THE MULTI-TASKER

Volume 15, Number 8 March 1982

The Newsletter of the RSX-11/IAS Special Interest Group

Contributions should be sent to: Editor, The Multi-Tasker, c/o DECUS, One Iron Way, MR2-3/E55, Marlboro, MA 01752

European members should send contributions to: Colin A. Mercer, Tennant Post, High Street, FAREHAM, PO16 7BQ, Hants, England

Members in Australia or New Zealand should send contributions to: Clive Edington, CSIRO, Computing Research 314 Albert St., East Melbourne, VIC 3002, Australia

Letters and articles for publication are requested from members of the SIG. They may include helpful hints, inquiries to other users, reports on SIG business, summaries of SPR's submitted to Digital or other information for the members of RSX-11/IAS SIG.

All contributions should be "camera-ready copy" e.g. sharp black type in a 160x240 mm area (8 1/2" x 11" paper with 1" margins) and should not include xerox copies. If you use RUNOFF to prepare your contribution the following parameters have been found to be satisfactory:

.PAPER SIZE 60,80 .LEFT MARGIN 8 .RIGHT MARGIN 72 .SPACING 1

These parameters assume output on a lineprinter with a pitch of 10 char/inch. Adjust the parameters to maintain the same margins if another pitch is used.

TABLE OF CONTENTS

Copyright ©. 1982, Digital Equipment Corporation
All Rights Reserved

It is assumed that all articles submitted to the editor of this newsletter are with the authors' permission to publish in any DECUS publication. The articles are the responsibility of the authors and, therefore, DECUS, Digital Equipment Corporation, and the editor assume no responsibility or liability for articles or information appearing in the document. The views herein expressed are those of the authors and do not necessarily express the views of DECUS or Digital Equipment Corporation.

READ THIS FIRST

If you have not registered for any the pre-symposium seminars at Atlanta, you must do so now! See the article from the Training Working Group for details.

SPR Questionaires and RSX-11M V3.2 SPR's (December/January issue) are starting to come in to the Multi-Tasker. At Los Angeles, I offered to buy beer at the next two Magic sessions if I got responses from 2500 sites and write a symbolic Fortran debugger if 10,000 sites responded. The offers still stand, but as of this writing, you have 2400 left to go just to get the beer. You can still respond!

Volunteers Wanted

Volunteers are urgently needed in two areas: people to report on the Atlanta symposium and people to report on the process on conversion from IAS to RSX, RT-11 to RSX, RSX-11M to RSX-11M Plus, and most importantly, RSX to VAX/VMS.

I would like to get complete coverage of the Atlanta Symposium, particularly the sessions that never get printed in the proceedings like the Digital product panels, Magic sessions, and others. To accomplish this, I would like to hear from anyone who will volunteer to cover one session and submit an article within one month after the symposium. Please phone me (anytime) at the number below. With many people involved, it should not be a backbreaking task for anyone and everybody will benefit.

Next, one of the biggest problems that ever confronts a site is converting from an old system to a new one. Margaret Knox is spearheading the SIG's effort and is leading by example. She will be submitting a monthly diary on her progress in converting to VAX/VMS, starting with this issue. We would like to find other sites who can do the same. Perfect, typeset articles are not needed. Just keep a informal notebook of your journey and send a copy to the Multi-Tasker once a month. We would also like to here about anyone developing RSX products using VMS and feedback the VAX Application Migration Executive (AME). What works, what doesn't, what is or is not needed. Again, contact me anytime at the number below if you can help.

We are also interested in diary's from people and sites brand new to RSX and the learning process that takes place. Digital is still selling RSX-11M and the people following you will benefit.

Finally, almost all of us will be confronted with RSX-11M V4.0 or RSX-11M Plus V2.0 this year. Anytime you encounter a problem or find a short-cut, drop a note to the Multi-Tasker. With all the time this damn newsletter takes, I don't have time to find bugs in RSX-11M.

Ralph Stamerjohn
Multi-Tasker Editor

Phone: (314) 694-4252 (3-5 pm, CST)

Hamma Runs for DECUS Executive Board

George Hamma, current chairman of the RSX/IAS Special Interest Group, has been nominated for the office of Special User Group Coordinator (SUG) on the DECUS U.S. Chapter Executive Board. The election for this position, and the positions of Program Library Coordinator and Standards Coordinator will be held this spring. All U.S. Chapter Installation members are eligible to vote. Ballots will soon be mailed to all installation members. The following is George's biography, past DECUS positions, and policy statement:

George Hamma received a BS degree in physics from the University of California in 1967. Since that time he has worked with a variety of minicomputer systems and vendors in realtime data-acquisition and control applications. Involvement with Digital Equipment Corporation computers began in 1969 at Lockheed with a PDP-9, and has since included PDP-8, PDP-11, and VAX systems. Since his co-founding of Synergistic Technology in 1978, George has worked with DEC computers as hosts for a variety of realtime systems in his role as Vice President, Systems.

George first became involved with DECUS in 1969 as a user of the Program Library. More active involvement began with attendance at the Fall 1970 Symposium. He has increasingly participated in more than half of the Symposia since then, presenting papers and serving as Chairman of several sessions. As a PDP-11 user, he joined the RSX-11/IAS SIG in 1975, and the SIG Steering Committee a year later. Since that time George's involvement in several SIG's has grown, including participation in Symposia and other activities.

George founded the San Francisco Bay Area RSX-11 Local Users Group (BAYLUG) in 1975 and served as its Coordinator for four years. Since then he has been an effective advocate for the foundation of LUG's and their support through both DECUS and DEC.

As an early participant in the RSX-11/IAS SIG Tape Copy process, George helped to formulate the LUG-supporting policies for access to the SIG Tape Collection. In addition, as Tape Copy Coordinator, he supervised the assembly of several of the RSX-11/IAS SIG Collections. Through membership in the Program Library Committee, George has led the activity to resolve Tape Collection and Library roles and their futures.

George Hamma is currently the RSX-11/IAS SIG Chairman, with responsibility for coordinating the present operations of a very active SIG as well as preparing for future growth of its membership.

George seeks the opportunity to expand DECUS support for LUG's, prepare for expected growth of the SIG's, and support improved visibility for DECUS among the users of DEC computers. Many users are simply not aware of the existence of DECUS or the benefits accompanying participation in DECUS activities, a situation George hopes to help solve as SUG Coordinator.

From Five Years Ago

The March 1977 (Vol.7, No.3) issue of the Multi-Tasker was a short one. Articles included were:

* A summary of some of the most recent RSX submissions to the DECUS library, including a BRO-like task for RSX-11D called SEND.

* A sound-off letter from David Stern concerning the RSX-11M V3.0 update course given by his local DEC office. David felt that rather than charging customers for the update seminar and making the seminar an income-producing occasion, DEC should be willing to make the session a freebie.The committment of one software specialist's time could ease the transition pains for local customers and encourage their support for the new software.

* An announcement of the formation of a new Toronto LUG for IAS users.

* The proposed "Guidelines for User-Submited Action Items" was published. The guidelines detailed the steps necessary for the resolution process and for formation of working groups.

* User input was solicited for the draft of the new bylaws.

Working Group News

Retired Versions Of RSX-11M

Bill Burton
Texas Research Institute of Mental Science
1300 Moursand Avenue
Houston, TX 77030

The file transfer program (FLX) distributed with RSX-11M versions 3.0 and 3.1 only handles 6-character filenames. This causes some problems when using FLX for backup and recent DECUS Symposium tapes. The limitation can be corrected. DOS format magtapes carry the full 9-character filename and 3-character extension in the tape headers. However, it was only until RSX-11M V3.2 that FLX would use the full fields. The patches below will give previous versions of FLX the ability to do 9-character filename matches and directory listings.

Patch Procedure:

1. Create a working copy of [1,20]FLX.OLB from your RSX-11M V3.0 or V3.1 distribution kit.

2. Use the librarian (LBR) to extract the following modules:

   LIPRC2
   MTFLCK
   SETODT - version 3.1 only
   SETOUT - version 3.0 only

3. Enter the appropriate patch files below and assembly them into three corresponding .POB files.

4. Use the object file patcher (PAT) to insert the patches into the matching .OBJ files:

   ```
   PAT xxxxx.OBJ;2=xxxxx.OBJ;1/CS:yyyyyy,xxxxx.POB/CS:zzzzzz
   ```

Modules	Version 3.0	Version 3.1
LIPRC2.OBJ	106254	070075
LIBRC2.POB	005134	004664
MTFLCK.OBJ	040477	040477
MTFLCK.POB	011701	011701
SETODT.OBJ	------	013244
SETODT.POB	------	010360
SETOUT.OBJ	013117	------
SETOUT.POB	010233	------

5. Replace the patched object modules in the working copy of FLX.OLB.

6. Rebuild FLX.TSK from the working copy of FLX.OLB and the appropriate command file. Note, on version 3.0, I had to edit FLXBLD.CMD to increase the partition size from 8.0 KW to 8.5 KW (change the line PAR=GEN:0:40000 to PAR=GEN:0:40600).

[1,20]	FLXBLD.CMD	-for unmapped systems
[1,24]	FLXBLD.CMD	-for mapped systems

7. The new version of FLX can be tested by the following commands. The old version gave a "FILE ALREADY EXISTS ERROR" error message, the new one will not.

   ```
   >RUN FLX
   FLX>MT0:/ZE
   FLX>MT0:/DO=ABCDEF.DAT/RS
   FLX>MT0:/DO=ABCDEFGHI.DAT/RS
   ```

I have tested the patches on our system (version 3.0) using a TM-11 look-alike magtape drive. I do not know whether these patches will affect the handling of

cassettes (CT:) or DECtapes (DT:). Feedback to me, as chairman of the Retired Versions of RSX-11M Working Group, would be appreciated. Either write to the address above or phone (713) 797-1976, extension 501.

Patch Files for FLX, RSX-11M V3.0

```
        .TITLE  LIPRC2
        .IDENT  /M02A/
        .PSECT  INSTR
.=.+1476
        .WORD   11.

        .END

        .TITLE  MTFLCK
        .IDENT  /V01A/
        .PSECT
        .GLOBL  N.FNAM
.=.+54
        BR      114$
        NOP
        NOP
62$:
.=.+26
110$:
.=.+4
114$:   CMP     2(R2),N.FNAM+2(R5)      ;Check filename 4-6
        BNE     110$                    ; If NE - not same
        CMP     24(R2),N.FNAM+4(R5)     ;Check filename 7-9
        BR      62$                     ; If EQ - go finish test

        .END


        .TITLE  SETOUT
        .IDENT  /M02A/
        .PSECT
        .GLOBL  O.UNTP
.=.+312
312$:
.=.+70
        BCS     662$                    ;Check for more
404$:
.=.+40
444$:                                   ;Finish name check
.=.+216
662$:   CMP     O.UNTP,#12              ;Complete filename testing
        BNE     404$                    ; If NE - not magtape (MT or MM)
        CMP     16(R0),16(R2)           ;Check last three characters
        BEQ     444$                    ; If EQ - found file
        JMP     312$                    ;

        .END
```

Patch Files for FLX, RSX-11M V3.1

```
        .TITLE  LIPRC2
        .IDENT  /M02A/
        .PSECT  INSTR
.=.+1406
        .WORD   11.

        .END

        .TITLE  MTFLCK
        .IDENT  /V01A/
        .PSECT
        .GLOBL  N.FNAM
.=.+54
        BR      114$
        NOP
        NOP
62$:
.=.+26
110$:
.=.+4
114$:   CMP     2(R2),N.FNAM+2(R5)      ;Check filename 4-6
        BNE     110$                    ; If NE - not same
        CMP     24(R2),N.FNAM+4(R5)     ;Check filename 7-9
        BR      62$                     ; If EQ - go finish test

        .END

        .TITLE  SETODT
        .IDENT  /M02A/
        .PSECT
        .GLOBL  O.UNTP

.=.+312
312$:
.=.+70
        BCS     662$                    ;Check for more
404$:
.=.+40
444$:                                   ;Finish name check
.=.+216
662$:   CMP     O.UNTP,#12              ;Complete filename testing
        BNE     404$                    ; If NE - not magtape (MT or MM)
        CMP     16(R0),16(R2)           ;Check last three characters
        BEQ     444$                    ; If EQ - found file
        JMP     312$                    ;

        .END
```

Training/Pre-Symposium Seminars

Tom Viana
Naval Underwater Systems Center
Code 3511, Building 1117-1
Newport, RI 02840

The RSX-11/IAS SIG has scheduled three pre-symposium seminars for the day before the Spring DECUS Symposium in Atlanta (May 9, 1982). Interested SIG members will find these seminars informative, and because they are scheduled with the symposium, convientent and cost effective.

RSX-11M Systems Programming for Applications Programmers
RSX-11M I/O Drivers: A Software Engineering Approach
Care and Feeding of the Files-11 Disk Structure

The SIG chooses the subjects and instructors with care. The first two seminars have been held previously to full classes and offer well-developed formats and complete handouts. The Files-11 seminar is being offered for the first time in response to the tremendous interest to previous symposium sessions on the subject. Full descriptions of each seminar are below:

TIME IS SHORT

The cutoff date for registering for a seminar is April 9th. There is a good chance this newsletter will not reach you until after then. The DECUS office may cancel a seminar on April 9th if not enough people have registered. However, they will still accept registrations, provided the class is not cancelled or already full. You must act immediately. If you would like to find out the status of a class, call Mary Burke at (617) 467-4166.

RSX-11M Systems Programming for Applications Programmers

This seminar presents an internal view of RSX-11M from an applications point of view. It should give Applications Programmers a better feel for the internal logic of RSX-11M and place them in a better position to use executive/file system features for their particular application. During the seminar, several examples will be studied to provide an indication of ways RSX-11M has been used or modified to meet the needs of an installation. A knowledge of Macro-11 would help the attendee, but is not essential. The following topics will be covered:

- o Executive concepts and control flow
- o Pool usage and abuse
- o Common scheduling/swapping problems
- o User written privileged tasks
- o Files-11 ACP QIO functions
- o Executive changes in RSX-11M V4.0
- o Examples and case studies

Instructor: Robert Bismuth, Redkite Software, St. Thomas, Swansea, Wales

Registration Fee: $150.00

RSX-11M I/O Drivers: A Software Engineering Approach

You have an I/O driver to write for RSX-11M. You have looked over the "Guide to Writing and I/O Driver" manual and have become totally bewildered by UCBs, SCBs, dispatch tables, function bits, FORKs, INTSVs, and interrupt vectors. But you have to write the driver by Friday, and you STILL do not have the foggiest idea where to start. This seminar is aimed at people described above and those who will find themselves in that situation.

The seminar will provide a roadmap to writing an I/O driver for RSX-11M. We won't be doing a bit-by-bit tour through the manual, but rather we will be highlighting and correlating what at first glance seems to be an unrelated mess of detail.

We will show a step-by-step development of an I/O driver, starting with a hardware specifiation and a skeleton driver. As we cover different aspects of driver development, we will add code to the skeleton. The entire driver development process will be presented in this step-by-step fashion, so that attendees leave the seminar having seen an actual driver developed from start to finish.

Several programming and debugging aids will be distributed as handouts. Case studies and examples of non-standard uses of I/O drivers will be discussed.

Instructor: James A. McGlinchey, Consultant, Horsham, PA.

Registration Fee: $150.00

Care and Feeding of the Files-11 Disk Structure

Have you ever had a "disk disaster?" Do you understand the Files-11 on-disk structure used by RSX? The Digital documentation regarding the organization of Files-11 disks is rather sketchy and is particularly confusing to newcomers to RSX. It is important that people responsible for RSX systems know how files are structured on their disks, what preventive maintenance may be done, and how to repair the damage done in a disk disaster.

This seminar is directed to those people at all levels of RSX system experience who want to learn about the Files-11 on-disk structure and how to go about repairing corrupted disks. It will cover the following topics:

- o RSX-11M/M-Plus file I/O mechanism and flow
- o Details of the Files-11 on-disk structure
- o Dynamics of disk space and file management
- o Preventive maintenance of Files-11 volumes
- o Techniques for recovery from disk disasters

Instructors: Robert B. Denny, Creative System Design, Pasadena, CA
Dr. L. Micheal Fraser, AFRRI/Computer Science Dept., Bethesda, MD

Registration Fee: $150.00

Help Yourself

"Help Yourself" is a place for you to get your tough questions answered. Each month, questions from readers will be published. If you have a question, send a letter to the Multi-Tasker at one of the addresses listed on the cover.

We would also like to publish the answers to questions. If you can help someone, send a letter to the Multi-Tasker or call Ralph Stamerjohn (see first page). Your answer will be sent directly to the person in need and published in the next edition of the Multi-Tasker.

