMENTS                     !
        !-----------------------------------------------------------!

710     DIM X9%(50%)
        ! Used in the CHANGE function

711     DIM     HELP$(16%),
              Z9%(16%,10%)
        ! Dimension statements for help messages and input parameters.
        ! See 25000 for data statements.  All prompts are hard-coded.

750     !-----------------------------------------------------------!
        !                      FILE OPENS                           !
        !-----------------------------------------------------------!

751     OPEN "JFS.DAT" FOR INPUT AS FILE #1%
      \ FIELD #1%, 2% AS SA.NEXT.RECORD$
      \ GET #1%, RECORD 1%
      \ NEXT.RECORD% = CVT$%(SA.NEXT.RECORD$)
      \ FIELD #1%,       2% AS SA.ID$,
                   4% AS SA.PREFIX$,
                 10% AS SA.FM.NAME$,
                  15% AS SA.L.NAME$,
                   30% AS SA.TITLE$,
                 30% AS SA.COMPANY$,
                    30% AS SA.ADDR$,
                    13% AS SA.CITY$,
                    2% AS SA.STATE$,
                      5% AS SA.ZIP$,
                   10% AS SA.PHONE$,
                2% AS SA.NEXT.DATE$,
                       24% AS FILL$,
               30% AS SA.ALT.CNTCT$,
                  1% AS SA.DELETED$
        ! Map statement for target file,
        ! First record of the file contains the next avialable
        ! record for a new record.
        ! The "ID Number ot a record is 1 less than the file record
        ! number where it is stored.  "ID Number" 00001 is stored in
        ! file record 2%, etc.

762     OPEN "KB:" AS FILE #12%, MODE 8%

800     !-----------------------------------------------------------!
        !                 INITIALIZE VARIABLES                      !
        !-----------------------------------------------------------!

810     F.IND$ = "YX.PR.FM.LA.TI.CO.AD.CI.ST.ZI.PH.NE.CR.LY.TY.AL."
```

```
820       ! Read help messages and input parameters from data statements

821       READ HELP$(J%) FOR J% = 1% TO 16%
     \    READ Z9%(J%,K%) FOR K% = 0% TO 10% FOR J% = 1% TO 16%

830       !Clear screen and print standard heading

831       PRINT #12%, FNS$("CE",1%,1%);
     \    PRINT #12%, FNS$("PC",1%,31%);"Mirfanta Corporation";
     \    PRINT #12%, FNS$("PC",2%,31%);"Name & Address File";

840       ! Inquire as to new mode.

841       PRINT #12%, FNS$("CE",3%,1%);
     \    PRINT #12%, FNS$("PC",23%,21%);
     \    PRINT #12%, "Create,Inquire,Modify,Delete (C/I/M/D)"
     \    PRINT #12%, "(<cr> to end)";

842       R% = -23%
     \    C% = 60%
     \    B9% = 0%
     \    L% = 1%
     \    M% = 0%
     \    X$ = FNI$(0%,1%)
     \    M.ODE% = INSTR(1%,"CIMD",X$)
     \    GOTO 32000 IF M.ODE%=0%

844       PRINT #12%, FNS$("CE",3%,35%);
     \    PRINT #12%, "Create "; IF M.ODE%=1%
     \    PRINT #12%, "Inquiry "; IF M.ODE%=2%
     \    PRINT #12%, "Modify "; IF M.ODE%=3%
     \    PRINT #12%, "Delete "; IF M.ODE%=4%
     \    PRINT #12%, "Mode";

850       ! Paint the screen
851       PRINT #12%, FNS$("PC",6%,16%);"ID Number";
     \    PRINT #12%, FNS$("PC",8%,13%);"Prfx-FM-Last";
     \    PRINT #12%, FNS$("PC",9%,20%);"Title";
     \    PRINT #12%, FNS$("PC",11%,18%);"Company";
     \    PRINT #12%, FNS$("PC",12%,18%);"Address";
     \    PRINT #12%, FNS$("PC",13%,14%);"City-St-Zip";
     \    PRINT #12%, FNS$("PC",15%,20%);"Phone";
     \    PRINT #12%, FNS$("PC",15%,41%);"Next Contact Date";
     \    PRINT #12%, FNS$("PC",17%,13%);"Credit Limit";
     \    PRINT #12%, FNS$("PC",18%,16%);"Sales: LY";
     \    PRINT #12%, FNS$("PC",19%,23%);"TY";
     \    PRINT #12%, FNS$("PC",21%,8%);"Alternate Contact";
     \    GOTO 1050

1000      !-------------------------------------------------------------!
          !                        PROGRAM LOGIC                        !
          !-------------------------------------------------------------!

1010      ! Clear the screen of old information
          ! In this application, all displayed information is displayed
          ! beginning in column 26 on rows 6,8,9,11,12,13,17,18,19 & 21,
          ! so we are just going to clear to end of line.
          ! Label at end of line 15 has to be reprinted.

1011      PRINT #12%,FNS$("CL",R%,26%); FOR R% = 6% TO 19%
     \    PRINT #12%,FNS$("CE",21%,26%);
     \    PRINT #12%,FNS$("PC",15%,41%);"Next Contact Date";

1050      GOTO 1200 IF M.ODE%>1%

1100      ! M.ODE%=1%, Create a new record
          ! The first field is the ID Number, which is system assigned.

1101      GOTO 24010 IF NEXT.RECORD%<0%
          ! In which case, the file is filled.

1102      Z9.START% = 2%
     \    Z9.END% = 16%
     \    GOTO 2000          ! The beginning of the input loop

1200      ! We are here because M.ODE%=2% OR M.ODE%=3% OR M.ODE%=4%
          ! We first have to retrieve the record which is to be worked with.
          !
          ! WARNING ...
          ! We are going to jump into and out of the input loop.
          !
1201      Z9.START% = 0%   ! in case we have to execute statement 2130
                           ! (entering a "/" or "#")
1202      I% = 1%
     \    GOTO 2030

1210      ! Return here from input loop
          ! Do what has to be done on the basis of the input,
          ! or else display the record retrieved.

1211      NUM.OF.CHANGES% = 0% IF M.ODE%=3%

1290       !The record retrieved will be formatted and displayed
1291      Z9.START% = 1%
     \    Z9.END% = 16%
     \    GOTO 2000


2000      !------------ S T A R T   O F   I N P U T   L O O P -----------!
          !

2010      FOR I% = Z9.START% TO Z9.END%

2020      GOTO 2400 IF M.ODE%=2% OR M.ODE%=3% OR M.ODE%=4%
          ! In any of these cases, we are just displaying a record which
          ! has been retreived from the file.

2030      R% = Z9%(I%,0%)

2031      C% = Z9%(I%,1%)
     \    IF       (M.ODE%=1% AND M%=3%)
                OR M.ODE%=3%
                OR M.ODE%>4%
          THEN  B9% = ABS(Z9%(I%,8%))
          ELSE  B9% = 0%

2100      GOTO 2450 IF Z9%(I%,2%)=0%
     \    L% = Z9%(I%,2%)
     \    P% = 0%
     \    M% = ABS(Z9%(I%,9%))
     \    A% = Z9%(I%,3%)
     \    X$ = FNI$(P%,A%)

2101      ON M% GOTO 2110,2120,2130,2200,2150      !X$="", M%=1%
                                                   !X$="*", M%=2%
                                                   !X$="/" OR "#", M%=3%
                                                   !X$=DATA, M%=4%
                                                   !X$="-", M%=5%

2110      ! Entered a null, M%=1%
2111      GOTO 841 IF I%=Z9.START%
     \    GOTO 841 IF M.ODE%=2% OR M.ODE%=3% OR M.ODE%=4%
     \    GOTO 2400 IF M.ODE%=5% OR M.ODE%=6%
     \    GOTO 2200

2120      ! Entered a "*", M%=2%, Start over
2121      GOTO 2400 IF M.ODE%=5% OR M.ODE%=6%
     \    GOTO 1010 IF I%>Z9.START% AND M.ODE%=1%
     \    GOTO 841

2130      ! A "/" or "#" was entered, M%=3%
          ! Back up to previous field.
2131      IF    M.ODE%<=4%
          THEN  GOTO 841 IF I%<=Z9.START% OR Z9.START%=0%
              \ I% = I% - 1%
              \ GOTO 2130 IF Z9%(I%,2%)=0%
              \ GOTO 2030
2132      GOTO 2400 !M.ODE%=5% OR M.ODE%=6%

2150      ! A "-" was entered, M%=5%
          ! Re-use the information in the buffer.
          ! Make sure that the field doesn't represent a key.
          ! Normally, all that has to be done is to go to
          ! statement 2400 to format and display the information.
          !
2151      GOTO 2030 IF Z9.START%=0%
     \    GOTO 2400

2200      ! Entered data, M%=4%
          ! Perform any necessary edit checks.
          ! Remember, all input is accepted in string form;
          ! therefore, in the edit routines, convert to integer
          ! or floating point, also.
          !
2201      GOSUB 11000
     \    IF    S%=0%
          THEN  ON M.ODE% GOTO 2300,
                               2400,
                               2400,
                               2400,
                               2300,
                               2300

2210      ! Input failed edit check
          ! Display help message if one is defined
          !
2211      GOTO 2250 IF Z9.START%=0%
```

```
     \  M% = ABS(Z9%(I%,9%))
     \  IF    M%=0%
        THEN  2250
        ELSE  PRINT #12%, FNS$("EL",24%,1%);
            \ PRINT #12%, HELP$(M%);

2250    ! Prompt that error occurred
        !
2251    PRINT #12%, FNS$("PC",23%,24%);
     \  PRINT #12%, "Input Error! - <cr> to continue";CHR$(7%);
     \  R% = 23%
     \  C% = 56%
     \  B9% = 0%
     \  L% = 1%
     \  M% = 0%
     \  X$ = FNI$(0%,1%)
     \  PRINT #12%, FNS$("CE",23%,1%);
     \  GOTO 2030

2300    ! Assign the input value to the file buffer
        !
2301    GOSUB 12000
     \  NUM.OF.CHANGES% = NUM.OF.CHANGES% + 1% IF M.ODE%=6%

2400    ! Call the subroutine to Format and Display the field,
        ! unless the virtual table specifies that the ouput length
        ! is zero (0%)
        !
2401    GOSUB 13000 IF Z9%(I%,8%)<>0%

2450    ON M.ODE% GOTO 2500,
                       2460,
                       2460,
                       2460,
                       5200,
                       5200

2460    ! This statement executes only
        ! IF M.ODE%=2% OR M.ODE%=3% OR M.ODE%=4%
        ! See statement 2450.
        ! If Z9.START%=0%, we are still in the
        ! process of getting the record.
        !
2461    GOTO 1210 IF Z9.START%=0%

2500    NEXT I%
        !
        !------------- E N D   O F   I N P U T   L O O P -------------!

2510    GOTO 5000

5000    !-----------------------------------------------------------!
        !                  ACCEPT/MODIFY/REJECT                     !
        !-----------------------------------------------------------!

5010    GOTO 5400 IF M.ODE% = 2%
     \  GOTO 5500 IF M.ODE% = 4%
     \  PRINT #12%, FNS$("CE",23%,15);FNS$("PC",23%,26%);
     \  PRINT #12%, "Accept,Modify,Cancel (A/M/C)";
     \  R% = -23%
     \  C% = 55%
     \  B9% = 0%
     \  L% = 1%
     \  M% = 0%
     \  X$ = FNI$(0%,3%)
     \  PRINT #12%, FNS$("CE",23%,1%);
     \  ON INSTR(1%,"AMC",X$)+1% GOTO 5010,5100,5200,5300

5100    ! Accept
5101    GOTO 5160 IF M.ODE%=3% OR M.ODE%=6%

5110    ! If accepting a new record,
        ! Do whatever has to be done to the system control file,
        ! perform whatever final processing has to be done,
        ! and "PUT" the record.
        !

5120    LSET SA.ID$ = CVT%$(NEXT.RECORD% - 1%)
     \  LSET SA.DELETED$ = "N"
     \  I% = 1%
     \  GOSUB 13000      ! Display the ID # assigned

5150    PUT #1%, RECORD NEXT.RECORD%
     \  NEXT.RECORD% = NEXT.RECORD% + 1%
     \  PRINT #12%, FNS$("CE",24%,1%);
     \  PRINT #12%, "Record created!"; CHR$(7%);
     \  SLEEP 3%
     \  M.ODE% = 1%
     \  GOTO 1010

5160    ! Update changes to an existing record.
        ! If NUM.OF.CHANGES%=0%, no changes have been made.
        !