This Month's Questions

4010 Graphics Package

We are looking for a RSX-11M V3.2, Fortran-callable graphics package for the RETROGRAPHICS installed in an ADM-3A terminal. This is the equivalent to the Tektronix 4010 graphics terminal.

P.A. Johnson, Raman Research Institute, Bangalore 560 006, India. Phone 30122 to 30129.

Hints And Things

"Hints and Things" is a monthly potpouri of helpful tidbits and rumors. Readers are encouraged to submit items to this column. Any input about any way to make life easier on RSX/IAS is needed. Please beware that items in this column have not been checked for accuracy.

Interfacing a Technicon Autoanalyzer

Raymond Willis
U.S. Forest Service
N.E. Forest Experiment Station, Rt. 2
Berea, KY 40403

At the N.E. Forest Experiment Station in Berea Kentucky, we have a set of programs that interface a Technicon Autoanalyzer to a RSX-11M system.

Using these programs, the recorder tracings that were orginally displayed on a strip chart recorder are now displayed on a graphics CRT. Everything that was done by the Technicon microprocessor is now done by the PDP-11.

One program continuously stores the latest analog value for each colorimeter in a static common region. Simultaneously, a program for each of the autoanlyzer channels is in operation and they take the analog value from the common region, calculates the concentration of each sample, and prints the value beside the corresponding peak displayed on the graphics CRT - plus storing the value in a disk file.

Another program allows the operator to assign sample numbers to each cup, obtain results for completed samples, and get a list of samples that need redoing due to some error condition.

The programs were written assuming one to three channels on the autoanalyzer are in operatuon simultaneously and they have been tested with either one or two channels in use. If more than three channels were being used, the programs would need modification. The CRT we use is a HP 2647, so the subroutines that plot points, clear the screen, and perform other graphic functions would have to be modified for other graphic CRT's.

We will distribute the programs either on RK05 disk packs or by paper listing. If you send an RK05 (our only exchange media), please include a pre-paided return mailer. Send all requests to the address above.

DDT, FPE Problems

Glenn Everhart
RCA GSD Engineering
Route 38
Cherry Hill, NJ 08358

I have found a bug in DDTKNL which is present in all version of DDT distributed on the RSX SIG tapes. The problem causes incorrect PSWs to be sent to tasks. The correction is to change one line in DDTKNL.MAC. The old line is:

```
        MOV     (R0)+,SNDDAT+14         ; BOTH PLACES
```

Change this to:

```
        MOV     (R0)+,SNDDAT+10         ; BOTH PLACES
```

This will correct the problem. I will be including code in the next release to handle I and D spaces for RSX-11M Plus V2.0, but the preliminary code has several restrictions and will be conditionalized. This one fix will however allow correct operation of dual task DDT in all other cases.

The Floating Point Emultator (FPEM) for RSX-11M which I put onto the SIG tapes has caused some concern. The program can crash your system if you allow it to checkpoint OR SHUFFLE! Just running it, even though fixed, in the middle of GEN can allow it to shuffle. The system will not long survice this. The solution is to place FPEM in its own partition or at the bottom of GEN. It is then reliable, though slow.

From The Wizards Book Of Magic

The Magic sessions at the symposium have become one of the most popular features of the RSX/IAS SIG. This column has the same purpose: to exchange and discuss ideas on non-standard RSX and IAS programming. Readers are encouraged to submit items to this column and are also warned that the material here have not been checked for accuracy. Also, implementation of any items from this column will be completely unsupported. The material here is potentially dangerous: incorrect usage could result in system crashes and other incorrect system operations.

Tricking DSC

Robert Bismuth
Redkite Software Limited
Swansea, Wales, U.K.

It is well-known that DSC does not allow RSX users to extract individual files or directories from the tapes which is produces. A problem occurs when a site receives a DSC tape image of a disk type that it does not have in its configuration. This articles gives a method of "tricking" DSC, without the use of external software, into recreating the disc volume on a larger or smaller disc.

If the site has a disc of larger capacity than the disc from which the tape was produced, there is no real problem: DSC allows the user to recreate the image of a smaller disc on a larger one. Note that the larger disc then has the same "virtual" size as the original smaller disc because of the storage map recreated by DSC. The user should then use BRU, FLX or RMSBCK to backup the volume, reinitialize (using INI, NOT BRU) the disc volume and restore.

Should the site not possess a disc of larger capacity than the original disk backed up by DSC, the solution is somewhat more complex. The site should first determine how many blocks were in use on the input disc when the DSC tape image was made. Should this be less than the total number of blocks available on some drive in the target configuration, then it is possible to "trick" DSC into restoring that tape onto that drive. The procedure is as follows:

1. Convert the total block size of the original input disc (not just blocks backed up) into a 32-bit (double word) integer.

2. Determine the executive address of the target disc's UCB from the task builder map of the executive (usually found as LB:[1,34]RSX11M.MAP - should it not exist, enough of it may be recreated by the taskbuilder command:

   ```
   TKB ,LB:[1,34]RSX11M/-SP=LB:[1,54]RSX11M.STB
   ```

   given from a privileged terminal).

3. Using the MCR OPEn command, open the UCB and skip down to the offset U.CW2 (reference: Guide to Writing an I/O Driver, ie. The Hitch-Hiker's Guide to RSX11M), note the contents and replace with the high 16 bits of the 32 bit integer block size.

4. Note the contents of the next word (U.CW3) and replace with the lower order 16 bits of the 32 bit integer block size.

5. Place a scratch pack in the drive and spin up. Load the DSC tape.

6. Install and invoke online DSC: copy the tape to the disc.

7. When finished, use the OPEn command to restore the two UCB words(U.CW2 and U.CW3) to their former values.

8. Load a scratch tape on the tape drive.

9. BRU the disc onto the tape.

10. When finished, allocate the disc drive and initialize using the MCR INI command.

11. MOUnt the disc and create the desired UFD's.

12. BRU the required files from tape to disc by directory/filespec.

The procedure works because U.CW2 and U.CW3 contain the size of a disk. You can safely make a disk larger than it really is, provided you do not try to write to the mythical blocks. Because DSC compress all files to the start of the disk, as long as the allocated size is less than the actual size of your disk, DSC will not write to blocks that do not exists.

Note, that this procedure will NOT restore a bootable system image as the target disc may require a different BOOT/SAVE driver, and File I.D.s will have been changed.

Should the site not possess a drive of larger capacity than the total size backed up on tape, the only solution is to obtain or write a utility to extract individual files from a DSC tape (Richard Kirkman has submitted such a tool to the European Sig Tapes).

This article is partly written as an "ad" for the upcoming Atlanta Symposium: it forms a small part of a session called "Weird Things to do with RSX11M".

Los Angeles Symposium Wrapup

Ralph Stamerjohn
Multi-Task Editor

The Los Angeles Symposium had everything: over 500 sessions, an actual night out on the farm, and plenty of exercise walking between the two hotels. Joe Sventek set the record by giving one session in the Bonaventure that ended at two and starting another in the Biltmore two minutes later.

At the closing session, I asked attendees to copy their trip reports to the Multi-Tasker. Eight sites responded and their sessions notes are summarized below. Thanks should go to Harrision Banks, W.R. Ledbetter, C.W. Holeman, David Eisenhauer, Don Hallberg, Susan Dakuzaku, Joelle Claudon Linda Slawson, Jon Estep, William Burton, Philip Cannon and Ken Johnson. Their comments are summarized below.

The RSX/IAS SIG had an extensive handout at the symposium and I will not repeat material published in it. Unfortunately, all copies were distributed at the symposium, so you cannot get a copy from DECUS. However, you should be able to find someone in your Local User Group with a copy they will loan you. Or if enough people show interest in a particular paper, I will publish it in a future issue. Included in the 167 page book are the following session notes:

	Introducting FORTRAN-77	Bob Abramson
	FORTRAN-77 Internals	Bob Abramson
*	A High Performance, Multikey ISAM	Bob Denny
	RUNOFF Tutorial	Bob Denny
	DECUS C Language System	Bob Denny
	CCL - Console Command Language	Jim Downward
	SYSLOG - Accounting and Performance	Jim Downward
	Accounting Enhancements to IAS/PDS	Richard Evans
*	Theory of Debuggers on PDP-11's	Glenn Everhart
*	Introduction to DDT	Glenn Everhart
	Introduction to DCL	Mike Fox
	Indirect as a Programming Language	Mike Fraser
*	Recovering from Disk Disasters	Mike Higgins
*	Fast F4P I/O	Margaret Knox
*	What is an ACP?	Jim McGlinchey
	Writing a Despooler for I/O	Jim Salman
*	Realtime Interrupt handling User F4P	David Schultz
*	What is a Virtual Disk?	Ralph Stamerjohn
*	A Walk Through RSX-11M	Ralph Stamerjohn

Better Computing Thru FCS Resident Libraries	Joe Sventek
What Resident Libraries Can Do	Joe Sventek

* Paper also published in Symposium Proceedings which can be ordered from DECUS (Volume 8, Number 2). The Proceedings have many other papers interesting to RSX/IAS sites.

Other sessions have already been published in the last two issues of the Multi-Tasker. These include the following:

Reducing the Size of a Fortran Program	Larry Baker
Multiple Writers to FCS Files	Ken Johnson
IAS Question and Answer Session Transcript	T. Bossert, P. Clayton
RSX-11M Question and Answer Session Transcript	B. Denny and others

Larry Baker (Reducing the Size of a Fortran Program), Bob Denny (RUNOFF Tutorial), and Joe Sventek (Resident Library papers) went in January to Digital's studio in Maynard and video taped their presentations. These will be made available to LUG's and through the DECUS library. More details will be published in the Multi-Tasker when available.

The following are rough notes taken from the reports sent to me and my own notes. There is no particular order or sense to them:

o The display hall, with its many exhibits and book store was very valuable. People got a chance to see and touch all of Digital's current products and discovered many things they were unaware existed:

  * "We discovered a self-help tutorial package. It is designed to train a person in the basics of Fortran, assembly language, and the operating system."

  * "I found information in the exhibit hall about DIGITAL products that our salesman has not yet mentioned, specifically small stand-alone or distributed process control systems."

o Various tricks were learned at sessions or just talking to people.

  * "A very simple technique for catching undeclared variables in a FORTRAN program is to do an IMPLICIT COMPLEX (A-Z). Then all undeclared variables will have a complex data type in the FORTRAN map. This is the easiest way I know of to locate undeclared variables in FOR or F4P/F77."

  * "If you use the shuffler, keep it in its own partition. It is only 485 words long."

  * Some simple techniques were noted for speeding up TKB. Eliminating the cross reference improves performance by 20%. Eliminating the map saves another 20%. If you have a big taskbuild and specify many modules by letting TKB search libraries (library/LB), you can save up to 30% by naming modules explictly (library/LB:modl:mod2...modn).

o There were field-test reports on RSX-11M V4.0 and Fortran-77. Most of the information for for RSX-11M V4.0 has already been published in the Multi-Tasker. Fortran-77 was tested for four months at 7 internal Digital sites and 20 customer sites (5 IAS, 4 RSTS/E, 6 RSX-11M, 5 RSX-11M Plus). The Fortran-77 field test comments were:

  * "Overall at one site, 2500 compiles, 2200 clean. Many users liked new features."

  * "Few bugs found, none by normal users."

  * "Found few problems."

  * "Prefered by students over WATFOR except for hassles in learning TKB for beginning programmers."

  * "Some programs ran faster."

  * "Biggest problem is full-set Fortran-77 features not implemented... Should have gone to FULL Fortran-77 to enable users to do away with pre-compilers."

  * "Not as full an implementation as is available for VAX. Possible problem for downward compatibility."

  * "Compiler is larger."

  * "RMS support does not allow character strings as key variable."

  * "Virtual array has some problems known for IAS."

  * "Partially implemented features are unpleasant. Constant expressions in PARAMETER statements is not allowed. Lack of string concatentation options."

o The Best of the SIG Tape session had people briefly name programs they had used and found useful. Among the programs mentioned were:

  * Index to SIG tapes (Fall 1977 through Fall 1980). Spring 1981 [346,100].

  * INDEX, a cross reference utility for FOR or F4P. Spring 1981 [302,304].

  * DISOBJ, a object module disassembler. Spring 1981 [373,23].

  * FDT, debugger package for F4P. Fall 1981 [302,212].

* REI, a file undeleter. Fall 1981 [307,22].
* CPA, crash pool analyzer. Fall 1980 [331,30].
* GREP, string pattern searcher. Spring 1981 [312,315].
* TYPE, file lister with many features. Spring 1981 [300,107].
* Loadable XDT. Spring 1981 [346,100] is best version. Previous had bugs.
* DDT, symbolic debugger. Spring 1981 [312,315].
* Software tools, 93 total utilties. Spring 1981 [307,30] to [307,34].
* FFL, fast FLX to disk. Spring 1981 [312,315].
* CVL, change volume label. Spring 1981 [344,43].
* XMITR, remote file transmission. Spring 1981, Fall 1981 [312,315] (both needed).
* SUPERMAC, structured Macro. Fall 1978, [312,1].

One common thread through all reports was how valuable the symposium was in dollars. Everyone reported information specific to their site that will save money with current application development, future capital expansions, problems fixed, or free software discovered.

Spring 1981 DECUS Symposium

Jim McGlinchey
RSX/IAS SIG Symposium Coordinator

The Spring 1982 DECUS Symposium will be held at the Atlanta Hilton and Towers from May 10-14, 1982. The RSX/IAS Special Interest Group has scheduled over 50 sessions. And remember, our sessions are just one set of the 15 parallel sessions going on at any one point in time. If you measure the success of a symposium by never getting bored, come to Atlanta.

The RSX/IAS symposium sessions will concentrate on two important areas, detailed information on the release of RSX-11M V4.0 and several sessions on migration, especially RSX to VAX and RSX-11M to RSX-11M Plus. There are also sessions directly targetted to new users. There will be a question and answer session for new user only and a series of tutorials on system generation.

The following is a synposis of the currently planned sessions:

o RSX/IAS BUSINESS SESSIONS

* RSX-11/IAS SIG ROADMAP SESSION

  The Roadmap session describes the week's activities and provides tips on how to survive the week. Sessions sponsored by the SIG will be highlighted, as well as other sessions that may be of interest to members. The session will be oriented for first time attendees, however, it is a good idea for all members to attend, since we will point out any last minute changes. If time permits, attendees will be asked to volunteer comments on their expectations, so we can use them for feedback to compare against the SIG's current goals (1 hour, Orientation: General)

* RSX-11/IAS SIG OPENING SESSION

  This session will introduce the SIG to new members and summarize various SIG activities over the last six months. All RSX-11/IAS SIG members are urged to attend, especially new members or first time attendees. Agenda items include: introduction of the new SIG Steering Committee, introduction of the Digital representatives, reports on current SIG activities, and a policy question and answer session. (1 hour, Orientation: General)

* RSX-11/IAS SIG CLOSING SESSION

  The purpose of this session will be to report on, discuss, and evaluate the past week's symposium. All SIG members are urged to attend as this is the time to bring up suggestions and comments. Your inputs are important as they will be used for both future Symposia and SIG planning. Also, at the end, a final question and answer session will he held with both SIG Steering Committee members and Digital representatives available. (1 hour, Orientation: General)

* RSX-11/IAS SIG WORKING GROUPS SESSION

  This will be an informal session during which all of the existing working groups sponsored by the RSX/IAS SIG will meet in parallel. The meeting will start with brief reports from the working groups. Then each group will move to a part of the room and conduct their own meeting. Anyone is welcome to join a working group or form a new one. (1 hour, Oriention: General/Technical)

o QUESTION AND ANSWER SESSIONS

* RSX-11M AND RSX-11M-PLUS NEW USERS QUESTION AND ANSWER SESSION

  At this session, a panel of experienced RSX-11M and RSX-11M-Plus users will answer questions typical to new sites and new users. Most attendees at the Symposium fall into this class. This session is for you. Questions too technical or which fall outside the expertise of the panel will be directed to the main RSX-11M question and answer session. (2 hours, Orientation: Technical)

* RSX-11M AND RSX-11M-PLUS QUESTION AND ANSWER SESSION

  During this session, a panel of Digital developers will attempt to answer all technical questions related to RSX-11M, RSX-11M-PLUS, and layered products. This session provids a public forum for technical and policy interchange with Digital personnel. (3+ hours, Orientation: Technical)

* IAS QUESTION AND ANSWER SESSION

  This session is the high point of the symposium for the serious IAS user. Several IAS experts and a room full of experienced users respond to questions from the floor. Bring your IAS problems, questions, and experiences to this valuable session. (3+ hours, Orientation: Technical)

* RSX-11M MAGIC

  This session is a free-wheeling discussion on the use of RSX-11M by Wizards and Apprentices alike. The RSX-11M Magic Session brings out the true personality of the RSX user: a person who oftentimes bends the operating system to his or her will, sometimes in a subtle fashion, sometimes using a bludgeon. Many traditions are associated with this session and all will be properly observed. (3+ hours, Orientation: Technical/Intermediate-Advanced)

* IAS MAGIC

  With the annoucement of IAS system maturity, IAS users may rely on an unchanging base for development and application of their magical ideas. This session is a forum for ALL IAS users to learn of programs, both new and old, and to compare experiences and ideas with many people. (3+ hours, Orientation: Technical/Intermediate-Advanced)

o DIGITAL SPONSORED SESSIONS

* RSX-11 PRODUCT PANEL

  An overview of the new releases of the RSX-11 family will be presented. In addition, an update of the SIG MENU responses will be provided. This will a relatively non-techical presentation, with more technical details available at other sessions. (1 hour, Orientation: General)

* IAS PRODUCT PANEL

  Digital Product Management will conduct a panel on IAS policy. This will be followed by a question and answer session for questions related to policy issues and other areas of general interest. (1 hour, Orientation: General)

* FUNCTIONAL DETAILS OF RSX-11M/S V4.0

  This session will be a more detailed presentation of the functionality of RSX-11M/S V4.0 than was presented at the RSX-11 Product Panel. (1 hour, Orientation: Technical)

* FUNCTIONAL DETAILS OF RSX-11M-PLUS V.20

  This session will be a more detailed presentation of the functionality of RSX-11M-PLUS V2.0 than was presented at the RSX-11 Product Panel. (1 hour, Orientation: Technical)

* PDP-11 FORTRAN-77 WORKSHOP

  PDP-11 FORTRAN-77 Version 4 is an upward compatible version of PDP-11 FORTRAN IV-PLUS. FORTRAN-77 runs on RSX-11M, RSX-11M Plus, IAS, and RSTS/E operating systems. The implementation contains all the features specified for the subset level FORTRAN as defined in the latest ANSI FORTRAN standard (X3.9-1978). An overview of the major features of this product will be given and time allotted for questions from present and prospective users. (1 hour, Orientation: General)

* INTRODUCTION TO DCL ON RSX SYSTEMS

  This session is a brief description of the DCL command language to be provided with RSX-11M V4.0 and RSX-11M-PLUS V2.0. This language is easy to use and highly compatible with the DCL implementations on other Digital operating systems. (30 minutes, Orientation: General)

* HOW TO BUILD APPLICATIONS WITH CLUSTER LIBRARIES

  The purpose of this session is to provide an overview of the concept of "clustered" memory resident libraries, as well as a discussion of the enhancement of virtual address space that clustering provides for most high-level language applications. The "mechanics" of how to build an application task that uses clustered libraries will be presented, including examples using RMS-11, FMS, and language OTS systems. (30 minutes, Orientation: Technical)

* RSX-11 SCHEDULING, SWAPPING, AND SHUFFLING

  This will be a tutorial on some of the more important parts of the system affecting responsiveness. Presented will be the major algorithms of the scheduler, swapper, and shuffler, and when that are needed and called. (30 minutes, Orientation: General)

* EXTERNAL HEADERS IN RSX-11

  RSX-11M-PLUS V2.0 now has the capability to never user the task address space reserved for the header after the task has started. RSX-11M V4.0 also has this feature, but hidden and unsupported. This session will cover how to enable and use this sometimes valuable feature (30 minutes, Orientation: Technical)

* IAS NODE POOL

  This session will explain one method of modifying the IAS operating system to provide a substantial increase in the number of useable nodes. Users that are experiencing IAS node pool limitations will find this session very interesting. (1.0 hour, Orientation: Technical)

* WALKTHROUGH THE IAS EXECUTIVE

  This session will present the inner workings of the IAS executive. It will include explanations of what the major modules are, what they do, and how they relate to each other. (1.0 hour, Orientation: Technical)

* THE SPM-11 PERFORMANCE MONITORS

  SPM-11M and SPM-11M-PLUS are software performance monitors for RSX-11M and RSX-11M-PLUS operating systems. The discussion will illustrate the goals, implementation concepts, and sample outputs of the SPM-11 software. (30 minutes, Orientation: Technical)

* UNIFIED PLANT MANAGEMENT: THE FACTORY OF THE FUTURE

  Throughout the world, manufacturers are automating their processes at an ever-increasing rate. This automation effort includes intelligent machines, robots, and powerful computers and terminals on the factory floor. The results of the automation effort are enabling companies to cope, and remain profitable, in the face of high interest rates, increasing government regulation, spiralling costs, and multi-national competition. This paper surveys the state-of-the-art in applying computer power to all levels of business, particularly the lowest level: the factory floor. (1 hour, Orientation: General)

o PANELS AND WORKSHOPS

* RSX-11M SYSTEM GENERATION PROCEDURES

  The intricacies of doing a SYSGEN are well known...or are they? This series of half-hour tutorials will cover the System Generation process from start to finish:

  o What to do with your distribution kit
  o Planning a system generation
  o Autopatching
  o Doing the system generation

* BEST OF THE SIG TAPES

  Starting with the Fall 1977 DECUS Symposium, the RSX/IAS SIG have collected and distributed a collection of software called the SIG tapes. To date, the collection contains over 250,000 blocks of

software. The purpose of this session is for you to call attention to programs on the past tapes that you have found useful. (1.5 hours, Orientation: General)

o MIGRATION SESSIONS

* RSX TO VMS MIGRATION, PART I

  This session will present the problems encountered in migrating high performance graphics applications at the University of Texas from a dedicated RSX-11M system to a VAX/VMS system. This work is in progress and will focus on the University's Vector General 3405 graphics display (3-d stroker), Grinnell 270 image processing system, and networking to a Cyber using a KMC/DMS interface (2 hours, Orientation: System Management)

* RSX TO VMS MIGRATION, PART II

  KMS FUSION has been very active in providing accounting, performance, and human interface tools for RSX-11M. We recently acquired a VAX/VMS system and this session will report on the migration effort to the VAX as well as development of RSX products using VMS. (30 minutes, Orientation: Technical)

* LSI-11 DEVELOPMENT USING VMS

  A LSI-11 development system, orginally located on an RSX-11M system, was migrated to a VAX/VMS system. This session will discuss the problems, successes, and benefits of the migration. (30 minutes, Orientation: Technical)

* AME: RSX DEVELOPMENT WITH VMS

  The RSX development team is now responsible for AME, the Applications Migration Executive portion of VMS. This session will compare the AME with RSX-11M version 4.0 capabilities and the development efforts underway to improve the compatibility. (30 minutes, Orientation: Technical)

* IAS TO RSX-11M PLUS CONVERSION: ISSUES AND EXPERIENCES

  This talk will deal with a few of the issues involved in converting applications from IAS V3.0 to the RSX-11M Plus operating system. Issues covered will include differences in task layout, size, programming, command language, device drivers, privilege tasks. We will also talk about some fancy tricks available under RSX-11M Plus to overcome some of the problems in the conversion. (30 minutes, Orientation: Technical)

o USER TUTORIALS AND PAPERS

* A DISTRIBUTED PROCESSING APPROACH TO ADVANCE FLEXIBLE MANUFACTURING SYSTEMS

  An Advanced Flexible Manufacturing System (AFMS) will be described. The software is modular with global functions performed through the host computer and process control functions through local satellite computers. The communications between machines is through the DECdataway, producing a fully integrated distributed processing system. (30 minutes, Orientation: General)

* DEVELOPMENT OF AN RSX-11M APPLICATION: A HISTORICAL PERSPECTIVE

  TANO Corporation has developed a product, TMS-4, using RSX-11M V3.2 and the PDP-11/44 to perform the functions of an industrial Supervisory Control and Data Acquisition (SCADA) system. TMS-4 represents 4 major product interations, over 10 years of calendar time, over 35 man-years of basic development effort, and over 200 man-years of SCADA applications development.

  Development of TMS-4 started with a PDP-11/05 and a home-grown executive called REX. Later iterations have included RSX-11M V2.0 on the PDP-11/35 and RSX-11M V3.0 on the PDP-11/34. The presentation will cover how new features of RSX-11M have been incorporated into TMS-4 and techniques tried and avoided. (1 hour, Orientation: General)

* ADAPTATION OF AN MCR-BASED IAS SYSTEM TO SERVE VERY UNSOPHISTICATED USERS

  IAS operating under MCR mode offers many pitfalls to an inexperienced user. Over the years, we have seen our people hang themselves and the system in many weird and wonderful ways. Some problems are solved by PDS, but at a prohibitively high overhead for a PDP-11/45 based system such as ours. To solve these problems, we haved modified HELLO, BYE, and MCR so that unsophisiticated user only need to remember how to log on to the system. These users never see - and cannot invoke - MCR, but immediately enter a "menu" mode which prompts them in English for their wants. The total package provides more security for us, and simple operation for the user community. (30 minutes, Orientation: Technical)

* THE MATHEMATICS OF RSX-11M

  With a little imagination, RSX-11M can be equated to a finite-state system governed by a set of basic axioms. Once the fundamental properties of RSX are understood, it becomes easier to solve your application. This session will walk through the basic axioms and attempt to build a better understanding of why RSX-11M works the way it does. (1 hour, Orientation: Technical)

* WIERD THINGS TO DO WITH RSX-11M

  The resourceful RSX system programmer often has to resort to the user of strange and wondrous techniques in order to coerce the system to behave according to plan. This presentation will contain a collection of practical wizardry that can be a valuable addition to any RSX system programmer's bag of tricks. (1 hour, Orientation: Technical)

* RSX COMMAND LINE PROCESSING: CSI AND TPARS

  RSX-11M contains a set of powerful library routines which can be used to handle any arbitrary command line. The basic facilities will be discussed with emphasis on fundamentals (30 minutes, Orientation: Technical)

* USER REPORT ON RSX-11M BRU

  This session will present user experience with BRU. The goal will be to state existing limitations and how to avoid them. Part of the session will cover all known problems and test cases when possible. Hints and tricks involving BRU will also be presented. Note, this session is not a beginner's tutorial on how to use BRU. (30 minutes, Orientation: Technical)

* UNIBUS MAPPING IN RSX-11M

  The first part of this session will be a tutorial on the hardware function of Unibus Mapping Registers (UMRs) on the PDP-11/34, 11/44, and 11/70. Then the allocation, user and deallocation of UMRs by DMA device drivers will be presented. This session will be most helpful to those who already have a general familiarity with PDP-11 memory management. (30 minutes, Orientation: Technical)

* WORD PROCESSING ON MINICOMPUTERS

  This is a presentation on a word processor written in BASIC-PLUS-2 running under RSX-11M. We are currently using it in our hospital to speed paperwork. It has been very well received. Among the many features are canned file retrival, memo form fill-in, line-mode entry, screen-oriented editing, and help modes. (30 minutes, Orientation: General)

* USING RUNOFF TO AUTOMATICALLY GENERATE REPORTS

  This session describes a text building program generating RUNOFF input. While a computer-controlled experiment is performed, experimental details and results are stored in various files. When the experiment finishes, the text building program access these files and constructs sentences describing the experiment. Almost all information can be obtained from the data files. Missing information is gotten from user at the terminal. The output is passed through RUNOFF and reports that previously took days to complete are now available in minutes. (30 minutes, Orientation: General)

* SO YOU NEED TO OPTIMIZE YOUR F77 CODE

  Since F77 is a high level language, programs are often too large and too slow. By observing the code generated by the compiler, one can find FORTRAN code sequences which, while functionally the same, differ appreciably in size and speed. Even greater improvements can be realized by using a few F77-callable MACRO routines. (30 minutes, Orientation: Technical)

* VS: - A VARIABLE SEND-DATA DRIVER FOR RSX-11M

  In order to have variable size messages sent between tasks, a simple driver was written with operations designed to be similar to variable send-data used on RSX-11M Plus. In recent months, more general features have been added: internal message pool, "wait for message", named queues, and examine and delete messages. This session will present an overview of the VS: driver and examples of its use. (30 minutes, Orientation: Technical)

* OPTIMIZING A DIVERSIFIED SYSTEM

  As a newspaper, we are doing an unusual variety of applications on a single PDP-11/70 RSX-11M system: typesetting, interactive advertising layout, accounts receivable, on-line circulation, process control for a newsprint system, and COBOL, MACRO and FORTRAN development. As neophyte users, we faced a series of bewildering decisions about how to best protect and optimize our system. This session will present some of the mistakes we made and the solutions we found to make the system work safely and efficiently. We will touch on choices in SYSGEN, partition layout, disk ACP choices, and use of tools from the SIG tapes. (30 minutes, Orientation: General)

Eating and Dining at Atlanta

The area around the Hilton is loaded with good restaurants and bars catering to the professional crowd in Atlanta. They are big on happy hours, which usually run from 4 to 7 pm. Most resturants serve dinner till 10 pm, but tend to close suprisingly early, like by midnight.

The Hilton is located in downtown Atlanta and the area at night, particularly to the south, should not be explored alone. The Omni (about a 15 minute walk) is the big shopping area close by, so get a big group together and visit it one day or night.

The following is a list some restaurants and nighttime entertainments in and around the Hilton. The Hilton staff can help you further:

In the Hilton:

Cafe de La Paix	reputed for fine dining
Trader Vic's	good dining
Nicolas's Roof	one of the best restaurants in Atlanta, hard to get in to but will take reservations.

In the Hyatt Regency (about four blocks away):

Clock o' Fives	good
Hugo's	fine gourmet restaurant
Polaris	in revolving dome at top of Hyatt
Club Atlantis	live nightclub-style entertainment

Peachtree Center (one block away):

Midnight Sun	reputed to be excellent and expensive

The Peachtree Center also has a lot of fast-food lunch places for cheap dining. There are also a lot of small steak and ale places between the Hilton and Peachtree Center.

Peachtree Plaza (two blocks away):

Savannah Fish Company	good dinner and reasonable
Sundial Restaurant	another rotating resturant about 75 stories up - stunning and expensive
Inner Circle	nightclub

Others:

O. Henry's Steak House	has best happy hour in Atlanta and food is good
Daily's	A good happy hour, good food, piano bar and evening entertainment.
Herren's	reputation for prime rib, shrimp Arnaud, sticky buns.

Field Testing RSX-11M V4.0

Robert Bismuth
Redkite Software Limited
Swansea, Wales, U.K.

It all started when a tape arrived inside a box of photocopied manuals, labelled RSX11M V4.0. From this inocuous event came three months of very interesting product testing, which resulted in a better grasp of the scale and timing involved in the production of a new operating system base level. This article details the findings and feelings unearthed during that period.

First the configurations of the two systems used for field testing need to be detailed. Both systems used were mixed systems, featuring DEC and non-DEC peripheral/memory components. The primary system used centered around a (sometimes steam powered) PDP 11/40 with 128KW of memory, FIS and KW11P. The peripheral configuration used was: 2 x RX02, 5 x RK05, 1 x RP03 (lookalike Winchester), 1 x TU10, 8 x DL11C, 4 x DL11E and one auto-answer 300 baud modem. This system was the first SYSGENed under V4.0 and has successfully run from the first GEN to date.

The second system used was a 'home-brewed' LSI 11/23-PLUS (i.e. 22-bit Q-bus) with 1 mbyte of memory. The peripheral configuration here was: 2 x RX02, 2 x RK06 (lookalike), 1 x TU10, 4 x DLV11, 1 x DZV11 with one auto-answer 1200 baud modem. After some "teething" problems with the 22-bit technology, V4.0 came up and runs quite successfully. These two systems demonstrated to us the reliability of V4.0 on both new and 'old' types of system. In both cases, less installation trouble was experienced than with V3.2, and V4.0 correctly recognised both processor types and configuration.

The following are the main points worth noting which we discovered during the field test:

V4.0 Sysgen

From what we have been told, we were the only test site to perform a V4.0 SYSGEN with V3.2 as the host system. This was largely as a result of the lack of standalone time on our target system. Apart from frustration to us, this has a largely beneficial effect for the RSX community: we hit all incompatibilities very quickly.

Problems only arose out of V4.0's extended indirect command processor. Two new directives, .SETD (set numeric decimal) and .SETO (set numeric octal), together with the new command processor allowing <TAB> as undelimiter in .DATA statements accounted for all the trouble we encountered. Since reporting these problems, we have received SLP correction files for V3.2 indirect, to allow this new functionality - the intention is to include them in the release notes for V4.0. Even with these problems forcing us to restart SYSGEN three or four times, V4.0 was up and running within one and a half days of the release arriving!