5170    PUT #1%, RECORD GET.RECORD% IF NUM.OF.CHANGES%

5180    PRINT #12%, FNS$("CE",24%,1%);
     \  PRINT #12%, "No changes were entered!"; IF NUM.OF.CHANGES%=0%
     \  PRINT #12%, "Record Changed!";CHR$(7%); IF NUM.OF.CHANGES%
     \  SLEEP 3%
     \  M.ODE% = 3%
     \  GOTO 1010

5200    ! Modify a field.
5201    M.ODE% = 6% IF M.ODE% = 3%
     \  M.ODE% = 5% IF M.ODE%=1%
     \  PRINT #12%, FNS$("CE",23%,18%);
     \  PRINT #12%, "Enter first two letters of field to change";
     \  R% = -23%
     \  C% = 61%
     \  B9% = 0%
     \  L% = 2%
     \  M% = 0%
     \  X$ = FNI$(0%,4%)
     \  GOTO 5010 IF M%<>4%
     \  X$ = X$ + "."
     \  I% = INSTR(1%,F.IND$,X$)
     \  GOTO 5200 IF I%=0%
     \  I% = I%/3% + 1%          ! the divisor depends on the input length
                                 ! and the offset depends on Z9.START%
```

```
5210    ! Here is the place to reject any fields which are not allowed
        ! to be changed or to ask for a password.

5220    GOTO 5200 IF I%=1%

5240    GOTO 2030

5300    ! Cancel
5301    M.ODE% = 1% IF M.ODE% = 5%
     \  M.ODE% = 3% IF M.ODE% = 6%
     \  PRINT #12%, FNS$("CE",24%,1%);
     \  PRINT #12%, "No record created!";CHR$(7%); IF M.ODE%=1%
     \  PRINT #12%, "Record not changed!";CHR$(7%); IF M.ODE%=3%
     \  SLEEP 3%

5320    GOTO 1010

5400    ! Inquiry section.
        ! Have just completed displaying the requested record.
5401    PRINT #12%, FNS$("CE",23%,32%);
     \  PRINT #12%, "<cr> to continue";
     \  R% = -23%
     \  C% = 49%
     \  B9% = 0%
     \  L% = 1%
     \  M% = 0%
     \  X$ = FNI$(0%,1%)
     \  GOTO 1010

5500    ! Delete an existing record.
        ! Ask first, just to make sure.
        !
5501    PRINT #12%, FNS$("CE",23%,14%);
     \  PRINT #12%, "Are you sure you want to delete this record ? (Y/N)";

5510    R% = -23%
     \  C% = 66%
     \  B9% = 0%
     \  L% = 1%
     \  M% = 0%
     \  X$ = FNI$(0%,5%)
     \  ON M% GOTO 5510,32700,1010,5520,5510

5520    PRINT #12%, FNS$("CE",24%,1%);
     \  IF    X$="N"
        THEN  PRINT #12%, "Record not deleted!";CHR$(7%);
            \ GOTO 5550

5530    ! We execute this statement only if X$="Y"
5531    LSET SA.DELETED$ = "Y"
     \  PUT #1%, RECORD GET.RECORD%
     \  PRINT #12%, "Record Deleted!"; CHR$(7%);

5550    SLEEP 2%
     \  GOTO 1010
```

```
11000   !Edit input, if necessary
        !-------------------------------------------------------------!
        !       X$= the input string being edited                     !
        !       X%= integer representation returned, if appropriate   !
        !       X = floating point reprsentation returned,            !
        !           if appropriate                                    !
        !       I%= Z9%(I%,x) reference                               !
        !       S%= status, returned as 0% if X$ successfully passed  !
        !           all edit checks                                   !
        !-------------------------------------------------------------!

11010   S% = Z9%(I%,4%)
    \   IF    S%=0%
        THEN  11999
        ELSE  ON S% GOTO 11100,11200,11300,11400,11500,
                         11600,11700,11800,11210

11100   ! Put through capitalization routine

11101   X$ = FNCAP$(X$)
    \   S% = 0%
    \   GOTO 11999

11200   ! Check that 5 digits have been entered for zip code

11201   S% = Z9%(I%,2%)
    \   S% = FNDIGITS%(X$,S%)
    \   GOTO 11999

11210   ! Check that only digits have been entered in floating point fields

11212   S% = LEN(X$)
    \   S% = FNDIGITS%(X$,S%)
    \   X = VAL(X$)
    \   GOTO 11999

11300   ! Check for valid telephone number

11301   GOTO 11999 IF      LEN(X$)<>0%
                      AND  LEN(X$)<>4%
                      AND  LEN(X$)<>7%
                      AND  LEN(X$)<>10%
    \   IF    LEN(X$)=0%
        THEN  S% = 0%
            \ GOTO 11999

11310   X$ = LEFT(SA.PHONE$,6%) + X$ IF LEN(X$)=4%
    \   X$ = LEFT(SA.PHONE$,3%) + X$ IF LEN(X$)=7%
    \   GOTO 11999 IF LEN(X$)<>10%
    \   S% = Z9%(I%,2%)
    \   S% = FNDIGITS%(X$,S%)
    \   GOTO 11999

11400   ! Check prefix and edit it if necessary

11401   TEST$ = CVT$$(X$,32%+128%)
    \   X$ = "Mr." IF TEST$="MR"
    \   X$ = "Mrs." IF TEST$="MRS"
    \   X$ = "Ms." IF TEST$="MS"
    \   X$ = "Dr." IF TEST$="DR"
    \   X$ = "Mr." IF TEST$="" AND SA.SEX$="M"
    \   X$ = "Ms." IF TEST$="" AND SA.SEX$="F"
    \   GOTO 11100

11500   ! Check title for abbreviation and substitute

11501   TEST$ = CVT$$(X$,32%+128%)
    \   X$ = "Publisher" IF TEST$="PU"
    \   X$ = "President" IF TEST$="PR"
    \   X$ = "Publisher & Editor" IF TEST$="PE"
    \   X$ = "Editor & Publisher" IF TEST$="EP"
    \   X$ = "Editor" IF TEST$="ED"
    \   X$ = "Executive Vice-President" IF TEST$="EV"
    \   GOTO 11100

11600   ! Check city and substitute "New York" for "NY"

11601   TEST$ = CVT$$(X$,32%+128%)
    \   X$ = "New York" IF TEST$="NY"
    \   GOTO 11100

11700   ! Retrieve the specified record from the file

11702   GET.RECORD% = VAL(X$) + 1%
    \   IF    GET.RECORD%>=NEXT.RECORD% OR GET.RECORD%<2%
        THEN  PRINT #12%, FNS$("PC",24%,1%);
            \ PRINT #12%, "Record specified was never created!";
            \ GOTO 11999

11704   GET #1%, RECORD GET.RECORD%

11706   IF    SA.DELETED$="Y"
        THEN  PRINT #12%, FNS$("PC",24%,1%);
            \ PRINT #12%, "Record specified was previously deleted!";
            \ GOTO 11999

11708   S% = 0%
    \   GOTO 11999

11800  ! Put through date function

11802   IF    M%=1%
        THEN  X% = -1%
            \ S% = 0%
            \ GOTO 11999

11804   X% = FNDATE%(X$)
    \   S% = 0% IF X%>0%
    \   GOTO 11999

11999   RETURN

12000   ! Assign value to file variables
        !-------------------------------------------------------------!
        !                                                             !
        !       X$=value to assign to string fileds                   !
        !       X%=value to assign to integer fields                  !
        !       X =value to assign to floating point fields           !
        !       I%=pointer to proper row of array Z9%(I%,x%)          !
        !                                                             !
        !-------------------------------------------------------------!

12010   ON Z9%(I%,5%) GOTO 12110,12120,12130,12140,12150,
                           12160,12170,12180,12190,12200,
                           12210,12220,12230

12110   LSET SA.PREFIX$ = X$
    \   GOTO 12999

12120   LSET SA.FM.NAME$ = X$
    \   GOTO 12999

12130   LSET SA.L.NAME$ = X$
    \   GOTO 12999

12140   LSET SA.TITLE$ = X$
    \   GOTO 12999

12150   LSET SA.COMPANY$ = X$

    \   GOTO 12999

12160   LSET SA.ADDR$ = X$
    \   GOTO 12999

12170   LSET SA.CITY$ = X$
    \   GOTO 12999

12180   LSET SA.STATE$ = X$
    \   GOTO 12999

12190   LSET SA.ZIP$ = X$
    \   GOTO 12999

12200   LSET SA.PHONE$ = X$
    \   GOTO 12999

12210   LSET SA.NEXT.DATE$ = CVT%$(X%)
    \   GOTO 12999

12220   FIELD #1%, (153% + Z9%(I%,6%)*8%) AS FILL$,
                                       8% AS SA.FP$
    \   LSET SA.FP$ = CVTF$(X)
    \   GOTO 12999

12230   LSET SA.ALT.CNTCT$ = X$
    \   GOTO 12999

12999   RETURN

13000   ! Format and print
        !-------------------------------------------------------------!
        !            I%=Points to proper row of array Z9%(I%,x%)      !
        !-------------------------------------------------------------!

13010   R% = ABS(Z9%(I%,0%))
    \   C% = ABS(Z9%(I%,1%))
    \   PRINT #12%, FNS$("PC",R%,C%);
    \   ON Z9%(I%,7%) GOTO 13110,13120,13130,13140,13150,
                           13160,13170,13180,13190,13200,
                           13210,13220,13230,13240

13110   X% = CVT$%(SA.ID$)
    \   X$ = "     "
    \   RSET X$ = "00000" + NUM1$(X%)
    \   GOTO 13990

13120   X$ = SA.PREFIX$
    \   GOTO 13990

13130   X$ = SA.FM.NAME$
    \   GOTO 13990

13140   X$ = SA.L.NAME$
    \   GOTO 13990

13150   X$ = SA.TITLE$
    \   GOTO 13990

13160   X$ = SA.COMPANY$
    \   GOTO 13990

13170   X$ = SA.ADDR$
    \   GOTO 13990

13180   X$ = SA.CITY$
    \   GOTO 13990

13190   X$ = SA.STATE$
    \   GOTO 13990

13200   X$ = SA.ZIP$
    \   GOTO 13990

13210   X$ = MID(SA.PHONE$,1%,3%) + "/" + MID(SA.PHONE$,4%,3%)
             + "-" + MID(SA.PHONE$,7%,4%)
    \   X$ = SPACE$(12%) IF SA.PHONE$=""
    \   GOTO 13990

13220   X$ = DATE$(CVT$%(SA.NEXT.DATE$))
    \   GOTO 13990

13230   F$ = "  $$##,###" IF Z9%(I%,6%)=0%
    \   F$ = "$$,###,###" IF Z9%(I%,6%)
    \   FIELD #1%, (153% + Z9%(I%,6%)*8%) AS FILL$,
                                       8% AS SA.FP$
    \   PRINT #12% USING F$; CVT$F(SA.FP$);
    \   GOTO 13999

13240   X$ = SA.ALT.CNTCT$
    \   GOTO 13990

13990   PRINT #12%, X$;

13999   RETURN

20010   DEF FNS$(A$,R%,C%)
        !-------------------------------------------------------------!
        !                                                             !
        ! Returns string to perform certain VT100 ANSI screen functions !
        !                                                             !
        ! A$ =  "PC" means position cursor                            !
        !       "CS" means clear entire screen                        !
        !       "EL" means clear entire line                          !
        !       "CL" means clear from cursor to end of line            !
        !       "CE" means clear from cursor to end of screen          !
        !       "CB" means clear from beginning of line to cursor      !
        !       "CT" means clear from top of screen to cursor          !
        !                                                              !
        ! R% & C% are row and column positions, respectively           !
        !                                                              !
        !--------------------------------------------------------------!

20012   B$="PC.CS.EL.CL.CE.CB.CT"

20013   CS$=CHR$(155%)+"[" \ CUR$=NUM1$(R%)+";"+NUM1$(C%)+"H"

20014   ON INSTR(1%,B$,A$)/3% + 1% GOTO 20015,20016,20017,20018,
                                        20019,20020,20021
20015   FNS$ = CS$+CUR$ \ GOTO 20025
                ! position cursor

20016   PRINT #12%, CS$+"2J"; \ GOTO 20015
                ! clear entire screen then position cursor

20017   FNS$ = CS$+CUR$+CS$+"2K" \ GOTO 20025
                ! clear entire line

20018   FNS$ = CS$+CUR$+CS$+"K" \ GOTO 20025
                ! clear from cursor position to end of line

20019   FNS$ = CS$+CUR$+CS$+"0J" \ GOTO 20025
                ! clear from cursor position to end of screen

20020   FNS$ = CS$+CUR$+CS$+"1K" \ GOTO 20025
                ! clear from beginning of line to cursor

20021   FNS$ = CS$+CUR$+CS$+"1J"
                ! clear from beginning of screen to cursor

20025   FNEND

20100   DEF FN1$(PROMPT%,A%)

        !------------------------------------------------------------!
        !                                                            !
        ! To handle terminal input.                                  !
        ! This function returns input from the terminal after        !
        ! performing certain checks on it to ensure that it conforms !
        ! to predefined parameters.                                  !
        !                                                            !
        !       RET$ refers to the input from the terminal           !
        !                                                            !
        ! Arguments passed:                                          !
        !  PROMPT% = element of PROMPT$ array, if = 0%, disregard,   !
        !          if < 0%, RET$ is cleared after input              !
        !       A% = input allowed: if 1%, any input is allowed      !
        !                           if 2%, null input is not allowed !
        !                           if 3%, input length must = ABS(L%)
        !                           if 4%, input length must be 0%   !
        !                                  or ABS(L%)                !
        !                           if 5%, input must be "Y" or "N"  !
        !                                  "0" is converted to "N"
        !                                  "'" is converted to "N"   !
        !                                  "4" is converted to "Y"   !
        !                                  "\" is converted to "Y"   !
        !              if A% is negative, RET$ will not be padded    !
        !                                                            !
        ! Variables set before                                       !
        ! referencing the function:                                  !
        !                                                            !
        !       R% = row coordinate - if R%<0% then the input        !
        !          string will be converted to upper case            !
        !       C% = column coordinate - if C%<0% (that is, the column !
        !          coordinate is preceded by a negative sign (-)),   !
        !          the entire line will be cleared prior to          !
        !          printing the message                              !
        !      B9% = number of blanks to print to blank out old      !
        !          information if in maintenance mode                !
        !       L% = number of boxes (indicating length of input     !
        !          string - if L%<0% the input string will be RSET   !
        !          otherwise it will be LSET, unless A% < 0%,        !
        !          in which case RET$ will not be padded             !
        !       M% = help message # to use if requested              !
        !          on return:                                        !
        !                 M%=1% if the input string is "" (<cr>)     !
        !                 M%=2% if RET$="*"                          !
        !                 M%=3% if RET$="/" or RET$="#"              !
        !                 M%=5% if RET$="-"                          !
        !                 M%=4% if anything else (data) is input     !
        !                                                            !
        !       N.B.  A%<0% and L%<0% is an illogical combination    !
        !                                                            !
        !------------------------------------------------------------!


20101   FLAG% = 0%        ! is set if help message is requested

20105   R.ABS% = ABS(R%)
     \  C.ABS% = ABS(C%)
     \  BYTE.COUNT% = ABS(L%)
     \  A.ABS% = ABS(A%)
     \  P.ABS% = ABS(PROMPT%)
     \  IF    B9% <> 0%
        THEN  PRINT #12%, CHR$(155%);"[";NUM1$(R.ABS%);";";NUM1$(C.ABS%);"H";
                          STRING$(B9%,32%);

20110   MSG$ = ""
     \  BACK% = 0%
     \  PRINT #12%, CHR$(155%);"[";NUM1$(R.ABS%);";";NUM1$(C.ABS%);"H";
     \  PRINT #12%, CHR$(155%);"[2K"; IF C%<0%
     \  GOTO 20120 IF PROMPT% = 0%
     \  MSG1$ = PROMPT$(P.ABS%)

20115   MSG1$ = CVT$$(MSG1$,128%) + "  "
     \  BACK% = LEN(MSG1$)
     \  PRINT #12%,MSG1$;

20120   PRINT #12%,STRING$(BYTE.COUNT%,95%);
                   CHR$(155%);"[";NUM1$(R.ABS%);";";NUM1$(C.ABS%+BACK%);"H";

20130   PRINT #12%, RECORD 256%, CHR$(0%) + CHR$(BYTE.COUNT%) + CHR$(95%);

20135   GET #12%
     \  RE.COUNT% = RECOUNT
     \  FIELD #12%, RE.COUNT% AS RET$

20140   IF    R%<0%
        THEN  RET$ = CVT$$(RET$,4%+32%)
        ELSE  RET$ = CVT$$(RET$,4%)
```

```
20145   IF    RET$="!"
        THEN  PRINT CHR$(0%);
            \ GOTO 20105

20150   GOTO 20175 IF RET$="/" OR RET$="-" OR RET$="*" OR RET$="#"

20155   IF    RET$="?"
        THEN  PRINT #12%, CHR$(155%);"[24;1H";HELP$(M%); IF M%>0%
            \ PRINT #12%, CHR$(155%);"[24;1H";
                          "No help available for this field!";
                          IF M%=0%
            \ FLAG%=1%
            \ GOTO 20110

20160   TYP.COUNT% = LEN(RET$)

20165   GOTO 20175 IF A.ABS% = 1%
      \ GOTO 20110 IF (    ( A.ABS%=2% AND TYP.COUNT%=0% )
                        OR ( A.ABS%=3% AND TYP.COUNT%<BYTE.COUNT% ) )
      \ GOTO 20110 IF A.ABS%=4% AND  ( TYP.COUNT%<>0%
                                   AND TYP.COUNT%<>BYTE.COUNT%)

20170   GOTO 20175 IF A.ABS%<>5%
      \ RET$ = "N" IF RET$="'"
      \ RET$ = "N" IF RET$="0"
      \ RET$ = "Y" IF RET$="\"
      \ RET$ = "Y" IF RET$="4"
      \ GOTO 20110 IF RET$<>"Y" AND RET$<>"N"
      \ PRINT #12%,CHR$(8%) + RET$;

20175   IF    PROMPT%<0%
        THEN  PRINT #12%, CHR$(155%);"[";NUM1$(R.ABS%);";";NUM1$(C.ABS%);"H";
                          SPACE$(BACK%+BYTE.COUNT%);

20180   M%=4%
      \ M%=1%  IF TYP.COUNT%=0%
      \ M%=2%  IF RET$="*"
      \ M%=3%  IF (RET$="/" OR RET$="#")
      \ M%=5%  IF RET$ = "-"

20185   GOTO 20195 IF A%<0%
      \ BLANK$=SPACE$(BYTE.COUNT%-TYP.COUNT%)
      \ IF    L%<0%
        THEN  RET$=BLANK$+RET$
        ELSE  RET$=RET$+BLANK$

20190   GOTO 20195

20195   IF    FLAG%
        THEN  PRINT #12%, CHR$(155%);"[24;1H";CHR$(155%);"[2K";
            \ FLAG%=0%

20199   FNI$ = RET$
      \ FNEND

20400   DEF FNCAP$(IN.PUT$)

        !-------------------------------------------------------------!
        !                                                             !
        ! The purpose of this function is to capitalize the first     !
        ! letter of each word of IN.PUT$.  This is useful in          !
        ! name & address applications, because it allows the          !
        ! operator to input without regard to the shift key.          !
        !                                                             !
        ! The maximum length of the input string is 50 characters.    !
        ! .                                                           !
        ! If the first letter of the string is not to be              !
        ! capitalized, or if the first letter of a word               !
        ! within the string is not to be capitalized, it              !
        ! should be preceded by a "\" (ASCII=92%).  If                !
        ! this character appears as the first letter of               !
        ! the string, it is dropped and every other character         !
        ! in the string is shifted one position to the left,          !
        ! and a space is appended as the final character.  If         !
        ! the character is imbedded in the string, it is changed      !
        ! to a space.  In both cases, no change is made to the        !
        ! character following the "\".                                !
        !                                                             !
        !------------------------------------------------------------ !

20420   CHANGE IN.PUT$ TO X9%

20430   X9%(1%) = X9%(1%) - 32% IF    X9%(1%)>96%
                                  AND X9%(1%)<123%

20431   IF    X9%(1%)=92%
        THEN  X9%(J%) = X9%(J%+1%) FOR J% = 1% TO X9%(0%)
            \ X9%(X9%(0%)) = 32%

20440   FOR J% = 2% TO X9%(0%)

20441   IF    X9%(J%)=92%
        THEN  X9%(J%) = 32%
            \ J% = J% + 2%
```

```
20442   X9%(J%) = X9%(J%) - 32% IF    X9%(J%)>96%
                                  AND X9%(J%)<123%
                                  AND X9%(J%-1%)<48%

20443   NEXT J%

20450   CHANGE X9% TO IN.PUT$

20499   FNCAP$ = IN.PUT$
      \ FNEND

20500   DEF FNDIGITS%(IN.PUT$,S%)

        !-------------------------------------------------------------!
        !                                                             !
        !  The purpose of the this function is to verify that·        !
        !  IN.PUT$ is a comprised of a given number of digits,        !
        !  with no other characters present.                          !
        !                                                             !
        !  IN.PUT$ - the string to be tested.                         !
        !                                                             !
        !  S% - the number of digits expected                         !
        !                                                             !
        !  FNDIGITS%: 0% if IN.PUT$ passed the test,                  !
        !             S% if IN.PUT$ failed the test                   !
        !                                                             !
        !-------------------------------------------------------------!

20520   CHANGE IN.PUT$ TO X9%
      \ GOTO 20570 IF S%<>X9%(0%)

20530   FOR J% = 1% TO X9%(0%)

20540   IF    X9%(J%)<48% OR  X9%(J%)>57%
        THEN  FNDIGITS% = S%
            \ GOTO 20570

20550   NEXT J%

20560   FNDIGITS% = 0%

20570   FNEND

20600   DEF FNDATE%(X$)

        !-------------------------------------------------------------!
        !                                                             !
        ! The purpose of this function is to translate a date string  !
        ! of the format MM/DD/YY to the corresponding RSTS integer    !
        ! value.  If the incoming string cannot be successfully       !
        ! translated, the function returns a value of -1%.            !
        !                                                             !
        !-------------------------------------------------------------!

20610   MO$ = LEFT(X$,2%)
      \ DA$ = MID(X$,4%,2%)
      \ YR$ = RIGHT(X$,7%)
      \ X$ = MO$ + DA$ + YR$
      \ GOTO 20690 IF FNDIGITS%(X$,6%)

20620   MO% = VAL(MO$)
      \ GOTO 20690 IF MO%<1% OR MO%>12%

20630   YR% = VAL(YR$)
      \ YR% = YR% + 100% IF YR%<3%
      \ GOTO 20690 IF YR%<70%
      \ LEAP% = 0%
      \ LEAP% = 1% IF YR%=YR%/4%*4%
20640   DA% = VAL(DA$)
      \ DA.LIMIT% = 31%
      \ DA.LIMIT% = 30% IF MO%=4% OR MO%=6% OR MO%=9% OR MO%=11%
      \ DA.LIMIT% = 28% + LEAP% IF MO%=2%

20642   GOTO 20690 IF DA%<1% OR DA%>DA.LIMIT%

20650   FDOM$ = "000031059090120151181212243273304334"
      \ X% = (YR%-70%)*1000% + VAL(MID(FDOM$,MO%*3%-2%,3%)) + DA%
      \ X% = X% + LEAP% IF MO%>2%

20660   FNDATE% = X%
      \ GOTO 20699

20690   FNDATE% = -1%
20699   FNEND

23000   !------------------------------------------------------------!
        !                       ERROR HANDLING                       !
        !------------------------------------------------------------!