The SYSGEN questions have not changed that much. New options have been added for new hardware, and SYSGEN now saves all answers for Phase 1 and 2, with the exception of the highest interrupt vector in use (warning: it defaults to 0!). SYSGEN now will build a working FCSRES, and also will link all tasks it builds to the library. However, this implies that SYSGEN has to generate the BLD and ODL files for all tasks which it builds: this lengthens SYSGEN by up to 45 minutes. Note that this "BLD the BLDs" process takes place, with no warning comments, in Phase 2, after it states which tasks will be built. The immediate visible appearance of this is "something's wrong - my terminals stopped printing!" Be ready for this - if you're lucky enough to possess a real PDP11 (i.e. one with front panel lights), you can go and watch indirect work.

Unfortunately, the V3.2 SET/COFFEEBREAK, SET/TEABREAK (which were conditionalized on clock line frequency) have gone: no light entertainment to relieve the philosophical moments of SYSGEN, just a professional process.

SAVE and BOOT

At our site, we write device drivers and ACP's for non-standard devices. We were affected by some changes to SAVE and BOOT. Their "special" device drivers have changed format, becoming more compact and "cleaner". SAVE is now overlayed: this should be studied carefully as TCB's are changed by SAVE during its operation in such a way as to prevent overlay loading. A further consequence is that SAVE now stops/blocks all other tasks during a system save to prevent accidental overlay loading.

Error Logging

Error logging for V4.0 has been completely rewritten. The modules in the executive have been moved "out" into the new directive common: sensible, since in an efficient system they should hardly be used. The executive components functioned well, i.e. they caught errors, logged them and did not crash the system. The field test issue of the new report generator did not function too well.

The documentation indicates the new error logging reports will be useful - they are certainly written and formatted in a fashion easily understood, possibly even believed, by field service personnel. Those which were able to be produced were good, unfortunately this was the one area of V4.0 in which we encountered problems. Each problem we reported was immediately fixed and will not occur in the final release. The developer's response made up for the code deficiencies we located: after all, finding problems is the function of field testing.

Digitally in-Compatible Language

DCL is provided with V4.0 as a SYSGEN option. MCR commands are always present under any V4.0 system, but development shops or teaching environments will certainly opt for the English (actually American) language command style of DCL. It has been implemented as a translation process: the non-privileged DCL task translates its command lines into the correct corresponding MCR command lines.

The biggest DCL problem we had was working through the documentation to find out how to enable it in the system after SYSGEN. Apart from this, we found a few minor problems, nothing serious. Being a translation process, it is not fast. However, the exec/terminal driver support required to enable a task to act as a command line interpreter is fully documented and supported. At last, 11M has user written CLI support!

DCL is user modifiable (through reassembly /link) but it is almost trivial to code simple CLIs and enable them in a host system. The real power of the CLI interface is that a CLI is a NON-PRIVILEGED task and it may be associated on a per terminal or per account basis.

Our only criticism of DCL is the lack of any documentation covering its "compatibility" with the other "standard" DCL's available under RT11, RSTS and VMS. Perhaps DEC could produce a "User Guide to Kernal DCL" for the community.

DSC and BRU

This release features new powerful standalone systems for both DSC and BRU which also contain the images of BAD and FMT. Apart from some new warning messages, no substantial problems were encountered with either DSC or BRU, but then we didn't "thrash" them fully because life is too short. Users should realize that V4.0 DSC and BRU are different from V3.2: BRU will read most V3.2 BRU tapes but V4.0 DSC will not read V3.2 DEC tapes at all! Our advice: keep copies of the older utilities available until either all old tapes have been converted or are not needed.

Conclusions

RSX11M V4.0 is a good, solid, well-engineered product. Even the field test release was better than the SDC release of V3.2. Our site had some communication problems with the developers, because of our geographical location, but we were impressed with the efforts made by the 11M group to contact us by telephone and solve any problems we had. There are many more points worth noting: a lot of the SIG wish list requests have been satisfied and several swapping/scheduling improvements have been made. While the new enlarged POOL should free up some systems, perhaps one of the best new environment features of V4.0 is, as some at our site put it: "It doesn't SHOUT AT US ANYMORE!" Yes, RSX11M now speaks in upper and lower case.

Idle Terminal Monitor

Bruce R. Mitchell
Engineering Systems and Technology
3M Company
St. Paul, MN 55144

On most development systems, a reasonably secure system is desired. RSX-11 is a fine tool for developing code for industrial applications, but its security regarding unattended terminals is not satisfactory. It is not desirable for anyone to be able to walk up to a terminal and possibly steal sensitive information or worse, if the terminal is privileged (as most are are on development systems) delete any and all files he wishes.

This monitor addresses this problem. It examines system driver data structures looking for logged-in terminals, and scans the active task list looking for the "owning terminal" for all active tasks. It gives idle terminals first, second and final warnings, and logs all forced logouts on the system console. The monitor serves all TT: devices on the system, and can be built to support DECnet HT: devices. VT100 support can be included. All time parameters are user-selectable.

Shown below are the prefix template file (USERPREO.MAC), the montior source code (USER.MAC), and the generation command file (USERGEN.CMD). The comments in each file more fully explain how the montior functions and the task is generated.

USERPREO.MAC

```
        .NLIST  BIN                     ; Disable binary code listing
        .IDENT  /X01.03/                ; Experimental
        .ENABL  LC                      ; Enable lower case
;
; USERPRE - User Terminal Activity Activty Monitor Prefix File
;
;       Companion source file to USER.MAC
;
; Authors:
;       Bruce R. Mitchell
;
;       Patterned from the POOL monitor task supplied by Digital Equipment
;       Corporation on RSX-11M-Plus V1.0 Autopatch E
;
; Source Site:
;       Engineering Systems and Technology Laboratory
;       3M Company, 3M Center, St. Paul, Minnesnota  55144
;
; Source Hardware and Operating System:
;       DEC PDP-11/70 under RSX-11M-Plus V1.0 Autopatch E
;
```

```
; Target Site:
;       Same
;
; Target Hardware and Operating System:
;       Same
;
; Revision History:
;       1-Jan-82  Source ripped out of SLPRE.MAC
;       2-Jan-82  Warning message bit definitions added

        .PAGE
;
;       Special characters
;
ESC = 33                        ; Escape
BEL = 7                         ; Bell
LF = 12                         ; Line feed
CR = 15                         ; Carriage return
;
;       Local assignments
;
LUN1 = 1                        ; LUN 1 is used for console I/O
LUN2 = 2                        ; LUN 2 is used for user terminal I/O
EFN1 = 1                        ; EFN 1 is used for all I/O
EFN2 = 2                        ; EFN 2 is used for for mark times
SPNEFN = 3                      ; EFN 3 is used for spawns
CVTPRM = 10012                  ; Conversion parameters for $CBTA
;
;       Bit masks for terminal and task characteristics
;
TM.LOG = 1                      ; Terminal logged in
TM.TSK = 2                      ; Task active on terminal
TM.MCR = 4                      ; Terminal CLI is MCR (not used)
TM.1ST = 10                     ; First warning message sent
TM.2ND = 20                     ; Second warning message sent
TM.3RD = 40                     ; Final warning message sent
;
;       Number of TT type terminals in system
;
NTT = D$$L11                    ; Start with number of DL11s in system

.IF DF D$$H11                   ; If there are DH11 async multiplexers
NTT = NTT + <D$$H11 * 16.>      ; Assume 16 line / DH11 and add them
.ENDC                           ; DF D$$H11

.IF DF D$$Z11                   ; If there are DZ11 async multiplexers
NTT = NTT + <D$$Z11 * 8.>       ; Assume 8 lines / DZ11 and add them
.ENDC                           ; DF D$$Z11

.IF DF D$$J11                   ; If there are DJ11 multiplexers
NTT = NTT + <D$$J11 * 16.>      ; Assume 16 lines / DJ11 and add them
.ENDC                           ; DF D$$J11

ITT = NTT - 1                   ; Used for 0-offset terminal counting
;
```

```
;       Number of HT type terminals in system (for DECnet support)
;
.IF DF RS.NSL                   ; If DECnet support is included

NHT = RS.NSL                    ; Start with number of DL11s in system
IHT = NHT - 1                   ; Used for 0-offset terminal counting

.ENDC                           ; DF RS.NSL
;
;       Time parameters
;
```

USER.MAC

```
        .TITLE  USERMN  User Terminal Monitor
        .IDENT  /V01.02/                ; DECUS release
        .ENABL  LC
        .NLIST  CND                     ; Don't list unsatisfied conditionals

        .SBTTL  Introduction
;
; USERMN - User Terminal Activity Monitor for RSX-11M/M+
;
;
; Authors:
;       Bruce R. Mitchell
;
;       Patterned from the POOL monitor task supplied by Digital Equipment
;       Corporation on RSX-11M-Plus V1.0 Autopatch E
;
;       Thanks to Michael J. Lynch for abundant help with system internals
;
; Source Site:
;       Engineering Systems and Technology Laboratory
;       3M Company, 3M Center, St. Paul, Minnesnota  55144
;
; Source Hardware and Operating System:
;       DEC PDP-11/70 under RSX-11M-Plus V1.0 Autopatch E
;
; Target Site:
;       Same
;
; Target Hardware and Operating System:
;       Same
;
; Revision History:
;       31-Dec-81  First version written
;       3-Jan-82   Command file generates ASCII strings for times
;       5-Jan-82   In-house finished version released
;       9-Jan-82   DECnet HT: support conditionally added
;       11-Jan-82  DECUS release version
;       20-Jan-82  VT100 support conditionalized
```

```
        .PAGE
        .SBTTL  Description and Philosophy 101
;
;            Problem:  On most corporate development systems, a somewhat
;       secure operating system is desired.  While RSX-11 is a fine tool
;       for developing code for industrial  applications,  its  security
;       regarding  the matter of unattended terminals leaves a good deal
;       to be desired.
;
;            One solution to this problem is to force every user on  the
;       system  to log out at noon and 5 PM, thereby ensuring that those
;       terminals which were left unattended up to those times  are  now
;       secure.   However, in between these forced logouts, any fool can
;       walk up to a terminal and delete files, possibly steal sensitive
;       information, or worse, if the terminal is  privileged  (as  most
;       users  are  on  development systems) delete any and all files he
;       wishes.
;
;            This task addresses itself to this  problem.   It  examines
;       the  terminal  driver  data  structures  looking  for  logged-in
;       terminals.  It also scans the active task  list  and  notes  the
;       "owning terminal" for all active tasks.
;
;            The monitor then follows this procedure for each  logged-on
;       terminal:  (1)  If a terminal has no active tasks, bumps an idle
;       time counter for the terminal, or (2) if a terminal  has  active
;       tasks,  resets  the  idle time counter for the terminal.  If the
;       idle time counter shows  that  the  terminal  should  receive  a
;       first,  second  or  final  warning,  a  warning  is  sent to the
;       terminal via a breakthrough write.  If  the  idle  time  counter
;       shows  that  the  terminal has been idle the maximum permissible
;       time, the terminal is logged out and a message is printed on the
;       system console noting this transaction.
;
        .PAGE
        .SBTTL  Caveats and Restrictions
;
;            Those sites which do not use VT100s as their  primary  type
;       terminal should not select VT100 support in the build file.  The
;       selection of VT100 support forces escape sequences  onto  target
;       terminals causing blinking, intensity changes and screen clears.
;       These  VT100  sequences  cause  particular problems on VT52s and
;       VT100s in VT52 mode.
;
;       The "console" terminal, TT0:, is never logged out by the monitor
;       due  to  the  fact that every task installed at startup time has
;       TT0: as its owning terminal.  Obviously, since the user  monitor
;       is  installed and run at startup time, TT0: will always have the
;       user monitor active on it.  We do not feel that this is a  major
;       problem  since  TT0: is usually located in the computer room and
;       access to it is therefore (probably) restricted.
;
        .PAGE
        .SBTTL  Macro Calls and Definitions
;
```

```
;       System Definition Macros from LB:[1,1]EXEMC.MLB
;
        .MCALL  DCBDF$                  ; Device Control Block offsets
        .MCALL  HWDDF$                  ; Hardware register symbols
        .MCALL  TCBDF$                  ; Task Control Block offsets
        .MCALL  UCBDF$                  ; Unit Control Block offsets

        DCBDF$
        HWDDF$
        TCBDF$
        UCBDF$
;
;       Directive Macros from LB:[1,1]SYSLIB.OLB
;
        .MCALL  ALUN$                   ; Assign Logical Unit Number
        .MCALL  DIR$                    ; Execute directive
        .MCALL  GTIM$                   ; Get system clock time
        .MCALL  MRKT$                   ; Mark time
        .MCALL  QIOW$                   ; Queue I/O
        .MCALL  SPWN$                   ; Spawn task
        .MCALL  STSE$                   ; Stop for single event flag
        .MCALL  WTSE$                   ; Wait for single event flag

        .PAGE
        .SBTTL  Local Variables
;
;       Local variables
;
DATBUF: .BLKW   8.              ; BUFFER AREA FOR DATE AND TIME
;
;       Local terminal data storage area
;
;       Format:  Flags word - Idle counter
;
;       TT      Offset          TT      Offset          TT      Offset
;
;       0       0               30      140             60      300
;       1       4               31      144             61      304
;       2       10              32      150             62      310
;       3       14              33      154             63      314
;       4       20              34      160             64      320
;       5       24              35      164             65      324
;       6       30              36      170             66      330
;       7       34              37      174             67      334
;
;       10      40              40      200             70      340
;       11      44              41      204             71      344
;       12      50              42      210             72      350
;       13      54              43      214             73      354
;       14      60              44      220             74      360
;       15      64              45      224             75      364
;       16      70              46      230             76      370
;       17      74              47      234             77      374
;
;       20      100             50      240             100     400
;       21      104             51      244             101     404
;       22      110             52      250             102     410
;       23      114             53      254             103     414
;       24      120             54      260             104     420
;       25      124             55      264             105     424
;       26      130             56      270             106     430
;       27      134             57      274             107     434

TRMDAT:                                 ; Beginning of data storage area

.REPT   NTT                             ; Repeat for number of terminals

        .WORD   0                       ; Storage for terminal flags
        .WORD   0                       ; Storage for idle time counter

.ENDR                                   ; # NTT
;
;       DECnet Terminal Data Storage Area (Conditionally assembled)
;

.IF DF RS.NSL                           ; If number of HT: terminals defined

NETDAT:                                 ; Beginning of data storage area

.REPT   NHT                             ; Repeat for number of terminals

        .WORD   0                       ; Storage for terminal flags
        .WORD   0                       ; Storage for idle time counter

.ENDR                                   ; # NHT

.ENDC                                   ; DF RS.NSL
        .PAGE
        .SBTTL  Data Structures
;
;       Directive Data Structures
;
;       Assign LUN for a terminal warning or logout message

ASSLUN: ALUN$   2, TT, 0

;       Obtain the current time and date from the system

DATDPB: GTIM$   DATBUF                  ; DPB for obtaining current time

;       Mark time for the monitoring interval in seconds

TIMDPB: MRKT$   EFN2, SECNDS, 2         ; DPB for mark time

;       Print messages on terminals

BRODPB: QIOW$   IO.WBT, LUN2, EFN1,,,, < , >

;       Print logging message on the system console
```

```
CONDPB: QIOW$   IO.WBT, LUN1, EFN1,,,, <CONMSG, CONLEN>

;       Spawn MCR... to log out the terminal specified at address MCRTRM,
;       setting event flag SPNEFN on exit or status emission

MCRSPN: SPWN$   MCR...,,,,,, SPNEFN,,, MCRBYE, BYELTH,, TT
                                        ; Spawn MCR... for the terminal
                                        ; Set SPNEFN event flag on exit
                                        ; Command line stored in MCRBYE
                                        ; Length of command line is BYELTH
                                        ; Terminal type TT

MCRBYE: .ASCII  /BYE/                   ; MCR command line to log out terminal

BYELTH = . - MCRBYE                     ; Length of command line

        .EVEN

;       Stop self while waiting to examine terminals again

SLEEP:  STSE$   EFN2                    ; Go to sleep and wait for wakeup

;       Wait for the spawn event flag or timeout event flag (EFNs 2 and 3)

SPNWAI: WTSE$   SPNEFN                  ; Wait for SPAWN exit or timeout

        .PAGE
        .SBTTL  Messages
;
;       Messages
;
        .NLIST  BIN

;       This message is printed on the system console or logging device

CONMSG: .ASCII  <CR><LF>                ; Return carriage and line feed
TIME:   .BLKB   8.                      ; Storage for time string
        .ASCII  /  /                    ; Separating spaces

HEDLEN  = . - CONMSG                    ; HEDLEN is length of header area

        .ASCII  /Idle terminal forced logout on /
        .EVEN
LGODEV: .BLKW   1                       ; Space to print terminal type
LGOTRM: .BLKB   2                       ; Space to print terminal number
        .ASCII  /:/<CR><LF>

CONLEN = . - CONMSG

;       This is the first warning message printed on a user's terminal
;       On a VT100 time remaining will blink in bright mode

WARN1:  .ASCII  <CR><LF><LF><LF>
        .ASCII  /From USER monitor:  First idle terminal warning!/<CR><LF>
        .ASCII  /                    /
```

```
.IF DF V$T100                           ; If VT100 support selected
        .ASCII  <ESC><133><65><73><61><155>     ; $[5;1m
.ENDC                                   ; DF V$T100
        .EVEN
        .WORD   TIME1                   ; Remaining time
        .ASCII  / minutes/
.IF DF V$T100                           ; If VT100 support selected
        .ASCII  <ESC><133><60><155>     ; $[0m
.ENDC                                   ; V$T100
        .ASCII  / before forced logout/
        .ASCII  <CR><LF><LF><LF>

WRN1LN =. - WARN1                       ; Length of warning message

;       This is the second warning message printed on a user's terminal
;       On a VT100 time remaining will blink in bright mode

WARN2:  .ASCII  <CR><LF><LF><LF>
        .ASCII  /From USER monitor:  Second idle terminal warning!/<CR><LF>
        .ASCII  /                    /
.IF DF V$T100                           ; If VT100 support selected
        .ASCII  <ESC><133><65><73><61><155>     ; $[5;1m
.ENDC                                   ; V$T100
        .EVEN
        .WORD   TIME2                   ; Remaining time
        .ASCII  / minutes/
.IF DF V$T100                           ; If VT100 support selected
        .ASCII  <ESC><133><60><155>     ; $[0m
.ENDC                                   ; V$T100
        .ASCII  / before forced logout/
        .ASCII  <CR><LF><LF><LF>

WRN2LN =. - WARN2                       ; Length of warning message

;       This is the third warning message printed on a user's terminal
;       On a VT100 time remaining will blink in bright mode

WARN3:  .ASCII  <CR><LF><LF><LF>
        .ASCII  /From USER monitor:  Final idle terminal warning!/<CR><LF>
        .ASCII  /                    /
.IF DF V$T100                           ; If VT100 support selected
        .ASCII  <ESC><133><65><73><61><155>     ; $[5;1m
.ENDC                                   ; V$T100
        .ASCII  /1 minute/
.IF DF V$T100                           ; If VT100 support selected
        .ASCII  <ESC><133><60><155>     ; $[0m
.ENDC                                   ; V$T100
        .ASCII  / before forced logout/
        .ASCII  <CR><LF><LF><LF>

WRN3LN = . - WARN3                      ; Length of warning message

;       This is the message printed on a user's terminal at forced logout
;       On a VT100 it will home the cursor, clear the screen and blink
```

```
BANZAI: .ASCII  <CR><LF><LF><LF>
.IF DF V$T100                           ; If VT100 support selected
        .ASCII  <ESC><133><61><73><61><110>     ; $[1;1H
        .ASCII  <ESC><133><62><112>     ; $[2J
        .ASCII  <ESC><133><65><155>     ; $[5m
.ENDC                                   ; V$T100
        .ASCII  /From USER monitor:  Forced idle terminal logout/<CR><LF>
.IF DF V$T100                           ; If VT100 support selected
        .ASCII  <ESC><133><60><155>     ; $[0m
.ENDC                                   ; V$T100
        .ASCII  <CR><LF><LF><LF>

BANZLN = . - BANZAI                     ; Length of warning message

        .EVEN
        .LIST   BIN

        .PAGE
        .SBTTL
        .SBTTL  Mainline Code
        .SBTTL
        .SBTTL  TSKS    Search Active Task List
;
;       Mainline code
;
;       We just woke up; the first thing to do is scan the active task
;       list to find out which terminals have active tasks.  Before that,
;       though, we have to clear the logged-in and task active flags for
;       the terminals.
;
;       Note that the "physical unit number" in a TT's UCB is not really
;       the physical unit number for the device; it is the offset from
;       the base unit number for the TT's DCB.  Therefore we have to get
;       the true unit number by dividing the distance between the TT's
;       UCB and the DCB's base UCB by the length of the TT UCB, and then
;       adding the resulting number to the TT's DCB base unit number.
;
;       Since a number of tasks may be active on CO0:, VT1:, HT3: and so
;       on, a check is made for terminals of type TTnn: .  (If DECnet is
;       supported, a check is made for terminals of  type  HTnn: .)   No
;       action  is  taken if the terminal is not a TT: .  (or HT: in the
;       case of a DECnet terminal.)
;

USERMN:                                 ; Entry point

TSKS:   MOV     #<ITT*4>, R0            ; R0 counts down TT: data blocks

10$:    BIC     #<TM.LOG!TM.TSK>, TRMDAT(R0)    ; Clear terminal active flags
        SUB     #4, R0                  ; Point at flag for preceding unit
        BGE     10$                     ; If TT0: not processed, loop again

.IF DF RS.NSL                           ; If DECnet is supported

        MOV     #<IHT*4>, R0            ; R0 counts down HT: data blocks
```

```
20$:    BIC     #<TM.LOG!TM.TSK>, NETDAT(R0)    ; Clear terminal active flags
        SUB     #4, R0                  ; Point at flag for preceding unit
        BGE     20$                     ; If HT0: not processed, loop again

.ENDC                                   ; DF RS.NSL

;       Switch to system state and check active task list

        CALL    $SWSTK, COMPAR          ; Switch to system state; return
                                        ;  from it at COMPAR

        MOV     $ACTHD, R0              ;; R0 points to active task listhead

30$:    MOV     T.UCB(R0), R3           ;; R3 points at UCB for task's "TI:"
        MOV     U.DCB(R3), R1           ;; R1 points at DCB for task's "TI:"
        CMP     D.NAM(R1), #"TT         ;; Is this a terminal DCB?

.IF NDF RS.NSL                          ; If DECnet is not supported

        BNE     70$                     ;; If not, we don't care about it

.IFF                                    ; If DECnet is supported

        BNE     40$                     ;; If not, check for DECnet terminal

        CLR     R4                      ;; R4 clear indicates a TT: device
        BR      50$                     ;; Go process it

40$:    CMP     D.NAM(R1), #"HT         ;; Is this a DECnet terminal DCB?
        BNE     70$                     ;; If not, we don't care about it

        INC     R4                      ;; R4 nonzero indicates an HT: device

.ENDC                                   ; NDF RS.NSL

;       Calculate terminal number and set active task flag for it

50$:    SUB     D.UCB(R1), R3           ;; R3 is distance of UCB from 1st UCB
        CLR     R2                      ;; Clear 32-bit R2 extension word
        DIV     D.UCBL(R1), R2          ;; Divide distance by length of UCB
        MOVB    D.UNIT(R1), R1          ;; R1 is lowest unit # on DCB
        ADD     R1, R2                  ;; R2 is now terminal number
        ASL     R2                      ;; Multiply R2 by 4 for addressing
        ASL     R2                      ;;  (2 bytes/word * 2 words)

.IF NDF RS.NSL                          ; If DECnet not supported

        BIS     #TM.TSK, TRMDAT(R2)     ;; Set active task flag for terminal

.IFF                                    ; If DECnet is supported

        TST     R4                      ;; Is this an HT: or a TT: device?
        BNE     60$                     ;; If an HT: device, go process it

        BIS     #TM.TSK, TRMDAT(R2)     ;; Set active task flag for TT: device
```

```
        BR      70$                     ;; Go set up to do next task

60$:    BIS     #TM.TSK, NETDAT(R2)     ;; Set active task flag for HT: device

.ENDC                                   ;; NDF RS.NSL

;       Set up pointers for next active task and check for list end

70$:    MOV     T.ACTL(R0), R0          ;; Point to next active task
        TST     R0                      ;; Is this the list termination?
        BNE     30$                     ;; Loop back for more if not

        .PAGE
        .SBTTL  DCBS    Search DCB List
;
;       Now we know which terminals have tasks active on them.  The next
;       thing to do is find out which terminals are logged in.  Before that,
;       though, we have to set and clear the flags indicating which units
;       are logged in.
;

DCBS:   MOV     $DEVHD, R0              ;; R0 points to DCB listhead

10$:    CMP     D.NAM(R0), #"TT         ;; Is this a terminal DCB?

.IF NDF RS.NSL                          ; If DECnet not supported

        BNE     70$                     ;; If not, go get a new DCB address

.IFF                                    ; If DECnet is supported

        BNE     20$                     ;; If not, go check for DECnet HT:

        CLR     R4                      ;; R4 clear indicates a TT: device
        BR      30$                     ;; Go process it

20$:    CMP     D.NAM(R0), #"HT         ;; Is this a DECnet HT: DCB?
        BNE     70$                     ;; If not, go get a new DCB address
        INC     R4                      ;; R4 nonzero indicates an HT: device

.ENDC                                   ; NDF RS.NSL

30$:    MOVB    D.UNIT(R0), R1          ;; R1 is now lowest unit number on DCB
        MOVB    D.UNIT+1(R0), R3        ;; Highest unit number served by DCB
        SUB     R1, R3                  ;; R3 is number of units served - 1
        INC     R3                      ;; Now R3 is usable as a down counter
        MOV     R1, R2                  ;; R2 is now lowest unit number on DCB
        ASL     R2                      ;; Multiply it by 4 for addressing
        ASL     R2                      ;;  (2 bytes/word * 2 words)
        MOV     D.UCB(R0), R1           ;; R1 points to first terminal's UCB

;       Loop through all the units on this DCB, checking logged-in terminals

40$:    BIT     #U2.LOG, U.CW2(R1)      ;; Is this unit logged in?
        BNE     60$                     ;; If not, go look at next terminal
```

```
.IF NDF RS.NSL                                 ;; If DECnet not supported

         BIS      #TM.LOG, TRMDAT(R2)          ;; Set logged-in terminal flag

.IFF                                           ;; If DECnet is supported

         TST      R4                           ;; Is this a TT: or an HT: device?
         BNE      50$                          ;; If an HT: device, go process it

         BIS      #TM.LOG, TRMDAT(R2)          ;; Set logged-in flag for TT: device
         BR       60$                          ;; Go set up to do next device

50$:     BIS      #TM.LOG, NETDAT(R2)          ;; Set logged-in flag for HT: device

.ENDC                                          ;; NDF RS.NSL

60$:     ADD      #4, R2                       ;; R2 points at next term flag word
         ADD      D.UCBL(R0), R1               ;; R1 now points at next UCB on DCB
         SOB      R3, 40$                      ;; Loop for each UCB on each DCB

;        Finished with DCB, prepare to examine next DCB

70$:     MOV      D.LNK(R0), R0                ;; R0 now points at next DCB in list
         BNE      10$                          ;; If not list termination, loop back

         RETURN                                ;; No more DCBs, return to user state

         .PAGE
         .SBTTL   COMPAR  Crosscheck Tasks and Terminals
;
;        Back in user state.  Now we have to cross-check logged-in terminals
;        against active tasks, and bump the idle counter for those terminals
;        which had no tasks active on them.  For those terminals which had
;        tasks active on them, the idle counter has to be reset.
;

COMPAR:  MOV      #<ITT*4>, R0                 ; R0 counts down terminal data blocks

10$:     BIT      #TM.LOG, TRMDAT(R0)          ; Is this terminal logged in?
         BNE      20$                          ; If so, skip and process it

         CLR      TRMDAT+2(R0)                 ; Not logged in; clear idle counter
         BR       40$                          ; Go set up to do next data block

;        Terminal logged in; check for an active task on it

20$:     BIT      #TM.TSK, TRMDAT(R0)          ; Does it have an active task?
         BNE      30$                          ; If so, skip around inactive code

         INC      TRMDAT+2(R0)                 ; No active task; bump idle counter
         BR       40$                          ; Go set up to do next one

30$:     CLR      TRMDAT+2(R0)                 ; Active task; clear idle counter

;        Set up to do next terminal data block
```

```
40$:     SUB      #4, R0                       ; Point at flag word of preceding unit
         BGE      10$                          ; If TT0: not processed, loop again

.IF DF RS.NSL                                  ; If DECnet is supported

         MOV      #<IHT*4>, R0                 ; R0 counts down HT: device data blocks

50$:     BIT      #TM.LOG, NETDAT(R0)          ; Is this HT: logged in?
         BNE      60$                          ; If so, skip and process it

         CLR      NETDAT+2(R0)                 ; Not logged in; clear idle counter
         BR       80$                          ; Go set up to do next data block

;        HT: is logged in; check for an active task on it

60$:     BIT      #TM.TSK, NETDAT(R0)          ; Does it have an active task?
         BNE      70$                          ; If so, skip around inactive code

         INC      NETDAT+2(R0)                 ; No active task; bump idle counter
         BR       80$                          ; Go set up to do next one

70$:     CLR      NETDAT+2(R0)                 ; Active task; clear idle counter

;        Set up to do next HT: data block

80$:     SUB      #4, R0                       ; Point at flag word of preceding unit
         BGE      50$                          ; If TT0: not processed, loop again

.ENDC                                          ; DF RS.NSL

         .PAGE
         .SBTTL   CKTIME  Check Terminal Idle Time
;
;        Now we have to check terminals with nonzero idle time counters to
;        see if they rate warnings or a forced logout.  Those terminals with
;        idle time counters of zero can be ignored.  (If DECnet HT: device
;        checking support is included, these devices will be examined also.)
;
;        Idle time for idle terminals is compared against the genration-time
;        limits.  If the idle time is greater than the limits, messages are
;        sent or logouts are spawned by the appropriate subroutines.
;

CKTIME:  CLR      R1                           ; R1 is terminal number
         CLR      R2                           ; R2 is offset in terminal data block
         MOV      #"TT, ASSLUN+A.LUNA          ; Set device to TT for ALUN$
         MOV      #"TT, MCRSPN+S.PWDN          ; Set device to TT for SPWN$
         MOV      #"TT, LGODEV                 ; Set device to TT for console message

10$:     TST      TRMDAT+2(R2)                 ; Is this an idle terminal?
         BEQ      60$                          ; If not, set up to examine next term

;        Terminal is idle; check against time limits for warnings and logout

         CMP      TRMDAT+2(R2), #FIRST         ; Idle less than first warning time?
```

```
        BGE     20$                     ; If not, go do something about it

        BR      60$                     ; If so, set up to examine next term

;       Check against logout time limit

20$:    CMP     TRMDAT+2(R2), #FOURTH   ; Does idle time indicate a logout?
        BLT     30$                     ; If not, go check for final warning

        CALL    KILLIT                  ; If so, go log it out
        BR      60$                     ; Go set up to process next terminal

;       Check against final warning time limit

30$:    CMP     TRMDAT+2(R2), #THIRD    ; Does idle time indicate final warn?
        BLT     40$                     ; If not, go check for second warning

        CALL    WARFIN                  ; If so, go issue the final warning
        BR      60$                     ; Go set up to process next terminal

;       Check against second warning time limit

40$:    CMP     TRMDAT+2(R2), #SECOND   ; Does idle time indicate second warn?
        BLT     50$                     ; If not, go issue first warning

        CALL    WARTWO                  ; If so, go issue the second warning
        BR      60$                     ; Go set up to process next terminal

;       Issue first warning

50$:    CALL    WARONE                  ; Issue the first warning

;       Check for last terminal and setup for next terminal

60$:    CMP     R1, #ITT                ; Has the last terminal been done?

.IF NDF RS.NSL                          ; If DECnet is not supported

        BGE     140$                    ; If so, exit this loop

.IFF                                    ; If DECnet is supported

        BGE     70$                     ; Go process DECnet HT: devices

.ENDC                                   ; NDF RS.NSL

        INC     R1                      ; R1 is now number of next terminal
        ADD     #4, R2                  ; R2 is now offset to next data block
        BR      10$                     ; Not finished yet, loop

.IF DF RS.NSL                           ; If DECnet is supported

70$:    CLR     R1                      ; R1 is HT: device number
        CLR     R2                      ; R2 is offset in HT: data block
        MOV     #"HT, ASSLUN+A.LUNA     ; Set device to HT for ALUN$
```

```
        MOV     #"HT, MCRSPN+S.PWDN     ; Set device to HT for SPWN$
        MOV     #"HT, LGODEV            ; Set device to HT for console message

80$:    TST     NETDAT+2(R2)            ; Is this an idle HT: terminal?
        BEQ     130$                    ; If not, set up to examine next term

;       HT: is idle; check against time limits for warnings and logout

        CMP     NETDAT+2(R2), #FIRST    ; Idle less than first warning time?
        BGE     90$                     ; If not, go do something about it

        BR      130$                    ; If so, set up to examine next HT:

;       Check against logout time limit

90$:    CMP     NETDAT+2(R2), #FOURTH   ; Does idle time indicate a logout?
        BLT     100$                    ; If not, go check for final warning

        CALL    KILLIT                  ; If so, go log it out
        BR      130$                    ; Go set up to process next HT:

;       Check against final warning time limit

100$:   CMP     NETDAT+2(R2), #THIRD    ; Does idle time indicate final warn?
        BLT     110$                    ; If not, go check for second warning

        CALL    WARFIN                  ; If so, go issue the final warning
        BR      130$                    ; Go set up to process next HT:

;       Check against second warning time limit

110$:   CMP     NETDAT+2(R2), #SECOND   ; Does idle time indicate second warn
        BLT     120$                    ; If not, go issue first warning

        CALL    WARTWO                  ; If so, go issue the second warning
        BR      130$                    ; Go set up to process next HT:

;       Issue first warning

120$:   CALL    WARONE                  ; Issue the first warning

;       Check for last HT: and setup for next HT:

130$:   CMP     R1, #IHT                ; Has the last HT: been done?
        BGE     140$                    ; If so, exit this loop

        INC     R1                      ; R1 is now number of next HT:
        ADD     #4, R2                  ; R2 is now offset to next data block
        BR      80$                     ; Not finished yet, loop

.ENDC                                   ; DF RS.NSL

;       All terminals are processed; go back to sleep

140$:   DIR$    #TIMDPB                 ; Set marktime for monitoring wakeup
```

```
        DIR$    #SLEEP                  ; Hide from users and go to sleep

        JMP     TSKS                    ; Go take another look at everybody

        .PAGE
        .SBTTL
        .SBTTL  Subroutines
        .SBTTL
        .SBTTL  DATSUB  Create ASCII Time
;
;       DATSUB - Obtain the Current Time and Convert to ASCII
;
;       Inputs:  None
;
;       Outputs:  TIME gets ASCII time
;
;       All registers used are saved and restored
;

DATSUB: MOV     R0, -(SP)               ; Push volatile registers on stack
        MOV     R1, -(SP)               ;
        MOV     R2, -(SP)               ;
        DIR$    #DATDPB                 ; Execute get time and date directive
        MOV     #TIME, R0               ; Load ASCII time field address in R0
        MOV     #DATBUF+G.TIHR, R1      ; Load binary time field address in R1
        MOV     #3, R2                  ; Specify HH:MM:SS format
        CALL    $TIM                    ; Convert time to ASCII
        MOV     (SP)+, R2               ; Pop volatile registers off stack
        MOV     (SP)+, R1               ;
        MOV     (SP)+, R0               ;

        RETURN                          ; Return to caller

        .PAGE
        .SBTTL  KILLIT  Log Out an Idle Terminal
;
;       KILLIT - Log Out an Idle Terminal
;
;       Inputs:  R1 - Number of target terminal
;                R2 - Pointer to terminal data block
;
;       Outputs:  Terminal data flag word is cleared
;
;       Registers used are not saved and restored
;

KILLIT: MOV     R1, MCRSPN+S.PWVT       ; Load terminal number in SPWN$ DPB
        DIR$    #MCRSPN                 ; Execute directive to log it out
        BCC     5$                      ; If successful, continue

        RETURN                          ; If spawn failed, just forget it

5$:     DIR$    #SPNWAI                 ; Wait for spawn complete

        MOV     R1, ASSLUN+A.LUNU       ; Load terminal number in ALUN$ DPB
```

```
        DIR$    #ASSLUN                 ; Assign LUN 2 to target terminal
        BCC     10$                     ; If it succeeded, go issue message

        RETURN                          ; If LUN assign failed, just forget it

;       Log the forced logout on the target terminal

10$:    MOV     #BANZAI, BRODPB+Q.IOPL  ; Load message address into QIO DPB
        MOV     #BANZLN, BRODPB+Q.IOPL+2; Load message length into QIO DPB
        DIR$    #BRODPB                 ; Send logout message

        CLR     TRMDAT(R2)              ; Clear all flags for terminal

;       Log the forced logout on the system console

        CALL    DATSUB                  ; Load console message time and date
        MOV     R0, -(SP)               ; Save volatile registers on stack
        MOV     R1, -(SP)               ; yup
        MOV     R2, -(SP)               ; yup
        MOV     #0, LGOTRM              ; Null fill terminal number
        MOV     #LGOTRM, R0             ; Move address of output field to R0
        CLR     R2                      ; Specify suppression of leading 0s
        CALL    $CBOMG                  ; Convert binary TT number to ASCII
        MOV     (SP)+, R2               ; Restore volatile registers
        MOV     (SP)+, R1               ; yup
        MOV     (SP)+, R0               ; yup
        DIR$    #CONDPB                 ; Print logging message on console

        RETURN                          ; Return to caller

        .PAGE
        .SBTTL  WARONE  Issue First Warning
;
;       WARONE - Issue First Warning to a Terminal
;
;       Inputs:  R1 - Target terminal number
;                R2 - Pointer to terminal data block
;
;       Outputs:  Terminal flag word is updated
;
;       Registers used are not saved and restored
;

WARONE: BIT     #TM.1ST, TRMDAT(R2)     ; Has the first message been sent?
        BEQ     10$                     ; If not, go issue it

        RETURN                          ; If so, return to caller

10$:    MOV     R1, ASSLUN+A.LUNU       ; Load terminal number in ALUN$ DPB
        DIR$    #ASSLUN                 ; Assign LUN 2 to target terminal
        BCC     20$                     ; If it succeeded, issue the message

        RETURN                          ; If LUN assign failed, just forget it

20$:    MOV     #WARN1, BRODPB+Q.IOPL   ; Load message address into QIO DPB
```

```
        MOV     #WRN1LN, BRODPB+Q.IOPL+2; Load message length into QIO DPB
        DIR$    #BRODPB                 ; Send warning message
        BIS     #TM.1ST, TRMDAT(R2)     ; Set first message sent flag

        RETURN                          ; Return to caller

        .PAGE
        .SBTTL  WARTWO  Issue Second Warning
;
;       WARTWO - Issue Second Warning to a Terminal
;
;       Inputs:  R1 - Target terminal number
;                R2 - Pointer to terminal data block
;
;       Outputs:  Terminal flag word is updated
;
;       Registers used are not saved and restored
;

WARTWO: BIT     #TM.2ND, TRMDAT(R2)     ; Has the second message been sent?
        BEQ     10$                     ; If not, go issue it

        RETURN                          ; If so, return to caller

10$:    MOV     R1, ASSLUN+A.LUNU       ; Load terminal number in ALUN$ DPB
        DIR$    #ASSLUN                 ; Assign LUN 2 to target terminal
        BCC     20$                     ; If it succeeded, issue the message

        RETURN                          ; If LUN assign failed, just forget it

20$:    MOV     #WARN2, BRODPB+Q.IOPL   ; Load message address into QIO DPB
        MOV     #WRN2LN, BRODPB+Q.IOPL+2; Load message length into QIO DPB
        DIR$    #BRODPB                 ; Send warning message
        BIS     #TM.2ND, TRMDAT(R2)     ; Set second message sent flag

        RETURN                          ; Return to caller

        .PAGE
        .SBTTL  WARFIN  Issue Final Warning
;
;       WARFOM - Issue Final Warning to a Terminal
;
;       Inputs:  R1 - Target terminal number
;                R2 - Pointer to terminal data block
;
;       Outputs:  Terminal flag word is updated
;
;       Registers used are not saved and restored
;

WARFIN: BIT     #TM.3RD, TRMDAT(R2)     ; Has the final message been sent?
        BEQ     10$                     ; If not, go issue it

        RETURN                          ; If so, return to caller
```

```
10$:    MOV     R1, ASSLUN+A.LUNU       ; Load terminal number in ALUN$ DPB
        MOV     R1, ASSLUN+A.LUNU       ; Load terminal number in ALUN$ DPB
        DIR$    #ASSLUN                 ; Assign LUN 2 to target terminal
        BCC     20$                     ; If it succeeded, go issue message

        RETURN                          ; If LUN assign failed, just forget it

20$:    MOV     #WARN3, BRODPB+Q.IOPL   ; Load message address into QIO DPB
        MOV     #WRN3LN, BRODPB+Q.IOPL+2; Load message length into QIO DPB
        DIR$    #BRODPB                 ; Send warning message
        BIS     #TM.3RD, TRMDAT(R2)     ; Set final message sent flag

        RETURN                          ; Return to caller

        .END    USERMN
```