23010   IF    ERR=19%
        THEN  SLEEP 1%
            \ RESUME

23020   IF    ERR=5% AND ERL=751%
        THEN  OPEN "JFS.DAT" FOR OUTPUT AS FILE #1%
            \ FIELD #1%, 2% AS SA.NEXT.RECORD$
            \ LSET SA.NEXT.RECORD$ = CVT%$(2%)
            \ PUT #1%, RECORD 1%
            \ CLOSE #1%
            \ RESUME

23030   RESUME 11999 IF ERR=52% AND ERL=11701%

23990   PRINT
      \ PRINT "Error"; ERR; "at line"; ERL
      \ CLOSE #I% FOR I% = 1% TO 12%
      \ ON ERROR GOTO 0

24000   !------------------------------------------------------------!
        !                 SPECIAL MESSAGES TO OPERATOR               !
        !------------------------------------------------------------!

24010   PRINT #12%, FNS$("EL",24%,1%);
      \ PRINT #12%, "File is full. Please see software manager!";CHR$(7%);
      \ SLEEP 3%
      \ GOTO 841

25000   !------------------------------------------------------------!
        !                       DATA STATEMENTS                      !
        !------------------------------------------------------------!

25010   ! Help messages
```

* * * * * INTRODUCING* * * * *

The New HYPERBUS Driver!

No PDP-11 should be without one!

Thanks to the latest in Black Hole technology, we are able to offer for the first time, the HYPERBUS driver and interface. With this you'll never have to wait on memory management again! For the first time you'll never have to worry about the number of slots on the bus when considering expansion!

OUR PATENTED MEMORY SYSTEM RUNS FASTER THAN THOSE SUPER-CONDUCTOR SYSTEMS COULD POSSIBLY TOUCH!

"But how can that be?" you may right now be asking. Simple — we have combined the original concepts of high speed memory with our special enhanced rotating black hole generators, thus producing a local field distortion in the space-time continueum (WARNING: special care must be taken when working in the immediate area of this bus due to the possibility of general and special relativity distortions).

Naturally, there are certain drawbacks to this new media (such as forcing you to be on a solid floor, due to the additional weight added to your system (somewhere in the nature of 23 tons), but the space savings are incredible), but we feel that this problem will be resolved shortly (we are currently developing an improved version that uses advanced field manipulation to remove the excess weight incurred by the HYPERBUS).

Send now for additional information on the HYPERBUS to:

Incredible Systems, Inc.
1777773 Trap Row
New Los Angeles, Luna

♥

```
25011   DATA  "Enter prefix: Mr., Mrs., Ms., Miss, etc.",
              "Enter first & middle names and/or initials",
              "Enter last name",
              "Enter person's title (see documentation for abbreviations)",
              "Enter Company Name",
              "Enter mailing address",
              "Enter City (use 13 character Post Office format)",
              "Enter State abbreviation",
              "Enter Zip Code (5 digits required)",
              "Enter phone number, including area code (10 digits required)",
              "Enter record ID Number (Range: 00001-32766)",
              'Enter next contact date in format "MM/DD/YY"',
              'Enter credit limit (0 implies unlimited credit!)',
              'Enter sales amount for last year',
              'Enter sales-to-date for the current year',
              'Enter person to contact when primary contact is not available'

25020   ! Input Parameters
               !-------Input-------                   Ary       Dspl
               ! Row  Col  Len Rstr Edit Asgn  Ele  Fmt  Len Help Xtra
               !---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ----
25021   DATA        6,  26,   5,  -1,   7,   0,   0,   1,   5,  11,   0,
                    8,  26,   4,   1,   4,   1,   0,   2,   4,   1,   0,
                    8,  31,  10,   1,   1,   2,   0,   3,  10,   2,   0,
                    8,  42,  15,   1,   1,   3,   0,   4,  15,   3,   0,
                    9,  26,  30,   1,   5,   4,   0,   5,  30,   4,   0,
                   11,  26,  30,   1,   1,   5,   0,   6,  30,   5,   0,
                   12,  26,  30,   1,   1,   6,   0,   7,  30,   6,   0,
                   13,  26,  13,   1,   6,   7,   0,   8,  13,   7,   0,
                  -13,  41,   2,   3,   0,   8,   0,   9,   2,   8,   0,
                   13,  45,   5,   3,   2,   9,   0,  10,   5,   9,   0,
                   15,  26,  10,  -1,   3,  10,   0,  11,  12,  10,   0,
                   15,  59,   8,   4,   8,  11,   0,  12,   9,  12,   0,
                   17,  26,   6,  -1,   9,  12,   0,  13,  10,  13,   0,
                   18,  26,   7,  -1,   9,  12,   1,  13,  10,  14,   0,
                   19,  26,   7,  -1,   9,  12,   2,  13,  10,  15,   0,
                   21,  26,  30,   1,   1,  13,   0,  14,  30,  16,   0

32000   !----------------------------------------------------------------!
        !                        EXIT  HANDLER                           !
        !----------------------------------------------------------------!

32700   ! Clear screen
32701   PRINT #12%, FNS$("CE",1%,1%)

32710   ! Save next record number
32711   FIELD #1%, 2% AS SA.NEXT.RECORD$
      \ LSET SA.NEXT.RECORD$ = CVT%$(NEXT.RECORD%)
      \ PUT #1%, RECORD 1%