USERGEN.CMD

```
.;
.; User Monitor Generation Command File
.;
.SETS VERSN "1.12"
.ENABLE SUBSTITUTION
.SETS UIC <UIC>
;
;       User terminal monitor program generation procedure
;       Version 'VERSN'
;
;       This user terminal monitor program is a  privileged  task  which
;       continuously  runs  under  RSX-11M  to monitor the status of all
;       logged-on terminals.  It is particularly useful for cleaning  up
;       otherwise unusable areas of POOL which might be allocated when a
;       terminal logs in and would not be released until it logs out.
;
;       This monitor program  periodically  runs  through  the  terminal
;       driver data structures looking for logged on terminals.  It then
;       runs through the  active  task  list  looking  for  the  "owning
;       terminal" for each task.
;
.ASKS NULL Use carriage return to proceed
;
;       If a check of the active  task  list  finds  no  activity  on  a
;       terminal,  a  counter  is  bumped  to indicate an idle terminal;
;       contrariwise, if the terminal is found to have active tasks, the
;       counter is reset.
;
;       If  no  activity  occurs  on  the  terminal  within  the   first
;       monitoring period, a warning is sent to the terminal.
;
;       If no activity occurs on the terminal after  tne  first  warning
;       during the second monitoring period, a second warning is sent to
;       the terminal.
;
```

```
;       If no activity occurs on the terminal after the second warning up
;       to  one  minute  before  the  expiration  of the third monitoring
;       period, a final warning is sent.  If no  activity  occurs  during
;       the third monitoring period, the user monitor spawns BYE for that
;       terminal.
;
.ASKS NULL Use carriage return to proceed
;
;
;       Option Generation Phase
;
;
;       Specify values in WHOLE MINUTES for these parameters:
;
.ASKN [2:33.:10.] MONP1 Enter the length of the first monitoring period
;
.ASKN [2:33.:10.] MONP2 Enter the length of the second monitoring period
;
.ASKN [2:33.:10.] MONP3 Enter the length of the third monitoring period
;
;       Specify a value in WHOLE SECONDS for this parameter:
;
.TIME:
.ASKN [5.:60.:15.] SECNDS Enter the time between task list checks
.;
.; Check to see that the number given divides evenly into a minute
.;
.SETN TEMP 60./SECNDS
.SETN TEMP 60.-(TEMP*SECNDS)
.IF TEMP EQ 0 .GOTO OK
;
;       The number must divide evenly into one minute.
;       Valid numbers are:  5, 6, 10, 12, 15, 20, 30, 60
;
.GOTO TIME
.OK:
.;
.;      Compute the number of monitoring periods for warning and logout times
.;
.SETN NO1 (MONP1*60.)/SECNDS
.SETS $NO1 "'MONP1'"
.SETN TEMP MONP1+MONP2
.SETN NO2 (TEMP*60.)/SECNDS
.SETS $NO2 "'TEMP'"
.SETN TEMP MONP1+MONP2+MONP3-1
.SETN NO3 (TEMP*60.)/SECNDS
.SETS $NO3 "'TEMP'"
.INC TEMP
.SETN NO4 (TEMP*60.)/SECNDS
.SETS $NO4 "'TEMP'"
.;
.;      Create string values for warning time messages
.;
.SETN WARN2 TEMP-MONP2-MONP1
.SETS $WARN2 "'WARN2'"
```

```
.SETS $WN2 "0"
.TEST $WARN2
.IF <STRLEN> EQ 2 .SETS $WN2 "''"+$WARN2[2:2]
.SETS $WARN2 "TIME2 = ''"+$WARN2[1:1]+" + <"+$WN2+"*256.>"
.SETN WARN1 TEMP-MONP1
.SETS $WARN1 "'WARN1'"
.SETS $WN1 "0"
.TEST $WARN1
.IF <STRLEN> EQ 2 .SETS $WN1 "''"+$WARN1[2:2]
.SETS $WARN1 "TIME1 = ''"+$WARN1[1:1]+" + <"+$WN1+"*256.>"
;
.ASK DECNET Is DECnet supported on this machine
.IFF DECNET .GOTO UICGET
;
.ASK NETSUP Do you want idle DECnet HT: terminals logged out
.IFF NETSUP .GOTO UICGET
;
;       The user monitor decides how many DECnet HT: terminals are in
;       your system by examining RMHPRE.MAC.  This file has to be in
;       with your DECnet subsystem files somewhere.
;
.ASKS NETUIC Enter the UFD on LB: where RMHPRE.MAC can be found [ggg,mmm]
.UICGET:
;
;       To build the user monitor, it is almost necessary that a copy
;       of RSXMC.MAC be present on LB: .  If one is not available, you
;       will have to define the symbols for your terminal controllers
;       manually in a dummy RSXMC.MAC file somewhere on LB: .
;
.ASKS RSXUIC Enter the UFD on LB: where RSXMC.MAC can be found [ggg,uuu]
;
;       Idle terminal logouts are printed on the statistic logging device,
;       usually CL: or CO: .  However, you may direct them anywhere your
;       heart desires.
;
.ASKS CONSOL Enter the console device for statistic logging (DDn:)
;
;       You can include VT100 cursor control sequences which will do
;       some fancy things if the majority of your terminals are VT100
;       compatible terminals.
;
.ASK VT100 Do you want VT100 support in the user monitor
;
.ASK LIST Do you want a listing of the user monitor
;
.ASK MAP Do you want a map of the user monitor
;
;
;       Assemble and Build Phase
;
.;
.;      Set up MACRO assembly prefix file from data given
.;
PIP USERPRE.MAC;*/DE
PIP USERPRE.MAC=USERPRE0.MAC
```

```
.OPENA USERPRE.MAC
.ENABLE DATA
SECNDS = 'SECNDS'.                              ; Task monitoring interval
FIRST = 'NO1'.                                ; First warning time '$NO1' mins
SECOND = 'NO2'.                               ; Second warning time '$NO2' mins
THIRD = 'NO3'.                                ; Third warning time '$NO3' mins
FOURTH = 'NO4'.                               ; Logout time '$NO4' mins
'$WARN1'                                      ; ASCII time left at first warning
'$WARN2'                                      ; ASCII time left at second warning
.DISABLE DATA
.IFT VT100 .DATA V$T100 = 0                     ; VT100 support included
.CLOSE USERPRE.MAC
;
;       Assemble the source file
;
Pip USER.OBJ;*/de
.SETS LST ""
.IFT LIST .SETS LST ",USER/-SP"
.OPEN USERMAC.CMD
.DATA SY:USER'LST' = LB:[1,1]EXEMC/ML, LB:'RSXUIC'RSXMC/PA:1, -
.IFT NETSUP .DATA LB:'NETUIC'RMHPRE/PA:1, -
.DATA SY:'UIC'USERPRE/PA:1, -
.DATA SY:'UIC'USER
.CLOSE USERMAC.CMD
MAC @USERMAC
PIP USERMAC.CMD;*, USERPRE.MAC;*/NM/DE
;
;       Build the task
;
PIP USERBLD.CMD;*/DE
.SETS $MAP ""
.IFT MAP .SETS $MAP ", SY:'UIC'USER/MA/-SP"
.OPEN USERBLD.CMD
.DATA LB:[1,54]USER/PR/-CP '$MAP'=
.DATA SY:'UIC'USER, LB:[1,54]RSX11M.STB/SS
.DATA /
.DATA UNITS=2
.DATA TASK=USERMN
.DATA ASG='CONSOL'1
.DATA PRI=150
.DATA //
.CLOSE
SET /UIC=[1,54]
PIP USER.TSK;*/PR:0/FO
PIP USER.TSK;*/NM/DE
TKB @'UIC'USERBLD
;
;       Clean up the mess
;
PIP SY:'UIC'USER.MAP;*/PR/FO
SET /UIC='UIC'
PIP USER.LST, USER.MAP/PU
PIP USER.OBJ;*, USERBLD.CMD;*/NM/DE
;
;       Remember to install and run the new user monitor at startup time
```

Fortran Wild-Card Filename Access

Victor DiCara
Programming Concepts, Inc.
Coram, NY 11727

The tenth most popular itme on the RSX/IAS SIG 1981 Menu is a wild-card find facility. The following is a set of Fortran callable routines which provide this facility - but without wildcard directory support. I have been using the programs for some time without problems. Typical usage would be as follows. The listing of the subroutines below has more documentation on the usage.

```
            call ASNWLD (1,'*.*;*')         !Assign wild-card string

    10      call NXTWLD (1,,ierr)           !Get next file to process
            if (ierr.NE.20) goto 20         !If no more files, exit loop

            call GFNAME (1,string)          !Get next filename into string
            call ASSIGN (1,string)          !Open file for processing

            ... Process file ...

            goto 10                         !Loop for next file

    20      continue
```

From the Editor

Documentation on the PIPUTL.OLB wild-card routines was published in the Multi-Tasker in December 1978 (Volume 9, Number 10). While these routines do allow wild-card directories, they are not Fortran-callable. Also the article was for RSX-11M V3.1 version of PIPUTL.OLB. I do not know if the routines have changed any.

```
        .TITLE  $WILD - WILD CARD FILE NAME PROCESSING

        .IDENT  /032581/

        .MCALL  CSI$,CSI$1,CSI$2,FDOF$L
        FDOF$L                  ; GET THE FCS SYMBOL DEFINITIONS
;
; FUDGE THE "F4" PREFIX EQUATES REQUIRED.
;
        D.FDB   = 12.           ; START OF THE RSX-11M FDB PROPER.
        FILPTR  = 30            ; DISPLACEMENT TO THE FILE POINTER WORD
```

```
;
;
;       THIS COLLECTION OF ROUTINES IS USED TO ACCESS FILES WITHIN A
; DIRECTORY USING THE WILD CARD SPECIFICATION TECHNIQUES. THE GENERAL
; OUTLINE OF THE USAGE IS AS FOLLOWS:
;
; CALL ASNWLD(LUN,NAME)           THIS SPECIFIES THE WILD CARD TEMPLATE
;                                 AND INITIALIZES THE FDB FOR REPETATIVE
;                                 DIRECTORY NAME RETRIEVALS. NOTE: THE
;                                 3 PARTS SHOULD ALWAYS BE SPECIFIED, OR
;                                 ELSE ERRATIC RESULTS CAN OCCUR. IN PARTICULAR
;                                 IF THE VERSION IS NOT GIVEN, THE SEARCH
;                                 WILL FIND THE NEWEST VERSION OF THE NAME
;                                 WHICH MATCHES THE OTHER CRITERIA, AND THEN
;                                 FIX ON THAT FILE'S VERSION NUMBER AS
;                                 THE REQUIREMENT FOR ALL REMAINING FILES.
;
; CALL NXTWLD(LUN,[VER],IERR)     THIS ROUTINE IS INVOKED REPETATIVELY TO
;                                 SET UP THE LUN FOR EACH SUCCESSIVE FILE
;                                 FROM THE DIRECTORY. THE IERR RETURN VALUE
;                                 INDICATES WHETHER ANOTHER FILE EXISTS (=1)
;                                 OR WHETHER THE END OF DIRECTORY HAS
;                                 BEEN REACHED (=-10). THE OPTIONAL [VER]
;                                 PARAMETER IS USED TO RESET THE SEARCHED FOR
;                                 VERSION NUMBER. THIS IS NECESSARY WHEN
;                                 SEARCHING FOR VER 0 ( NEWEST ) OR
;                                 VERSION -1 (OLDEST).
;
; CALL GFNAME(LUN,NAME)           THIS ROUTINE IS USED TO RETRIEVE THE
;                                 ASCII FILE NAME, EXTENSION AND VERSION
;                                 NUMBER FROM THE SPECIFIED LUN. A FILE
;                                 MAY OR MAY NOT CURRENTLY BE OPEN ON
;                                 THE LUN. THE NAME IS RETURNED IN THE
;                                 FORMAT: <NAME>.<EXT>;<VER>
;                                 THERE ARE NO EMBEDDED BLANKS.
;
; CALL CLOSE(LUN)                 THIS ROUTINE IS USED TO TERMINATE THE
;                                 WILD CARD ACCESS ON THE GIVEN LUN.
;
;
        .PAGE
        .PSECT  WILD,GBL,CON

ASNWLD::
;
;       THIS ROUTINE FUNCTIONS IN A MANNER SIMILAR TO THE STANDARD OTS
; ASSIGN SUBROUTINE, EXCEPT THAT IT CAN HANDLE WILD CARD OPERATIONS.
;
; CALL ASNWLD(LUN,STRING,[,LEN])
;
;       LUN = LOGICAL UNIT NUMBER TO USE
;       STRING = FILE NAME STRING
;       LEN = (OPT) LENGTH OF STRING.
;
        MOV     (R5)+,R4        ;GET THE NUMBER OF ARGS
        MOV     @(R5)+,R2       ;GET THE LUN NUMBER
```

```
        MOV     @#$OTSV,R3      ; FORTRAN IMPURE AREA POINTER

        JSR     PC,$GETFILE     ;SET THE ADDRESS OF FILPTR
        BPL     1$              ;BRANCH IF FILE UNOPENED
        TRAP    128.+34.        ;VERY BAD TO CHANGE FDB WITH OPEN FILE
        BR      EXITR           ;JUST EXIT
1$:     JSR     PC,GETARG       ;GET THE ADDRESS OF THE NEXT ARG
        BEQ     EXITR           ;EXIT LEAVING DEFAULT SETTING IF NULL
2$:     MOV     R2,R1           ;SAVE ADDRESS OF FILE SPEC
        CLR     -(SP)           ;SET INIT LENGTH OF FILE SPEC
        DECB    R4              ;IS THERE A LENGTH ARG?
        BEQ     3$              ;BRANCH IF NOT TO SCAN
        JSR     PC,GETCNT       ;GO GET ITS ADDRESS
        BEQ     3$              ;BRANCH IF NULL
        MOV     @R2,@SP         ;SET GIVEN COUNT AND GO PROCESS
        BNE     4$              ;USE NON-ZERO LENGTH
3$:     MOV     R1,R2           ;MAKE COPY OF FILE SPEC ADDRESS
6$:     INC     @SP             ;BUMP CHAR COUNT
        TSTB    @R2             ;SEE IF CHAR NULL
        BEQ     5$              ;WE'VE GOT COUNT +1
        CMPB    #' ,(R2)+       ;SEE IF BLANK?
        BNE     6$              ;BRANCH IF NOT TO FIND ENDOF STRING
5$:     DEC     @SP             ;REMOVE +1 FROM CHAR COUNT
4$:     MOV     (SP)+,R2        ;PUT FILE SPEC LENGTH IN R2
        BLE     8$              ;BRANCH IF LENGTH ERROR
        JSR     PC,FNBST        ;SET UP THE FNB
        BCC     EXITR           ; SKIP IF GOOD FILE NAME
8$:     TRAP    128.+43.        ;BAD FILE SPECIFICATION
        BR      EXITR

;
EXITR:  CLR     FILPTR(R3)      ;RELEASE I/O SYSTEM
        RTS     PC

GETARG: DECB    R4              ;DECREMENT ARG COUNT
        BLE     DONE            ;DONE WHEN WE HIT 0 (OR LESS??)
GETCNT: MOV     (R5)+,R2        ;GET ADDRESS OF NEXT ARG
        CMP     #-1,R2          ;SET CONDITION CODE FOR NULL ARGS
        RTS     PC              ;RETURN
DONE:   TST     (SP)+           ;REMOVE SUBR RETURN ADDRESS
        BR      EXITR           ;FREE I/O SYSTEM AND EXIT

        .PAGE
;
; FNBST
;
;       SET UP THE FNB FOR THE STRING PASSED.
;
;   INPUTS
;       R1 - STRING ADDRESS OF DATA
;       R2 - LENGTH OF STRING
;       R3 - WORK AREA POINTER
;
;   OUTPUTS
;       VALUE - NONE.
;       FNB IS SET UP
;       R0 IS DESTROYED
;
FNBST:
        JSR     R5,.SAVR1               ;SAVE THE REGISTERS
        MOV     R2,R0                   ;GET STRING LENGTH
        INC     R0                      ;ROUND IT
        BIC     #1,R0                   ;TO THE NEAREST WORD
        MOV     SP,R5                   ;SAVE CURRENT SP
        SUB     R0,SP                   ;NOW RESERVE SOME SPACE ON THE STACK
                                        ;FOR THE TEXT
        MOV     SP,R3                   ;GET ADDRESS OF TEMP STORAGE
        MOV     R2,R0                   ;COUNT OF ITEMS TO MOVE
1$:     MOVB    (R1)+,(R3)+             ;MOVE THE STRING
        SOB     R0,1$                   ;ONE BYTE AT A TIME
        MOV     SP,R1                   ;NEW STRING ADDRESS
        MOV     #C.SIZE/2,R3            ;CSI CONTROL BLOCK
3$:     CLR     -(SP)                   ;       IS ALLOCATED ZEROED
        SOB     R3,3$
        CSI$1   SP,R1,R2                ;INIT CONTROL BLOCK
        BCS     2$                      ;EXIT IN ERROR
        CSI$2   R0,OUTPUT               ;SET UP THE NEEDED DSPT
        BCS     2$                      ;BRANCH IF ALL NOT WELL
        SEC                             ;IN CASE WE TAKE QUICK EXIT
        BITB    #CS.MOR,C.STAT(SP) ;IS IT LEGAL?
        BNE     2$                      ;NO
        BITB    #CS.WLD,C.STAT(SP)      ; WILD CARD GIVEN?
        BEQ     2$                      ; NO, THAT'S A MISTAKE.
        MOV     @#$OTSV,R3              ; FORTRAN IMPURE AREA POINTER
        MOV     FILPTR(R3),R0           ;RESTORE FDB ADDRESS
        ADD     #D.FDB,R0               ;POINT TO FDB PROPER
        MOV     R0,R1                   ;POINT TO
        ADD     #F.FNB,R1               ; FNB
        MOV     SP,R2                   ;SET ADDRESS OF DSPT
        ADD     #C.DSDS,R2              ;DATA SET DECSRIPTOR ADDRESS
        CLR     R3                      ;NO DEFAULT FNB
        JSR     PC,.PARSE               ;
2$:     MOV     R5,SP                   ;REMOVE STUFF FROM STACK
        RTS     PC                      ;RETURN

        .PAGE
;
; NXTWLD -
;
;       THIS ROUTINE FILLS IN THE FILE NAME BLOCK FROM THE DIRECTORY,
; AS INITIALIZED BY ASNWLD. FILE NAMES WILL BE RETRIEVED IN THE DIRECTORY
; (PIP) ORDER.
;
; CALL NXTWLD(LUN,[VER],IERR)
;
;       LUN = LOGICAL UNIT NUMBER IN USE.
;       VER = OVERRIDE VERSION NUMBER (OPTIONAL)
;       IERR = RETURN CODE OF OPERATION:
;               1 = OK TO OPEN
;             -10 = END OF DIRECTORY FOR GIVEN WILD CARDS.
```

```
;
NXTWLD::
        CMPB    (R5),#3         ; WE SHOULD HAVE THREE PARAMETERS
        BEQ     1$              ; CONTINUE IF ALL IS WELL.
        TRAP    128.+80.        ; WRONG NUMBER OF ARGS
        RTS     PC              ; RETURN, IF ALLOWED TO
1$:     TST     (R5)+           ; ONTO THE DATA POINTERS
        MOV     @(R5)+,R2       ; GET THE REQUESTED LUN
        MOV     @#$OTSV,R3      ; FORTRAN IMPURE AREA POINTER
        JSR     PC,$GETFILE     ; GET THE FORTRAN FDB AREA ADDRESS
        BPL     2$              ; SKIP IF NO FILE CURRENTLY OPEN.
        TRAP    128.+34.        ; CAN'T DO THIS IF ALREADY ONE OPEN
        BR      3$              ; TAKE SOME SORT OF ERROR EXIT
2$:     MOV     FILPTR(R3),R0   ; ADDRESS OF THE FTN FDB
        ADD     #D.FDB,R0       ; POINT TO THE REAL FDB
        MOV     R0,R1           ; COPY IT
        ADD     #F.FNB,R1       ; POINT TO THE FILE NAME BLOCK.
        MOV     (R5)+,R4        ; GET THE ADDRESS OF THE VER PARAMETER
        CMP     R4,#-1          ; WAS IT GIVEN?
        BEQ     3$              ; SKIP IF NOT THERE.
        MOV     @R4,N.FVER(R1)  ; RESTORE THE VERSION NUMBER
3$:     JSR     PC,.FIND        ; LOOK FOR ANOTHER ONE.
        BCS     4$              ; SKIP IF NO MORE
        MOV     #1,@(R5)        ; RETURN GOOD STATUS
        BR      5$              ; CONTINUE THE EXIT
4$:     MOV     #-10.,@(R5)     ; INDICATE EOF
5$:     CLR     FILPTR(R3)      ; NO MORE I/O IN PROGRESS.
        RTS     PC              ; BACK TO CALLER.
;

        .PAGE
;
; GFNAME:
;       THIS ROUTINE RETRIEVES THE FILENAME,EXTENSION AND VERSION
; ASSOCIATED WITH THE LUN SPECIFIED. A FILE MAY BE OPEN, OR THE LUN
; MAY BE USED FOR WILDCARD ACCESS.
;
; CALLING SEQUENCE:
;       CALL GFNAME(LUN,NAME)
;
;       LUN     = LOGICAL UNIT TO RETRIEVE THE NAME FROM.
;       NAME    = BYTE ARRAY TO RECEIVE THE NAME. NOTE:
;                 THE NAME IS RETURNED WITHOUT THE DEVICE/UIC CODES.
;                 AND THE NAME, EXTENSION AND VERSION FIELDS ARE
;                 VARIABLE IN LENGTH:
;
;       SUBSCRIPT               1         2
;                       123456789012345678901
;       NAME(MAX)       NNNNNNNNN.EEE.VVVVVV
;
        LUN = 2         ; PARAMETER OFFSET TO LUN
        NAME= 4         ; PARAMETER OFFSET TO RETURNED FILENAME.
;
GFNAME::
        CMPB    @R5,#2          ; CORRECT NUMBER OF PARAMETERS?
        BEQ     1$              ; SKIP IF SO
        TRAP    128.+80.        ; ISSUE RUN-TIME ERROR
;
1$:     MOV     @LUN(R5),R2     ; GET THE FILE LUN NUMBER
        MOV     @#$OTSV,R3      ; FORTRAN IMPURE AREA POINTER
        JSR     PC,$GETFILE     ; GET THE FTN FDB ADDRESS
        MOV     FILPTR(R3),R4   ; INTO R4
        ADD     #<D.FDB+F.FNB>,R4       ; POINT DIRECTLY TO FILE NAME BLOCK
;
        MOV     NAME(R5),R0     ; ADDRESS TO RETURN THE NAME TO
        MOV     N.FNAM  (R4),R1 ; CHARS 1-3
        JSR     PC,$C5TA        ; CONVERT RAD50 TO ASCII
        MOV     N.FNAM+2(R4),R1 ; CHARS 4-6
        JSR     PC,$C5TA
        MOV     N.FNAM+4(R4),R1 ; LAST THREE CHARACTERS
        JSR     PC,$C5TA
2$:     CMPB    #' ,-(R0)       ; LOOK BACKWARDS FOR LAST REAL CHARACTER
        BEQ     2$              ; (THERE WILL ALWAYS BE AT LEAST ONE.)
        INC     R0              ; POSITION POINTER TO NEXT AVAILABLE
        MOVB    #'.,(R0)+       ; PERIOD TO SEPARATE NAME AND EXTENSION.
        MOV     N.FTYP  (R4),R1 ; FILE TYPE (EXTENSION)
        JSR     PC,$C5TA        ; ALSO TO ASCII.
3$:     CMPB    #' ,-(R0)       ; LOOK BACKWARDS FOR LAST REAL CHARACTER
        BEQ     3$              ; (PERIOD, AT LEAST IS THERE.)
        INC     R0              ; ONTO NEXT USABEL SLOT.
        MOVB    #';,(R0)+       ; SEPARATE EXTENSION AND VERSION
        MOV     N.FVER  (R4),R1 ; VERSION IS AN OCTAL ASCII
        CLR     R2              ; REPRESENTATION, WITHOUT LEADING ZEROS
        JSR     PC,$CBOMG
        CLRB    @R0             ; ZERO AFTER THE END OF THE NAME.
        CLR     FILPTR(R3)      ; FILE SYSTEM NOT IN USE.
        RTS     PC              ; BACK TO USER.
;
        .END
```

The Journey from RSX to VMS: Part I, Assemble the Goods

Since I've appealed for user experiences in the RSX to VMS migration area, I thought I should contribute my laboratory's ongoing experience. This series of articles will chronicle our journey from an RSX-11M site to a VAX/VMS site. This first article will discuss our predelivery decisions such as configuration. I should also note that this is a story unfolding in real-time, I'm writing it as we go. Delivery is expected in about two months, although the staff has just gained access to another VAX to better plan our conversion effort.