32760   CLOSE #J% FOR J% = 1% TO 12%
      \ ! If you are going to chain somewhere, do it here.

32767   END
```

♥

THE RSTS/E SYSTEM MANAGER

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

Last issue I mentioned that there appeared to be a problem with the TU77 tape drive showing **both** EOT and BOT conditions at the same time (with the expected operational problems from this situation). This condition has now been identified and (at my site) rectified, but let's take this opportunity to look at the chain of events and how RSTS/E significantly aided the problem resolution.

Several months ago I began to **occasionally** see a situation where, during BACKUP, the tape would "hang" at the EOT marker shuttling back and forth. The error log indicated that **both** EOT and BOT were set (you have to decode the error registers for this information). Field Service called in and they spent some time readjusting and finally replacing the EOT/BOT sensor on the drive. The problem appeared to be resolved for about a week, at which point it reappeared, but this time with the tape shuttling at the **beginning** of the tape!

Field Service again worked with readjusting and replacing the EOT/BOT sensor but the problem began to occur with increasing frequency towards the point where I was losing more BACKUPs than were successful. Of course, their diagnostics would not reproduce the problem, so they were limited to the information available once the BACKUPs had failed, and herein lay part of the problem: The Branch did not have the ability to monitor the internal states of the drive when it failed, and due to the intermittent nature of the problem, it was not escalated to the District and Regional levels (where such equipment and personnel **is** available) until the problem had assumed "major" significance to our operation.

At this point, the error logs indicated the following scenario: BACKUP would be proceeding as normal, showing an acceptable number of retrys during the "write" phase of a tape. Towards the end of the reel (as determined by comparing the time of the error entry and the BACKUP message on the console printer indicating that the "write" phase was "out of room" on that volume as well as the "record number" counter kept in the drive's DDB (as displayed in the error log)), the drive's implicit "read-after-write" check of the data it had just written indicated that there was a "bad" spot on the tape. The MM Driver software in RSTS/E correctly initiated the industry standard recovery procedure where the drive backspaces over the "bad" record it had just written, writes a long interrecord gap over the "bad" spot on the tape, and attempts to re-write the record (now slightly farther into the tape). In this case, this procedure continued several times (which is OK), but, having begun near the EOT marker, **continued its recovery operation PAST the EOT marker**, where it finally found a "good" spot of tape and successfully wrote the record. The next entry in the error log, about 4.5 minutes after the last "write" error, was an indication of "Operation Incomplete," and (by decoding the registers in the error log with the aid of the "Peripherals Handbook") both EOT **and** BOT were set. This condition would continue to generate hundreds of identical errors (with the DDB's Record Number indicating that it **thought** that it was still at the end of the reel!) until the drive was taken Off-line.

Now for the detective work: There was no argument that it was an illegal condition for EOT **and** BOT to be set at the same time, and a monitoring of the actual sensors indicated that they were set correctly. Additionally, examination of a failing tape indicated that the EOT and BOT reflective strips were placed within tolerance. So, where was the error coming from? The TM03's manual indicated that the Operation Incomplete error was **correct** when the drive was at BOT and doing a Space Reverse operation (also indicated by decoding more error log registers). OK, then why was the drive attempting to Space Reverse from the **beginning** of the tape, and why was EOT set when we were obviously at BOT?

The next key was that 4.5 minutes during which the tape "rewound," plus the fact that the DDB in the error log indicated that it still believed itself at the **end** of the reel: The TU77 manual indicates that rewind time is around 2.5 minutes, and **this** "rewind" was taking around twice that long. It turns out, however, that at 125 inches per second (the Space Reverse speed of this drive) it would **take just over 4 minutes to Space Reverse, rather than Rewind, the entire tape!** In fact, that is just what was happening . . . the drive, while recovering from some bad tape near the EOT marker, **passed** the EOT marker (which **stays** set until it is reset by one of several conditions, including a Rewind command or backspacing over the EOT marker), finally successfully wrote the record, noticed that it was at EOT, Space Reversed to get "before" the EOT marker (so that it could do its end-of-volume work), **but still found itself at EOT** (EOT was still set) so it Space Reversed again, **but again still found itself at EOT** so Space Reversed again, and indeed kept this up until it got to the beginning of the tape and encountered BOT. At this point, it **still** had to do a Space Reverse (because EOT was set), but this was no longer successful because that operation at **BOT** is illegal and (correctly) generated the Operation Incomplete error. This error kept occurring because once the Operation Incomplete error occurred, the original operation (Space Reverse) was retried because EOT was still set, with the same results.

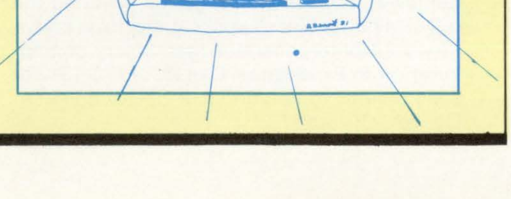
Note that the problem is **now** defined: All of the symptoms can be the direct result of EOT appearing to be set when it shouldn't be! Now, of course, the question is "hardware or software." While the MM driver was examined to see if it was misreading the EOT bit or was not re-reading it each time, District and Regional Field Service brought in a Logic Analyzer and probed various points within the drive. Additionally, based upon the theory that this series of events was triggered only during a retry after encountering a "bad" spot near the EOT marker, a test tape was prepared with EOT placed near the beginning of the tape (so that tests wouldn't take a long time) and a scratch was made on the tape near EOT. Sure enough, this would consistently cause the problem and the Logic Analyzer was able to verify that the EOT indicator was **not** being cleared (hence hardware and not software)!

All of the pieces have now fallen into place, except **why** the EOT indication was not being reset when the drive Space Reversed back over the EOT marker. The Field Service personnel rapidly found a section in the tape drive's logic which, in effect, **prevented** EOT from being cleared when Space Reversing back over the marker! They then devised a modification which defeated this logic, and the drive operated **perfectly** on the test tape and (so far) for all subsequent BACKUPs! (It should be noted that this field modification to the TU77 may not be the final "fix" which comes out, but **has** proven quite successful at my site.)

What was the point of going through this detailed description of the quest for my too prevelant EOT problem? There are several:

- RSTS/E error logs are **extremely** valuable to the System Manager and should be printed on a periodic basis **before** you've exceeded the "100 error limit" for any logged area;
- You should retain these "full" listings for as long as feasible to allow you to examine previous invocations of a problem at a later time (in this case, my old error log allowed me to quickly verify that all of these conditions were initiated during a "retry" near EOT once the evidence pointed in this direction and saved additional investigation of other potential causes);
- Familiarity with your hardware (not at the component level but at the "conceptual" level) and their manuals (remember the key of the 4.5 minute "rewind"?) can significantly aid you in determining the actual cause of a problem;
- Familiarity with the error log printouts can aid you in providing Field Service with the most pertinent information relating to your problem: remember that not all Field Service engineers are familiar with the error logs from all operating systems;
- Tracking the historical performance of all of your systems' components can indicate trends and provide valuable perspective on "new" problems, and can help you to get escalation on truly "long-term intermittent" problems (I have a set of programs which operate on all of my systems and, using DECnet, provide me with a comprehensive database of detailed information on all errors which occur, and which can be queried via Datatrieve for information such as "PRINT ALL ERRORS WITH DEVICE EQ "MM" AND DESCRIPTION CONTAINING "EOT" AND DESCRIPTION CONTAINING "BOT" AND DESCRIPTION CONTAINING "OPERATION INCOMPLETE" SORTED BY SYSTEM");
- Examine **all** potential sources of information relating to a problem . . . in this case the Console Terminal (KB0:) provided significant information in the form of OPSER's time/date stamp for the BACKUP messages;
- Attempt to examine **all** of the evidence to determine if a slippery problem has **changed** . . . in this case the sensor was apparently misadjusted which caused the first problems (with the tape shuttling at **EOT**) while the **actual** problem left the tape shuttling at **BOT**, and this important difference took a while to filter down from the Operations folks.

In general, even a "small" RSTS/E system is a complex mix of hardware and software which has the ability to "mask" problems and make them appear in ways which do not directly point to their origin. The error logs, handbooks, and manuals constitute a wealth of information (tools) which can aid you in ferreting out just where the problem is occurring, and can greatly reduce the time required for its resolution. You will have a "better managed" system which will yield increased "System Uptime," the bottom line. See you next issue. ♥



CLASSIFIED

Send Classified Ads to: RSTS Classified, P.O. Box 361, Ft. Washington, PA 19034-0361. $1^{00} per word, first 12 words free with one year's subscription. [Be sure to include a phone number or address in your message.]

PROGRAM Development Aids. This British software will cut your costs. Waxride, London, 01-636-5963.

FOR software assistance, call Ron at 895-5011 or Bob at 895-5613.

CORPANE Industries Inc. (502-491-4433) manufactures Ultrasonic Vapor Degreasers. See our advertisement in the Thomas Register.

Author of UNCOMP/UNCMP2, call Mark at (617) 646-2473. Evenings EST.

Just how many [1,*] passwords do you use that are catchy and easy to remember??
Lots of Luck from LOCK-11 (p. 67).

RSTS/E PROFESSIONALS

Positions now open in exciting WALL STREET projects.

-TOP MONEY-
-SUBCONTRACTORS WELCOMED-

Send resume or call:
LINK SOFTWARE INC.
P.O. Box 208, Wall Street Station, New York, 10268
(212) 426-8921

SAVE YOUR quarters, games for RSTS/E are here. Interactive, real-time games for VT52 and VT100 terminals. Some of the games available:

BLKADE: As many as eight players try to force each other to collide with their growing tails on the screen.

STRWRS: You have five minutes to destroy the Death Star with your single X-wing fighter.

SUBS: Two players manuever around islands in an attempt to sink the other player's submarine.

Games come on 9-track magtape with their own high-quality user manuals. Order the first game for $39.95, $29.95 for each additional game. Individual manuals available for $9.95 each.

Send your check, or write or call:

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

RSTS CHEAP!

11/70, 24 Hours, 7 Days

$5/hour connect +
$20/MB/MO......or
$350/port/mo. unlimited

B+, B+2, DIBOL, etc.
WORD-11, TECO, ED2

NATIONWIDE DATA DIALOG
Dave Mallery
215-364-2800

DEC BEST VALUES

PRE-OWNED DEC EQUIPMENT

BUYING AND SELLING

SYSTEMS • CPU's • PERIPHERALS • TERMINALS
OPTIONS • MEMORY • COMPATIBLES

CALL DICK BAKER (305) 979-2844

dataware incorporated
Carico Center
2845 NW 62nd Street
Ft. Lauderdale, Florida 33309
Telephone (305) 979-2844

GROSSMAN GRAPHICS
215
537-0782
5041
frankford
phila., pa.
19124

RSTS RESCUE SQUAD

We salvage all kinds of disasters:

- unreadable disks
- ruined UFDs and MFDs repaired
- immediate response
- telephone DIAL-UP
- on-site
- software tools
- custom recovery
- 90% success to date
- more than 1 GB rescued to date

Brought to you by
On Track Systems, Inc.
and a well known (and read)
Disk Directory expert.
CALL 24 HOURS
215–542-7133

PROGRAMMERS

Large Denver based service bureau serving an international clientele, is looking for programmers to fill positions within its equipment dealer services on-line processing division. Two years experience required with BASIC+2 DEC hardware and RSTS/E operating systems. Requires a desire to work with large and complex applications programs covering the full array of business applications from inventory control to general ledger. Salary comensurate with experience. Call (303) 773-1313 or send resume to Personnel, P.O. Box 5606 TA, Denver, CO 80217.

The *FAMOUS*
RSTS PROFESSIONAL
TEE-SHIRT
is now for sale!

Send size desired and $6.95
for each shirt to:
RSTS TEE-SHIRT
P.O. Box 361
Ft. Washington, PA 19034-0361
Shirts available in adults sizes only:
Small - Medium - Large - X-Large

LOOKING FOR DEVELOPMENT TIME?

$7 PER HOUR CONNECT TIME

NO KILOCORE TICK CHARGES
NO CPU CHARGES
RSTS/E TIME

BASIC PLUS
BASIC PLUS, PASCAL, "C" } WITH CROSS COMPILER SUPPORT
WORD-II WORD PROCESSING
WAFE, TECO, EDT } PROGRAM EDITING

BUDGET BYTES
212-944-9230

OMNICOMPUTER, INC.
1430 Broadway, New York, N.Y. 10018

IMMEDIATE DELIVERY!!!

New 11/34 32KB Memory	$ 285
MUX Boards	$1250
11/44 256KB	$3875
11/34 256KB	$2475
128KB	$1975
11/34 Floating Point	$2190
VAX 256KB	$4900
11/70 512KB	$8950

Ray (617) 275-6800

WE put the ARTS in RSTS. Robert W. Hunt & Assoc., 424 Skinner Bldg., Seattle, WA 98101, (206) 625-9582.

PASS Corp., Garden Grove, CA. "The Perfect 10" in computer services. 714-636-2576.

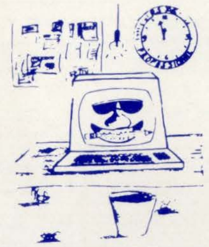
PLAN PLUS FINANCIAL PLANNING SYSTEM

RSTS/E, $4,500.

10 Barley Mow Passage
London, W4 4PH
Tel. UK 01 994 6477
Telex 8811418.

. . . continued on page 95

NEWS RELEASES



Occasionally we are requested to print news that may be of interest to the RSTS community. We are happy to offer this feature to our readers. We reserve the right to print only as time and space permit. We cannot return photos or manuscripts. Send news releases to: RSTS News Release, P.O. Box 361, Ft. Washington, PA 19034-0361.

February, 1982

WHY SYSTEMS, INC. ANNOUNCES DIGICALC VERSION 1.1

Redmond, WA — WHY Systems Incorporated is announcing verison 1.1 with a significationly larger spreadsheet capacity, rapid worksheet save and recall, multiple worksheet consolidation, enhanced formatting and data display, and data editing. The built-in training procedures, on line HELP, and reference manual have all been revised and enhanced. Worksheet file structures are now documented so that users may create utilities to interface their existing files with DIGICALC. DIGICALC runs on RSTS/E, RSX-11M, VMS and TOPS-10/20.

DIGICALC is the electronic spreadsheet for DEC computers developed and marketed by WHY Systems Inc. By the end of December over 50 copies were installed in such diverse areas as banking, accounting, budgeting, manufacturing, engineering, utilities, and corporate research and development centers. Version 1.1 is being distributed to original customers and is ready for shipment to new customers.

Improvements to DIGICALC include a larger spreadsheet capacity. A user many now designate the size worksheet desired up to a maximum of 600 columns by 500 rows. Even with the larger size, worksheets can be rapidly saved and recalled with version 1.1's new binary worksheet files.

Formatting options have been expanded to include number of decimal places displayed, optional dollar sign display, protected fields, and ability to designate several different formats for a single cell, for example: display three decimal places, left-justify and protect against unintentional data alteration or erasure. Another improvement allows the user to edit data and equations character by character.

DIGICALC

VERSION 1.1
Effective January 1, 1982

	LICENSES	
OPERATING SYSTEM	FIRST CPU	2ND-10TH CPU
RSTS/E	$4000	$2800
RSX-11/M	4000	2800
VAX/VMS	6000	4200

Each License includes: Magtape distribution, 1 reference manual, 1 Laminated Keypad Reference Card and 30 day warranty.

Each additional Reference Manual including 1 Laminated Keypad Reference Card	$35.00
Each additional Laminated Keypad Reference Card . .	6.00
Alternative Distribution: (Applies to maintenance updates also)	
RLO2 Disk Pack	$200.00

TERMS:

Purchase orders invoiced net 10, 1½% per month charge on balance due over 30 days.

5% Cash discount for payment accompanying order.

Washington State customers add 6.6% state sales tax.

Shipment will be made only upon receipt of completed and approved software license agreement.

DEMONSTRATIONS:

1. A direct dial-up demonstration is available. Complete information for its use is included on a separate page.

2. Customers may obtain a thirty day money back guarantee if they enclose full payment with their order and signed license agreement and specify their desire for the guarantee. If within thirty days after shipment, they notify WSI they are not satisfied with and do not wish to have DIGICALC, the agreement will be terminated in accordance with its terms and conditions and full payment, less $200 for shipping and handling costs, shall be refunded.

All prices and terms subject to change without notice.

Worksheet consolidation allows a multiple number of worksheets having identical form but different input data to be aggregated into one master worksheet. This may be used when different departments of a company each using the same report form submit reports needing to be summarized. Individual cell data in one worksheet may be used in another worksheet simply by referencing that worksheet and cell.

DIGICALC is available for dial-up demonstration.

One of DIGICALC's users is creating a utility program to interface VisiCalc and DIGICALC worksheet files. When it is ready, WHY Systems will make it available at no cost to DIGICALC customers.

Contact: WHY Systems Inc., 17130 Avondale Way NE, Suite 118, Redmond, WA 98052. Telephone: (206) 881-2331.

November, 1981

USERS GET POOR MAN'S DECNET(R)

Bedford, MA — Clyde Digital Systems, a leading international supplier of stand alone utility software, reports that its new CALL-11 package is expected to find wide acceptance as an inexpensive alternative to DECNET. Released early this year, the product is now installed at many sites across the U.S. and Canada with sales increasing rapidly.

According to the spokesman at Clyde Digital Systems, no special communication hardware is required. CALL-11 is run by the user at any terminal connected to the host system. A standard communication port is used to call up a second computer over ordinary business telephone lines with an existing modem or accoustic coupler.

In addition, communication between two local computers can be achieved by a simple cable, making a direct connection between a standard terminal port on each computer.

Files of any type can be transferred between the two computers with CALL-11. This includes binary program files together with their file attributes. Files can be transferred in either direction. The correctness of the information transferred is assured independent of telephone line noise. Another feature not found in DECNET is the ability to do wildcard transfers with a file name inspection latch.

The spokesman from Clyde Digital emphasized that the CALL-11 package need only reside on the computer that initiates the connection. This frees the user from a limited network

cost of a license for each correspondent computer together with the cost of special hardware. This is in direct contrast with DECNET which requires the added cost of a license for ech correspondent computer.

It is also noted that CALL-11 does not require a system generation. Indeed there is no installation required. It is particularly easy to use. A new user can become a confident user of CALL-11 in 10 to 20 minutes.

CALL-11 is currently implemented on the DEC PDP-11 and VAX series of computers under the RSTS and VMS operating systems. The correspondent system may be a computer other than a PDP-11 or VAX.

Clyde Digital Systems offers an unusual warranty. It is unconditionally guaranteed to work perfectly. No maintenance contract is needed. CALL-11 may be delivered by dial up line to a new user's computer within 24 hours of receipt of order. Further information can be obtained from Bill Keefe at Clyde Digital Systems, Inc., P.O. Box 348, Bedford, MA 01730. Tel: (617) 275-2924.

(R)Registered trademark of Digital Equipment Corporation.

December, 1982

NEW SOFTWARE PRODUCT FOR VAX/VMS USERS

Denver, CO — A new software product for DEC VAX/VMS users is now available from Interactive Software Systems Inc.

ADT-11 is an applications development tool which provides a structured framework from which a new application system can be designed, programmed, documented, and easily maintained. The heart of ADT-11 is a fully integrated, totally available data dictionary that provides a centralized, unique description of all data files, records, data elements, and on-line application screens. A BASIC program generator is available to build new application programs from a standard program layout, which insures program uniformity throughout the application system. The program generator also provides an interface to the data dictionary so that record descriptions and screen layouts are included in each application program as needed.

ADT-11 provides data access utilities which can perform data entry and maintenance (add, change, delete), general inquiry (select and extract), and a full function report generation for any data file that has been described in the data dictionary.

The non-technical user will find that ADT-11 is a friendly system. Activities are selected from a series of menus, all of which have "HELP" messages to provide guidance when needed. The consistency of operation within ADT-11 is also comforting. Certain applications may even be designed and maintained without assistance from the data processing department. For example, one-time reports that generally are placed at a very low priority by busy data processing departments can be user-defined and generated.

ADT-11 is a multi-facted software package which can save a data processing staff a great deal of time and effort during applications systems development. In addition, ADT-11 also provides the kind of structured framework which can ease the job of maintaining and enhancing the application system.

For further information, send inquiries to: Interactive Software Systems Inc., P.O. Box 3314, Englewood, CO 80155. Tel. (303) 534-2847.

December, 1981

DIGITAL MANAGEMENT GROUP INTRODUCES NEW NETWORKING SYSTEM FOR RSTS/E

Ontario, Canada — Digital management Group Ltd., a DEC-oriented software and consulting company, has recently announced the introduction of DMG/NET, a new network communication system for RSTS/E users.

"DMG/NET evolved out of a project for one of our clients, Labatt Brewing Company Limited", says John Dightam, President of DMG. "Their need was for a communication network that would provide low cost interactive access from any of Labatt's Head Office terminals to a wide variety of DEC and non-DEC computers throughout Canada and the United States. Labatt also needed low cost file transfer capability between their seven RSTS/E systems across Canada. We were asked to provide all this, with the mandatory specification that no modifications be made to either the standard RSTS/E system or PDP-11 hardware as supported by DEC."

Clearly Labatt's problems with RSTS/E as a network host were not unique, and DMG/NET was developed as a solution to the communication problems of any RSTS/E system. In addition to the Datapac interface capability, DMG/NET interfaces with Telenet and overseas X.25 networks. It provides two-way file transfer and interactive dialogue capabilities between a local RSTS/E host and other RSTS/E systems. Communication with IBM and other non-RSTS/E systems is also possible but must be initiated by the local RSTS/E host.

"I believe that DMG/NET is truly an effective, innovative answer to some of today's complex technology issues," states Bill Jarvis, Director of Information Services for Labatt. "It has allowed us to integrate our national distributed processing network, to access outside data processing and office systems facilities, and to take maximum advantage of common carrier networks - all at an amazingly low cost."

The "alternate path" capability of DMG/NET allows all available alternatives for network connection to be explored, including all outgoing ports, remote access numbers and leased lines, if these facilities are available on the computer. If a connection cannot be completed because all lines are in use, or because of a fault in the network or communications unit, DMG/NET will describe the situation in an easily understood message displayed on the user's screen.

According to John Kowalyshyn, Labatt's Manager of Technical Services, the need was for a trouble-free network based on Datapac. "We could see the tremendous cost advantages of X.25, but what we wanted was a reliable user interface that required only short "plain English" commands. Now, anyone with a RSTS-connected terminal on our London Head Office computer facilities can access all our own computers - plus a whole variety of service bureau computers in both Canada and the States. The RSTS-to-RSTS file transfer even handles compiled code - a tremendous help to our software maintenance in the remote sites."

Through table-driven software, DMG/NET provides total interface and network flexibility. For example, as new micro-computers become available and are interfaced with the RSTS/E host, DMG/NET can handle the dialogue requirement between the two central processors.