The Advanced Graphics Laboratory (AGL) of The University of Texas at Austin Computation Center provides high performance graphics and data acquisition services using a PDP 11/34A host running RSX-11M. We operate AGL in a sign-up reservation fashion to minimize user group conflicts. The users are not unhappy with the current system, they just need more. The two serious flaws are program address space and raw throughput power. In four years I've been unable to think of a way to make 1 bit good for 3 or more choices. Failing at that, the obvious solution to both problems is to replace the 11/34 with a VAX. The tremendously increased addressing space per user program will alleviate the coding and thrashing problems of our graphics users. And we feel that the increased power of the VAX, properly configured, will allow us to provide timesharing style graphics and program preparation in addition to the high performance work. This should lower the individual cost and widen the community base. So then the questions become "which VAX" and "how much". Breifly, I'll list our current equipment:

Vector General 3405 3-D line drawing system (UNIBUS DMA)

Grinnell 270 image processing and true color display (DR11-B DMA)

Versatec 1200A printer/plotter (UNIBUS DMA)

Matrix 4007 color photographic hardcopy (on Grinnell)

Summagraphics Bit Pad data digiting tablet (DZ11 line)

DUP-11 local UTRJE protocol connecting Cybers

K-series A/D, D/A, digital IO modules (UNIBUS)

Plus disks (RM02s, RK05s), tapes (TE16s), cache, floating point, 124k words, DZ11-A, etc.

All the services currently provided must be provided on the VAX.

The first thing done was to contact each vendor to determine if the existing equipment would move easily onto a VAX (I specifically did not want to be the first!) and what software changes were needed. The results were immediately encouraging, we will be able to preserve the vast majority of user level software. The user levels subroutine calls are the same for the Vector General (VG), Grinnell-Matrix, and Versatec. New UNIBUS interfaces are needed for the VG and Versatec. This left open how to provide data tablet services, data acquistion, and Cyber communications:

- Bit Pad data tablet can still go on a VAX DZ line on the VAX, but this had proved to be a mistake on the PDP11. Demand input is very difficult to do on a multiunit line through a terminal driver and the Bit Pad will not recognize software control sequences to stop (e.g. CTRL-Q or X-OFF). It will only recognize the data-terminal-ready signal. A more sensible plan then is to put the Bit Pad on its own asynchronous line with a special driver. Guess what? DL11-E's aren't listed as VAX products, only DZ's are listed. Hmm, a big pause for thought. Why aren't DL's listed -- just no software support or is there something funny with the UNIBUS? Apparently it's the former, which causes us no problem since I need a simple little driver for it anyway. So we order a DL11-E for our Bit Pad, Corporate red flags it, we assure DEC we know there is no software, and everyone is happy.

- Cyber communications is essential for a happy user community here. Even though our current interface is a clunky DUP-11 (interrupt per character), the microcode already exists for the UTRJE protocol in a KMC microprocessor with DMC line unit. This will make our networking DMA and greatly offload the host. Another miracle occurred a few weeks ago when UT's Fusion Research Lab decided to get on our network. They agreed to provide the VMS code in exchange for the KMC code. One fewer problem for us as we'll only need to tailor their package. We also already own the KMC/DMC unit.

- K-series data acquistion is the big problem. Some of our users are using the DEC provided software but many are just overlaying the IO page with a COMMON block and directly banging away. On the VAX, the K-series modules are supported only via the LPA which will not support the streaming of mixed A/D and D/A's (e.g. perform 2 D/A's followed by an A/D and repeat process 100,000 times quickly). Uh oh! Now what? Well, I could put the K-series modules onto the VAX UNIBUS and write some drivers. That will mean a lot of interrupts on the VAX and the majority of the users really just want to digitize to/from tape. Since we have KMC experience, we can code up a KMC to offload as many interrupts as possible from the VAX. For example, provide digitizing directly to tape; or monitor a locked page in memory that corresponds to the input/output wanted for each module.

  This sounds reasonable so now we need another KMC. Guess what? It is not marketed for VAXes, so Corporate red flags it. Again, we tell DEC we know there is no software support. The debugger/loader should be easy to code for VMS (an RSX-11M version exists).

We're making progress, all of our secial equipment is acounted for. What about the VAX itself? Should it be a 750 or 780? How much memory? Disks? Tapes? The process was arduous but the 11/780 is the choice. Three reasons guided this decision: the UNIBUS load will be large on the VAX, two UNIBUSes will help; memory is limited to 2MB on a 11/750 which is not adequate for our color image processors; and finally, the general expandability of the 780 is desirable. The configuration is, therefore, a 4 megabyte 11/780 with 2 UNIBUSes. 24 DZ lines will allow 16 remote lines and 8 local lines (some local lines are used to support devices, not terminals). 20 users is, hopefully, a good conservative number insuring good response time in a graphics environment. The disk and tape choices are straight forward. We require compatibility with existing systems and the lowest reasonable hardware contract. Our configuration includes a TU77 tape, REM80 Winchester (124MB) drive with mass bus adapter (MBA) for the operating system disk, REP07 Winchester (512MB) with MBA for user disk, and a RM03 (67MB) removable pack drive. And, because our graphics users do a lot of floating point we include a hardware floating point unit. Both FORTRAN and PASCAL are listed for languages.

Whew! What's left? Cabinets, drawers, and backplanes must all be configured. I layout which items will go on which UNIBUS, count up the slots used, and configure the number of backplanes, etc. (each UNIBUS goes to a separate drawer). UNIBUS 1 will hold the DZ's, KMC/DMC communications, Versatec DMA controller, the line printer controller, and Grinnell DR11-B interface. UNIBUS 2 will hold all K-series modules (7), Bit Pad's DL-11E, KMC for data acquisition, and the Vector General. I just realized I had not mentioned the line printer before. Our Versatec plotter is currently doing double duty -- both printing and plotting. The load is tremendous so the printing will go onto a 300 lpm printer for the VAX.

Everything is going smoothly. Field service and software support and sales all review and approve the configuration. And then out of nowhere arises an operating system question -- a user has access to some nice UNIX programs he wants to run. Now discussing operating systems is worse than discussing either editors, religion, or politics. <u>It's a no win situation</u>. Time to do some accounting with a pencil. If we run UNIX instead of VMS, what will the conversion time be? None of our special products vendors market UNIX versions, so basically we'd be doing a full software conversion, at least several "man years". It's out of the question, we've got a maximum of 90 days. Is there a middle ground? I begin to track down rumors of "UNIX under VMS" products. The verdict is not in yet but it appears that something like Stanford Research Institute's Eunice package may provide the best of both worlds. Eunice converts UNIX system calls to/from VMS. Berkeley UNIX can be brought up under VMS. Sounds great -- perhaps too good to be true? How much difference is there between native UNIX and this package? What is the impact on the system? What other similar packages are there? I know Interactive Systems also markets one. This issue will be resolved in the future (I'd like to hear from knowledgeable sources on my worries) but at least the configuration is jelled to include VMS and some UNIX environment package.

Next month: the glories of being a new VMS user.

Standalone BRU Re-visited

C. T. Mickelson
Goodyear Aerospace Corp
Akron, Ohio 44315

At our site, we perform a scheduled full save once a week. Prior to Autopatch D, this full save was performed using standalone DSC. One irritant of using DSC was the need to manually re-boot the CPU to return to normal operations.

When we started using BRU about 9 months ago we wanted to eliminate this manual step to allow novice user/operators to perform the save following a simple recipe from the system console terminal.

Additionally, we wanted BRU on alternate media to allow a system recovery from past back-up tapes, without an existing system disc, in the event of a catastrophic system crash.

Following are a set of files that we use to build a standalone BRU that meets these requirements. This version of BRU has been used weekly for the last 9 months or so, and with each new patch (and they occur too frequently!), the ability to automatically maintain our standalone version saves considerable time.

BRUWAI.MAC is a program that searches the active task list every 5 seconds and jumps to the console emulator when it finds BRU is no longer active. BRUWAI.CMD and BRUWAI.BLD are support files that are used to build BRUWAI.TSK.

BRUVMR.CMD is a command file, ala SYSVMR, that configures a bootable file called BRU.SYS. BRUSTAND.CMD is a driver file to build BRU.SYS with a full RSX11M exec, drivers, BRU, and the BRU monitor, BRUWAI.

```
[200,201]BRUWAI.MAC

        .TITLE  BRUWAI
        .IDENT  /V1.0/
;
; PROGRAM TO AUTOMATICALLY REBOOT SYSTEM AFTER BRU
; EXITS FROM WEEKLY SAVE OPERATION
;
        .MCALL  TCBDF$,MRKT$C,WTSE$C EXIT$S
        TCBDF$
MKTEFN=1                                ;MARK TIME EVENT FLAG
MKTTMG=5                                ;WAIT FOR NUMBER OF SECONDS
MKTTMU=2
SYBOOT=165020                           ;CONSOLE EMULATOR BOOT ADDRESS
        .ENABL  LSB
START:
        CALL    $SWSTK,100$             ;SWITCH TO SYSTEM STATE
        MOV     RO,-(SP)                ;SAVE RO
        MOV     $ACTHD RO               ;GET 1ST TCB IN ACTIVE TASK LIST
        BEQ     30$                     ;IF NO ACTIVE TASKS, REBOOT


10$:
        CMP     BRUNAM,T.NAM(RO) ;CHECK 1ST WORD OF TASK NAME
        BNE     20$                     ;BRANCH IF NOT A MATCH
        CMP     BRUNAM+2,T.NAM+2(RO)  CHECK 2ND WORD OF TASK NAME
        BEQ     30$                     ;BRANCH IF MATCH ON ...BRU
20$:
        MOV     T.ACTL(RO),RO           ;LINK TO NEXT ACTIVE TASK TCB
        BNE     10$                     ;LOOP IF MORE ACTIVE TASKS
30$:
        MOV     RO,BRUFLG               ;SAVE SEARCH RESULT
        MOV     (SP)+,RO                ;RESTORE RO
        RETURN                          ;RETURN TO USER STATE AFTER SEARCH
100$:
        TST     BRUFLG                  ;IF ZERO, ...BRU IS NOT ACTIVE
        BEQ     110$                    ;REBOOT SYSTEM
        MRKT$C  MKTEFN,MKTTMG,MKTTMU  ELSE WAIT FOR SOME TIME
        WTSE$C  MKTEFN
        BR      START                   ;SEARCH LIST AGAIN
110$:
        CALL    $SWSTK,120$             ;SWITCH TO SYSTEM STATE
        RESET                           ;RESET UNIBUS AND
        JMP     @#SYBOOT                ;JUMP TO BOOT ROUTINE
        RETURN                          ;CANT GET HERE IF REBOOTING
120$:
        EXIT$S
BRUFLG: .WORD   0
BRUNAM: .RAD50  /...BRU/
        .DSABL  LSB
        .END    START
```

```
[200,201]BRUWAI.CMD

MAC BRUWAI=[1,1]EXEMC/ML,[11 10]RSXMC,[200,201]BRUWAI
TKB @BRUWAI.BLD
PIP BRUWAI.*/PU
[200,201]BRUWAI.BLD

BRUWAI/PR,BRUWAI/-SP=BRUWAI,[1,54]RSX11M.STB/SS
/
UNITS=0
STACK=64
TASK=BRUWAI
PRI=30
PAR=BRUWAI 0:600
//
[1,2]BRUVMR.CMD

BRU                               ! SELECT BRU.SYS FOR VMR OPERATIONS
SET /POOL=1200                    ! SET POOL TO BOTTOM OF KAPR5

! THE PROGRAM BRUWAI IS A BRU MONITOR. IT WAITS FOR BRU TO EXIT, AND THEN
! JUMPS TO THE CONSOLE EMULATOR TO BRING UP THE SYSTEM VIA OPERATOR KEYIN
! THE PROGRAM BRUWAI IS BUILT TO RUN AT 120000 VIRTUAL.  IF LOADED NOW
! IT WILL OCCUPY THE SAME PHYSICAL ADDRESSES AS ITS VIRTUAL ADDRESSES.
! THUS THE PROGRAM WILL RUN IN EITHER MAPPED OR UNMAPPED MODES.  THIS IS
! IMPORTANT SO THAT THE JMP #@SYBOOT INSTRUCTION WILL GET TO THE BOOTSTRAP,
! AFTER THE RESET INSTRUCTION CAUSES THE MEMORY MANAGEMENT HARDWARE TO GO TO
! UNMAPPED MODE.
SET /MAIN=BRUWAI.*:6:TASK        ! CREATE PARTITION FOR BRU WAIT TASK
INS [200,201]BRUWAI/CKP=NO       ! INSTALL BRU WAIT TASK
FIX BRUWAI                       ! LOAD BRUWAI INTO BOOTABLE IMAGE
SET /MAIN=TTPAR:*:400:TASK       ! SET UP TT: DRIVER PARTITION
LOA TT:                          ! LOAD TT: DRIVER
SET /MAIN=DRVPAR:*:*:SYS         ! SET UP PARTITION FOR OTHER DRIVERS
! LOAD DRIVERS THAT ARE NEEDED FOR ANY SYSTEMS ON WHICH THIS BRU WILL BE USED
LOA DR:/PAR=DRVPAR               ! LOAD DISC AND TAPE DRIVERS
LOA DM:/PAR=DRVPAR
LOA DK:/PAR=DRVPAR
LOA DL:/PAR=DRVPAR
LOA DY:/PAR=DRVPAR
LOA MM:/PAR=DRVPAR
SET /TOP=DRVPAR:-*               ! SHRINK DRVPAR
! UP TO HERE, EXCEPT FOR BRUWAI STUFF, WE ARE CREATING A NORMAL RSX11-M SYSTEM.
! NOW THE FOLLOWING WORKS FOR ANY NON-OVERLAYED TASK
SET /MAIN=BRU:*:1777:TASK        ! CREATE PARTITION FOR BRU
INS BRU/PAR=BRU/CKP=NO           ! INSTALL BRU WITHOUT CHECKPOINTING
FIX ...BRU                       ! LOAD BRU INTO BOOTABLE IMAGE
RUN ...BRU                       ! START BRU TO COME UP RUNNING
RUN BRUWAI                       ! START BRUWAI TO COME UP RUNNING
! SET TERMINALS TO SLAVE TO PREVENT USERS FROM CAUSING TROUBLE
SET /SLAVE=TT0:                  ! SET ALL TERMINALS TO SLAVE
SET /SLAVE=TT1:
SET /SLAVE=TT2:
SET /SLAVE=TT3:
SET /SLAVE=TT4:
SET /SLAVE=TT5:
SET /SLAVE=TT6:
PAR
```

```
[1,2]BRUSTAND.CMD

! FROM THE NOVEMBER 1980 MULTI-TASKER, PG 219
SET /UIC=[1,54]
PIP BRU.SYS=RSX11M.TSK/BL:258.          ! CREATE 64KW SYSTEM
PIP BRU.STB=RSX11M.STB                  ! WITH SYMBOL TABLE
 SETT VMR
.IFINS VMR .GOTO 10
.SETF VMR
INS $VMR
.10:
VMR @[1,2]BRUVMR
.IFF VMR REM VMR
! NOW YOU HAVE A NUMBER OF OPTIONS:
!
!       1. BOO [1,54]BRU        ! BOOT STANDALONE BRU.
!          XDT>G                ! START EXEC IF XDT BUILT IN
!          BRU>                 ! WHEN DONE, CTRL Z TO BRU AND
!                               ! BRUWAI WILL DETECT BRU INACTIVE
!                               ! AND JUMP TO BOOTSTRAP WITHIN 5 SEC.
!
!       2  VMR BRU              ! MAKE COPIES OF BRU ON OTHER MEDIA
!          VMR>SAV DY:BRU       ! MAKE A STANDALONE BRU ON FLOPPY
!          VMR>SAV MM:BRU       ! MAKE A STANDALONE BRU ON MAGTAPE
!          VMR>^Z               ! EXIT VMR
!
!       3. HARDWARE BOOT THE FLOPPY OR MAGTAPE TO RESTORE BRU TAPES TO
!          A SYSTEM DISC AND RECOVER FROM A CATASTROPHE, ASSUMING YOU
!          HAVE A RECENT SET OF BACKUP TAPES AVAILABLE.
```

DECUS

DIGITAL EQUIPMENT COMPUTER USERS SOCIETY
ONE IRON WAY, MR2-3/E55
MARLBORO, MASSACHUSETTS 01752

BULK RATE
U.S. POSTAGE
PAID
PERMIT NO. 129
NORTHBORO, MA
01532

INSTALLATION [redacted]

MOVING OR REPLACING A DELEGATE?

Please notify us immediately to guarantee continuing receipt of DECUS literature. Allow up to six weeks for change to take effect.

( ) Change of Address
( ) Delegate Replacement

DECUS Membership No.: ____________

Name: ____________

Company: ____________

Address: ____________

____

State/Country: ____________

Zip/Postal Code: ____________

Mail to: DECUS - ATT: Membership
One Iron Way, MR2-3
Marlboro, Massachusetts 01752 USA

Affix mailing label here. If label is not available, print old address here. Include name of installation, company, university, etc.