"DMG/NET has been operating on our PDP-11/70 for almost a year now," says Mr. Lynn LaRouche, Manager of Data Processing for Labatt. "The packet costs are amazingly low and we are very happy with the reliability. When a network problem does occur, the DMG/NET diagnostic routines help us pinpoint the problem quickly. I also like the statistics and "log" facilities since these really allow me to manage the day-to-day traffic on the network."

Bill Jarvis sums it up this way: "The DMG/NET concept has proven to be a major advance in providing friendly tools for productivity improvements in both Labatt's office and data processing environments".

DMG/NET is available on either a perpetual license or a rental basis - both with comprehensive support plans. "We are really committed to support," says DMG President Dightam, "and we have designed DMG/NET so that it lends itself to remote diagnosis. From our Toronto office, we can reconfigure, diagnose faults, and load new versions into customer computers anywhere in the world - just as if they were in the next office. Essentially, it's this kind of remote communication that X.25 and DMG/NET is all about!"

For more information on DMG/NET, contact Ken Allsopp, Digital Management Group Ltd., 4800 Yonge Street, Suite 208, Willowdale, Ontario M2N 6G5, Tel: (416) 225-7788.

December, 1981

SOFTWARE TECHNIQUES ANNOUNCES FAST RSTS/E DIRECTORY "RE-ORDER" UTILITY

Los Alamitos, CA — Software Techniques Inc. today announced the release of RDR, a fast disk directory "re-order" utility for users of Digital Equipment Corp.'s RSTS/E V7.0 operating system. Periodic use of RDR significantly improves overall system performance by reducing the number of disk accesses required to find and retrieve files.

RDR is designed to replace the DEC-supplied REORDR program. Written in MACRO-11, RDR is not only 30 times faster than the DEC utility, but also incorporates many new features.

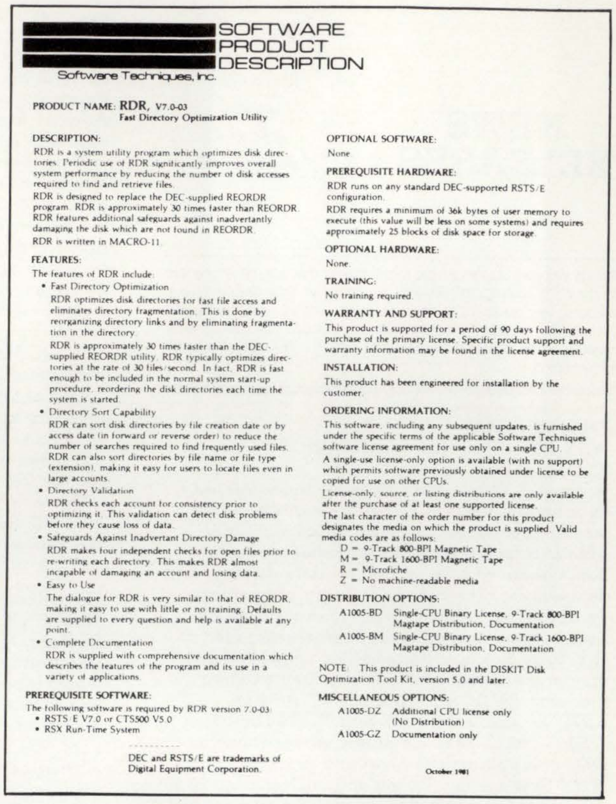
RDR's features include:

- Very high speed (re-ordering an RM02 disk in less than one minute)
- Reorders the disk MFD (Master File Directory), a RSTS/E first!
- Extended directory sort options (including file name, file type, access date, or creation date)
- Can reorder dismounted disks to prevent directory damage
- Comprehensive documentation (which includes a description of the RSTS/E directory structure)

(Please see product description for further details.)

RDR is available directly from Software Techniques, Inc. (Software Techniques, Ltd. in the U.K.) and from authorized distributors for $150 (U.S. cash price, quantity one). The **RDR User's Guide** is available alone for $30. OEM and quantity discounts are available.

Software Techniques, Inc., headquartered in Los Alamitos, CA, is one of the world's leading mini-computer consulting groups. Specializing in Digital's RSTS/E and VMS operating systems, Software Techniques provides products and services world-wide, ranging from business accounting software packages to high-technology consulting services.


SOFTWARE
PRODUCT
DESCRIPTION
Software Techniques, Inc.

PRODUCT NAME: **RDR,** V7.0-03
Fast Directory Optimization Utility

DESCRIPTION:

RDR is a system utility program which optimizes disk directories. Periodic use of RDR significantly improves overall system performance by reducing the number of disk accesses required to find and retrieve files.

RDR is designed to replace the DEC-supplied REORDR program. RDR is approximately 30 times faster than REORDR. RDR features additional safeguards against inadvertantly damaging the disk which are not found in REORDR.

RDR is written in MACRO-11.

FEATURES:

The features of RDR include:

- Fast Directory Optimization

  RDR optimizes disk directories for fast file access and eliminates directory fragmentation. This is done by reorganizing directory links and by eliminating fragmentation in the directory.

  RDR is approximately 30 times faster than the DEC-supplied REORDR utility. RDR typically optimizes directories at the rate of 30 files/second. In fact, RDR is fast enough to be included in the normal system start-up procedure, reordering the disk directories each time the system is started.
- Directory Sort Capability

  RDR can sort disk directories by file creation date or by access date (in forward or reverse order) to reduce the number of searches required to find frequently used files. RDR can also sort directories by file name or file type (extension), making it easy for users to locate files even in large accounts.
- Directory Validation

  RDR checks each account for consistency prior to optimizing it. This validation can detect disk problems before they cause loss of data.
- Safeguards Against Inadvertant Directory Damage

  RDR makes four independent checks for open files prior to re-writing each directory. This makes RDR almost incapable of damaging an account and losing data.
- Easy to Use

  The dialogue for RDR is very similar to that of REORDR, making it easy to use with little or no training. Defaults are supplied to every question and help is available at any point.
- Complete Documentation

  RDR is supplied with comprehensive documentation which describes the features of the program and its use in a variety of applications.

PREREQUISITE SOFTWARE:

The following software is required by RDR version 7.0-03:
- RSTS/E V7.0 or CTS500 V5.0
- RSX Run-Time System

OPTIONAL SOFTWARE:

None.

PREREQUISITE HARDWARE:

RDR runs on any standard DEC-supported RSTS/E configuration.

RDR requires a minimum of 36k bytes of user memory to execute (this value will be less on some systems) and requires approximately 25 blocks of disk space for storage.

OPTIONAL HARDWARE:

None.

TRAINING:

No training required.

WARRANTY AND SUPPORT:

This product is supported for a period of 90 days following the purchase of the primary license. Specific product support and warranty information may be found in the license agreement.

INSTALLATION:

This product has been engineered for installation by the customer.

ORDERING INFORMATION:

This software, including any subsequent updates, is furnished under the specific terms of the applicable Software Techniques software license agreement for use only on a single CPU.

A single-use license-only option is available (with no support) which permits software previously obtained under license to be copied for use on other CPUs.

License-only, source, or listing distributions are only available after the purchase of at least one supported license.

The last character of the order number for this product designates the media on which the product is supplied. Valid media codes are as follows:
D = 9-Track 800-BPI Magnetic Tape
M = 9-Track 1600-BPI Magnetic Tape
R = Microfiche
Z = No machine-readable media

DISTRIBUTION OPTIONS:

A1005-BD	Single-CPU Binary License, 9-Track 800-BPI Magtape Distribution, Documentation
A1005-BM	Single-CPU Binary License, 9-Track 1600-BPI Magtape Distribution, Documentation

NOTE: This product is included in the DISKIT Disk Optimization Tool Kit, version 5.0 and later.

MISCELLANEOUS OPTIONS:

A1005-DZ	Additional CPU license only (No Distribution)
A1005-GZ	Documentation only

DEC and RSTS/E are trademarks of Digital Equipment Corporation.

October 1981

November, 1981

SSI ENTERS DECsystem MARKET WITH SEVEN PRINTER SYSTEMS

Fort Lauderdale, FL — Seven line printer systems, each compatible with all models of the DECsystem 20 computers, have been introduced by Southern Systems, Inc. (SSI). Available speeds range from 300 to 1,200 lines per minute.

Introduction of the band-technology line printer systems by SSI was triggered by the printer specialist's development of the new S-20 line printer controller. The S-20 is designed to be completely hardware and software compatible with the DECsystem 2020, the 2040, the 2060 and 1091. SSI is the **only** supplier of add-on printers for DECsystems 10 and 20.

The S-20 line printer system is priced 35 to 55 percent less than DEC's comparable LP20 line printer system. The lower price of the SSI controller is combined with the economical pricing of SSI printers to provide DEC users with a cost-effective printer product suitable for expansion or upgrading existing DECsystems or installation with new DECsystems. SSI offers installation and continuing maintenance for all its line printer systems.

The S-20 is a single board controller which requires only a singel SPC slot (small, peripheral controller slot) within the DECsystem 20 processor. This gives SSI printer systems equipped with the S-20 the distinct advantage of requiring only one-sixth of the interfacing space within DECsystems. As a result, DEC users have the flexibility of operating up to eight printers on one DECsystem 20.

Southern Systems S-20 is a microprocessor design using a Z-80. Since it is a single board, it also eliminates the need for a special backplane or the unibuss cables required by DEC's LP20 controller.

With S-20-equipped printer systems all circuitry is included to perform the following operations: programmable character set, horizontal tab features, DAVFU of Paper Tape VFU, delimeter features, auto line feed on line overflow and DMA transfers.

The S-20 controller operates in the following four modes: RAM load, DAVFU load, print and test. The RAM load mode allows the character set to be fully progammable. Character translation is performed by the S-20 where required. The DAVFU load mode allows the printer DAVFU, if applicable, to be loaded from the DECsystem 20's main memory. The printer mode uses DMA transfers of DECsystem 20 memory data to the line printer system. Translation of data is made when required. The test mode allows the S-20 to be tested without accessing the printer.

With the new printer system products, Souther Systems continues to follow its policy of guaranteed compatibility of printer systems with processors.

Southern Systems is located at 2841 Cypress Creek Road, Fort Lauderdale, Fla. 33309; (305) 979-1000; telex 522135; (800) 327-5602.

October, 1981

SOFTWARE PACKAGE TRANSFERS FILES BETWEEN TWO DEC MINICOMPUTERS WITHOUT USING COMMUNICATIONS INTERFACES.

Suitable for LSI-11's and VAX-11's as well as PDP-11's, including processors with different operating systems and different storage media. London, England — A low-cost and easy-to-implement means of transferring files (data and/or programs) between two Digital Equipment Corp. PDP-11, LSI-11 or VAX-11 computers, is making its debut in the U.S. Called XOREN IPL-11, it is a software package which enables file-transfers to be carried out asynchronously between any two machines from the three families, either on the same site or in different geographical locations.

Developed by Xoren Computing Ltd, of London, England, IPL-11 enables a two-way communications system to be set up for less than $3,000 (excluding line costs) and put into operation as soon as the program has been installed on each machine.

Transfers can take place in both directions simultaneously at rates up to 9600 bits/second, over a direct line or telephone line; and can be initiated manually or, in certain circumstances, under the control of a user's applications program.

The package carries out and monitors the entire transfer process. It performs CRC error-correction to CCITT Recommendation V41 and, when it detects errors, re-transmits the block or blocks in which the errors were found.

A key feature of the package is that it eliminates the need to buy dedicated communications interfaces. Transfers are via a terminal I/O port on each computer. Thus, for inter-processor communications between two computers on the same site, each machine requires only a standard DEC DL-11 or equivalent single-channel terminal interface card, or a single port on a multi-channel interface card such as a DZ-11 or DH-11.

For communications between computers on different sites, each processor's interface card requires only the addition of the necessary modem.

IPL-11 provides a number of operational advantages. It is media-independent, i.e. transfers data irrespective of the data storage devices in use on each system. It eliminates the need to transport disks and other media when transferring data on or off site, and it allows data to be quickly stored at a remote site for security.

The package also allows multi-system access to expensive peripherals such as large disk systems; rapid updating of data and/or programs on computers at remote sites; communications between word-processing and information-processing systems, and instant distribution of information in electronic mail applications.

Operating System Independent - Another attractive feature of IPL-11 is that it is largely operating-system independent. Versions are available for most major DEC operating systems, including RSX-11M, RT-11, VAX/VMS (RSX-11 compatibility mode), IAS and, most recently, RSTS/E. Each version can communicate with any other.

The version for RSX-11M supports indirect command files and has a Remote Activation facility which enables a user at one computer to transfer files to or from another without an operator being present at the other machine. An optional file-conversion utility, FLC-11, provides a one-pass conversion for applications where the two operating systems use different file formats.

A further option IPLLIB, enables transfer under the RSX-11M version, to be controlled by users applications programs. The package is supplied under a 5-year licence. A separate licence is required for each combination of cpu and operating system under which IPL-11 is to run. The two licences required to link two operating systems cost $1350 each whichever pair of operating systems is specifiec. For large orders a system of discounts is applied.

Xoren is currently setting up a network of distributors in the US and Canada to market the product to OEM's and end-users

Xoren Computing Ltd is an independent systems/software company formed in 1974. It has developed real-time computer systems and software for several large organisations in the UK, Europe and North America, including the British Post Office, I.T.T. and the British Columbia Telephone Company and has developed a number of communications-oriented software products.

For more information contact: Mr. John Jarvis, Xoren Computing Ltd., 28 Maddox St., London W1R 9PF, England. Tel. LONDON(01)6295932.

October, 1981

EGH ANNOUNCES RELEASE OF VERSION F10 OF FSORT3

Lexington, MA — Evans Griffiths & Hart, Inc. (EGH) announced the release of version F10 of FSORT3, their sort package that runs on both DEC's PDP-11 under RSTS/E and the VAX under VMS (using ROSS/V, another EGH software package, to provide a RSTS/E environment).

FSORT3 is a machine-language package for sorting unblocked and blocked files of fixed-length records. A representative of EGH said that, for speed, record files are sorted directly without the use of intermediate key files. She also added that EGH has run stand-alone timing tests of FSORT3 on a VAX-11/780 and found that 100,000 50-byte randomly generated records could be sorted in under 4.5 minutes of wall time with under 1.5 minutes of CPU time.

New to the F10 release is a merge utility that can be used either to merge or concatenate up to eleven input files into a single output file. The input files may be of different types from one another and from the output file. The merge utility can also be used to convert files from one type to another (blocked to unblocked, RMS to non-RMS, etc.) and to replace records in a master file with corresponding records from an update file.

The new FSORT3 release directly supports RMS sequential fixed-length-record files, both span and nospan. To support RMS files and virtual arrays, three new data types have been added: a PDP-11 internal-format integer (which can be used to represent RMS and virtual array integers), an unsigned PDP-11 internal-format integer, and a PDP-11 internal-format floating point number (which can be used to represent RMS and virtual array floating point numbers).

FSORT3 Version F10 is available for $2,500 for a single CPU license from Evans Griffiths & Hart, Inc., 55 Waltham St., Lexington, MA 02173. Tel: 617-861-0670.

November, 1981

DATA ENTRY SUB-SYSTEM (DES) FOR DATA BOSS/32

North Miami, FL — Florida Computer, Incorporated (FCI) released version 1.1 of Data Boss/32, a relational Data Base Management System (DBMS), for VAX/VMS systems.

Version 1.1 includes the Data Entry System (DES) as well as enhancements to Data Boss/32 and improved user documentation.

DES enables the user to create data entry formats for checking and verifying data prior to the update of a data base. DES features include range checks, valid items tables, DUP button, and default values, to mention only a few.

Data Boss/32 contains a powerful English-like query language and is implemented in native mode Fortran IV. Users can interface custom code together with Data Boss/32 utilities.

Data Boss, initially under RSTS/E, was introduced over five years ago by Florida Computer, Incorporated of North Miami, Florida. Modified and upgraded over the years it is in use by several hundred users worldwide including a number of OEM's and time-sharing services.

For more information call or write, Florida Computer, Inc., John H. Wright, 99 NW 183rd St., North Miami, FL 33169, 305-652-1710 or in Europe contact Turnkey Software, 12 High St., Chalfont St. Giles, Bucks, Canada HP8 4QA. ♥

CLASSIFIEDS

. . . continued from page 92

AUTHORS!!!

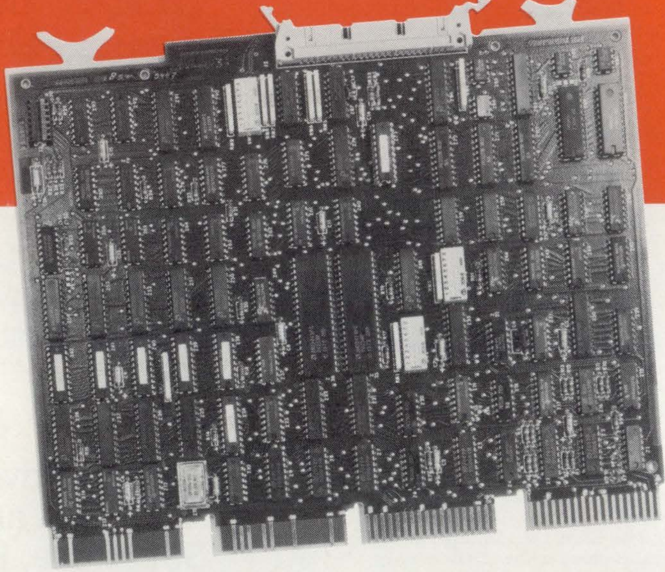
Send your articles of interest to the RSTS community to the ***RSTS Professional*** on mag tape in either RNO, PIP or WORD-11 format. Eighty percent of this issue was transmitted via tele-communications from author's mag tapes to phototypesetting equipment and was not retyped. ♥

LIST OF ADVERTISERS

Able Computer	I.B. Cover
Advanced Software Products	p.61
Amcor Computer Corp.	pp.77,79,81,83
Bob 'Macro Man' Meyer	p.27
Braegen MPD	p.31
Business Controls Corp.	p.50
C.D. Smith & Associates, Inc.	p.90
Clyde Digital Systems	p.2
ComDesign, Inc.	p.10
Data Processing Design, Inc.	B. Cover
Dataram Corp.	p.96
Dataware Inc.	p.87
Digital Associates Corp.	p.73
Digital Management group Ltd.	p.5
Digital Technologies Co.	p.14
Distributed Logic Corp.	p.33
EEC Systems	p.66
Emulex Corp.	pp.19,21
Enterprise Technology Corp.	pp.7,87
EPS, Inc.	p.89
Evans Griffiths & Hart, Inc.	pp.47,55
Expoconsul International, Inc.	p.25
Finar Systems Ltd.	p.82
Florida Computer, Inc.	p.15
Hamilton Rentals	p.9
Hinditron Computer Systems & Consultants Pvt. Ltd.	p.63
Infinity Software Corp.	p.75
Innovative Data Technology	p.43
Interactive Software Systems Inc.	p.87
Interactive Technology Inc.	p.36
McHugh, Freeman & Associates, Inc.	p.91
Monosson	p.29
National Peripherals, Inc.	p.11
Nationwide Data Dialog	pp.62,67,72,85
North County Computer Services, Inc.	pp.48-49
On-Track Systems, Inc.	pp.70,77
Plycom Services, Inc.	p.71
Raxco Inc.	p.17
Ross Systems, Inc.	I.F. Cover
RTZ Computer Services, Ltd.	p.65
Software Results Corp.	p.88
Software Techniques, Inc.	p.41
Southern Systems, Inc.	p.1
System Industries	p.37
Telecom Computer Systems, Inc.	p.23
T.F. Hudgins & Associates, Inc.	p.90
TransNet Corp.	p.44
Trendata Corp.	p.13
Unitronix Corp.	p.84
WHY Systems, Inc.	p.35
Xoren Computing, Ltd.	p.24

♥

DEC®-COMPATIBLE PERIPHERAL CONTROLLERS



Dataram Corporation offers the industry's widest range of DEC-compatible peripheral controllers — from comparatively simple NRZI tape controllers to complex 300 MB storage module drive (SMD) controllers.

An impressive array of state-of-the-art controllers, all built around high-speed bipolar microprocessors. All software compatible with the host LSI-11®, PDP®-11, or VAX® minicomputer...and all available now.

And Dataram's controllers are designed to save you money, and, more importantly, space — our controllers typically occupy half the space required for the comparable controller from DEC. Doing it with a level of performance that makes any member of this family worth looking at.

The chart shows our current family of peripheral controllers, growing every day. If you don't see the controller you need, we're probably working on it right now. Call us and discuss your requirements.

CONTROLLER	DESCRIPTION	COMPATIBILITY
C03	Cartridge disk controller	RK05
C33	Cartridge disk controller	RK05
T03	NRZI mag tape controller	TM11/TU10
T04/N	NRZI mag tape controller	TM11/TU10
T04/D	Dual density mag tape controller	TM11/TU10
T34/N	NRZI mag tape controller	TM11/TU10
T34/D	Dual density mag tape controller	TM11/TU10
T36	Dual density mag tape controller	TM11/TU10
S03/A	80MB/300MB SMD controller	RM02/RM05
S03/A1	160MB SMD controller	RM02
S03/B	80MB/300MB SMD controller	RK07
S03/C	200MB/300MB SMD controller	RP06
S03/D	96MB CMD controller	RK06
S33/A	80 MB/300 MB SMD controller	RM02/RM05
S33/A1	80 MB/160 MB SMD controller	RM02
S33/B	80 MB/300 MB SMD controller	RK07
S33/C	200 MB/300 MB SMD controller	RP06
S33/D	96 MB CMD controller	RK06

Products printed in red are LSI-11 Bus compatible.

Products printed in black are UNIBUS® compatible for PDP-11 and/or VAX minicomputers.

DEC, LSI-11, PDP, UNIBUS and VAX are registered trademarks of Digital Equipment Corporation.

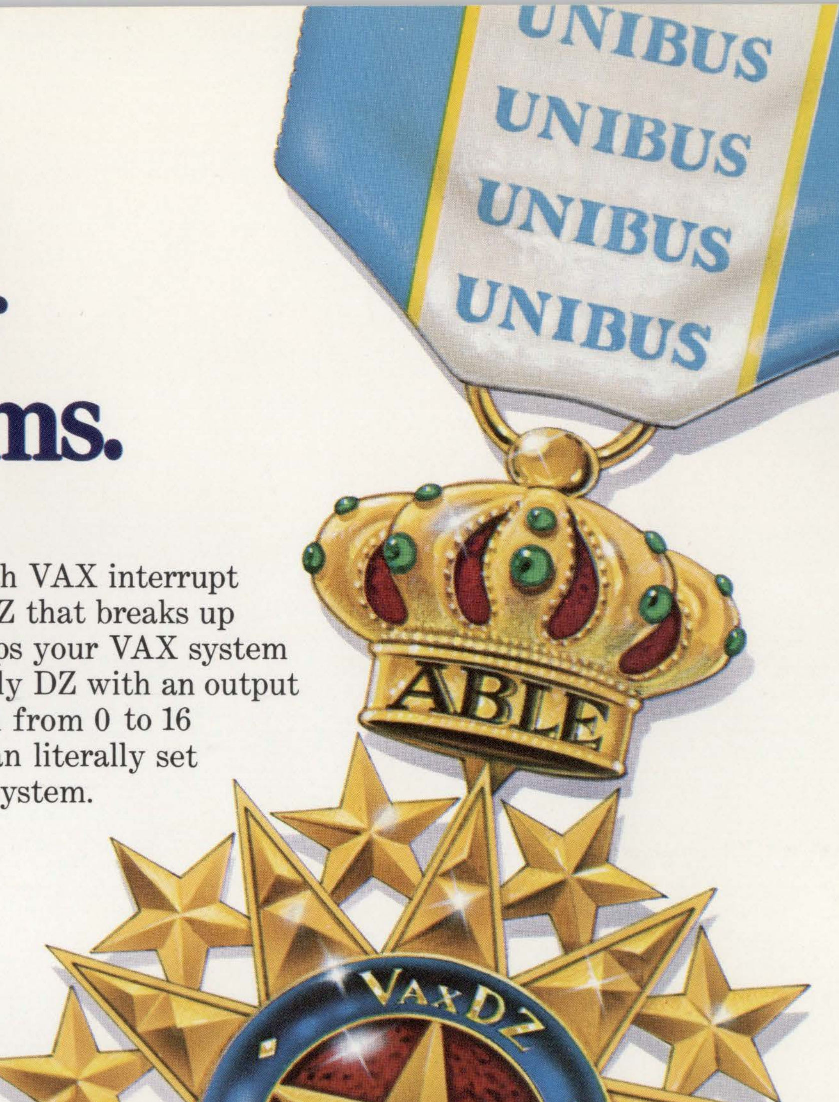
DATARAM
CORPORATION

Princeton Road
Cranbury, New Jersey 08512
Tel: 609-799-0071 TWX: 510-685-2542

Canada: Ahearn & Soper Ltd., 416-245-4848 • **Denmark:** Technitron ApS, 02 96 98 22 • **Finland:** Systek OY, (80) 73 72 33 • **France:** YREL, (03) 956 81 42 • **Hungary/Poland/Rumania:** Unitronex Corporation, WARSAW 39 6218 • **Italy:** ESE s.r.l., 02/607 3626 • **Netherlands:** Technitron b.v., (020) 45 87 55 • **Sweden:** M. Stenhardt AB, (08) 739 00 50 • **Switzerland:** ADCOMP AG, 01/730 48 48 • **United Kingdom:** Sintrom Ellinor Ltd., (0734) 85464 • **West Germany:** O.E.M.-Elektronik GmbH, 07 11-79 80 47 • **Yugoslavia:** Institut "Jozef Stefan", 263-261 • **Australia/New Zealand:** Anderson Digital Equipment, (03) 544-3444 • **India:** Infosystems Private Limited, 79281 • **Israel:** Minix Computers & Systems Ltd., 03-298783 • **Japan:** Matsushita Electric Trading Co., Ltd., 06 (282) 5111 • **Taiwan:** Rabbit Associates, Ltd., 7219573-5 • **Hong Kong:** Automated Systems (HK) Ltd., 5-630256-9 • **Malaysia:** Automated Systems (M) Sdn Bhd., 773777 • **Indonesia:** P. T. Daya ASL, 584306 • **Singapore:** Automated Systems (PTE) Ltd., 2354133

CIRCLE 55 ON READER CARD

ABLE VaxDZ clears up your data traffic jams.



You've been hurting over the problem with VAX interrupt capacity for some time. Now we have a DZ that breaks up the terminal handling bottleneck and keeps your VAX system in the fast lane. It's ABLE VaxDZ, the only DZ with an output buffer which lets you select any silo depth from 0 to 16 characters. With this novel feature, you can literally set the optimum performance level for your system.

That alone should clear up the traffic, but there's more! We've given ABLE VaxDZ an intelligent input silo two times as big as the standard DZ buffer. Both big and smart means doubling the VAX input data-handling capacity in some systems or providing dramatic improvement in every system all the way up to the maximum line configuration. We've even included a "data throttle" which allows any external device to control the clear-to-send (CTS) line and optimize its own data rate.

VaxDZ puts sixteen lines with modem control on a single hex-width board at one unit load and includes a panel which supports EIA only (an optional panel supports a mix of EIA/CL). Other features include an on-board LED display for pinpointing malfunctions automatically, an on-board self-test for immediate verification of system integrity and a variable PROM set for proprietary OEM applications.

Now, here is the best part. ABLE VaxDZ will match or beat DH performance in VAX systems without the addition of foreign software.

You don't have to be a hero to deserve an ABLE VaxDZ medal. Just be smart enough to use our new multiplexer. Write for details. We'll include information on the ABLE line of UNIBUS-compatible products, as well as the MAGNUM™ Series of computer systems.

ABLE
the computer experts

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

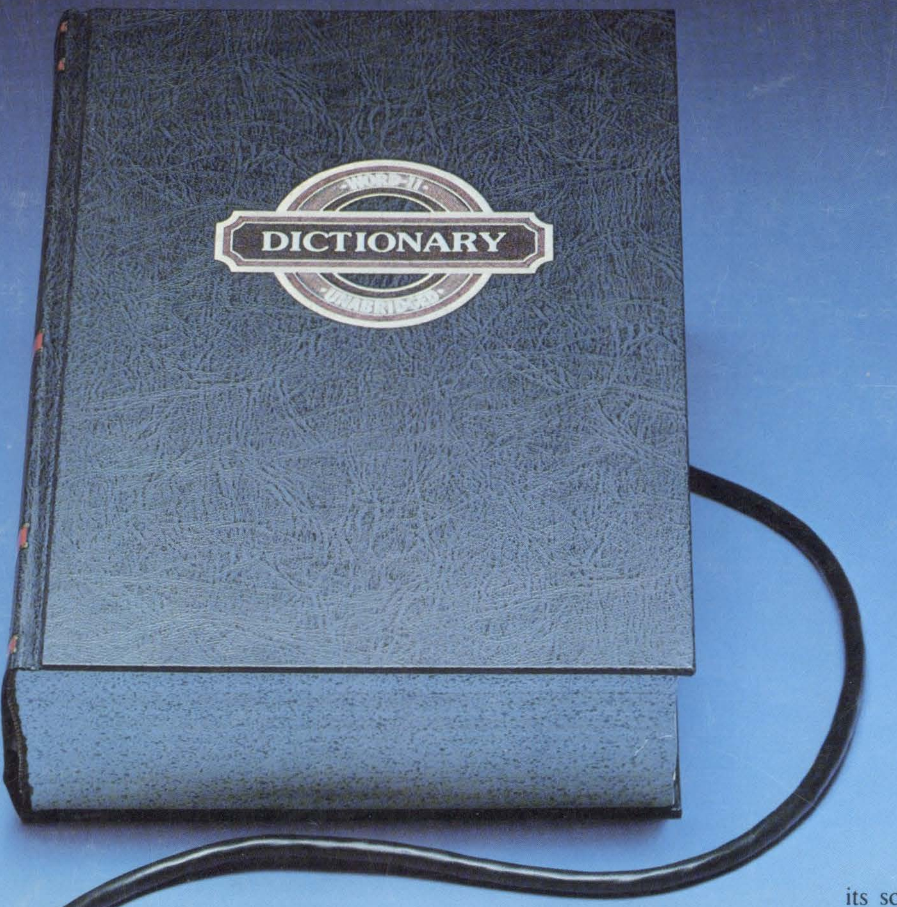
ABLE COMPUTER-UK, 74/76 Northbrook Street,
Newbury, Berkshire, England RG13 1AE.
(0635) 32125. TELEX 848507 HJULPHG.

ABLE COMPUTER-GERMANY, Forsthausstrasse 1,
8013 Haar (Near Munich), West Germany.
089/463080, 463089.

VAX and UNIBUS are trademarks of Digital Equipment Corporation.

CIRCLE 56 ON READER CARD

Responsive Word Processing. Take Our Word For It.



WORD-11.

WORD-11 is a complete word processing system. It's responsive. It's powerful. And it's sharable on up to fifty terminals while running concurrently with data processing.

WORD-11 is talented, too. Designed to work on Digital's family of mini-computers, WORD-11 has all the standard word processing functions. For more sophisticated requirements, WORD-11 provides multiple dictionaries for spelling error detection. Automatic index and table of contents creation. Text search and replace. User defined keys. User-controlled hyphenation. And automatic footnoting.

Included with comprehensive list processing, WORD-11 offers fast sorting. Flexible selection. And extensive math functions.

And WORD-11 has been proving itself for years. You'll find successful installations in small businesses, Fortune 500 companies, in universities and in banks—wherever Digital computers are in place.

Yet despite its sophistication, WORD-11 is easy to learn and easy to use. It's extremely cost effective. And it's available on RSX-11M, RSX-11M-PLUS, and RSTS/E.

See how WORD-11 responds for you. Write for our brochure or call us direct:

- **Corporate Office:** 181 W. Orangethorpe Ave., Suite F,
  Placentia, CA 92670, (714) 993-4160,
  Telex 182-278.
- **New York Office:** 420 Lexington, Suite 647,
  New York, NY 10170, (212) 687-0104.
- **Washington D.C. Office:** 4520 East-West Hwy.,
  Suite 550, Bethesda, MD 20814, (301) 657-4098.

dpd Data Processing Design, Inc.
AUTHORIZED digital COMPUTER DISTRIBUTOR

Overseas Distributors:

Management Information Services PTY. LTD.
Melbourne, Australia

Jenson, LTD.
Bristol, England

Network Computer Services PTY. LTD.
Sydney, Australia

Systime, LTD.
Leeds, England

On-Line Computing PTY. LTD.
Subiaco, W. Australia

CIRCLE 59 ON READER CARD