RSX

MULTI-TASKER

APRIL 1985 ISSUE

RSX Multitasker

Table of Contents

From the Editor

This month, the Multitasker a new columnist: Gary Maxwell. Gary joins our other columnist Ed Cetron and Bruce Mitchell. Gary's column is The RSX System Manager. The column will address issues relating to the management of an RSX system. For this column to work, reader input is needed as it is with our other columns. Please send your questions, comments, or ideas to either the Multitasker or directly to the columnists. Let's make these columns work.

Disk recovery, a subject best to be prepared for before you need it. Jerry Vaughn tells us how to prepare for recovery of damaged floppy disks - well worth remembering.

Terry Medlin looks into the two flavors of the FORTRAN run time system available to us on RSX systems. He explains how to make FORTRAN process multiple versions of a file.

On page three, the RSX development team at Digital responds to the Fall 1984 menu items, certainly MUST reading. The Menu is the official vehicle for us to communicate with DEC concerning RSX changes. More on this later.

RSX Mascot candidates are once again in the pages of the Multitasker. Bob Freeborn presents three more mascot possibilities on pages twenty-two to twenty-four. What do you think?

Finally, the RSX SIG will be soliciting menu items on a monthly basis. The RSX menu is the means to communicate our requests to Digital. If you want to see something new or something changed in RSX, fill out the form and send it in.

SIG CHAIR REPORT - SIG STATUS

I have just received the latest of my monthly reports which covers the month of December. The news is ALL good and very positive. DECUS revenue is currently $3,031,797. which is 15% over projections while expenses are currently $1,453,504 which is 47% under projections. While the expense totaL does NOT include several large bills from the symposium, it is obvious that DECUS is (as always) in GREAT financial shape.

Our SIG has is doing VERY well. We now have 10,456 members which is second only to the VAX SIG. Our subscriptions to the Multi-Tasker total 2,355 which is again the second largest and this represents $43,310 of the newsletter revenue.

Our seminars in Anaheim disseminated information to 101 people:

Industrial Automation	22
RSX SIG tapes	28
RSX Internals	51

From a health standpoint, I would suggest to you that the RSX SIG is continuing the traditions of old in being top-notch in all areas. Once you see the sessions planned for New Orleans (a total of about 60!), I think you will be VERY pleasantly surprised.

The status and health of a SIG is DIRECTLY attributable to the contributions of the members of the YOUR Steering Committee which includes several people! It is also related to the contributions of those of you who chose to be ACTIVE in the SIG. Thanks to all of you who have helped.

Terry Medlin

A Problem with the RSX SIG Tape Index Microfiche

The RSX SIG tape index microfiche, sold last fall at Anahiem, had an error on the second sheet of the index sorted by filename. This sheet can be exchanged for a corrected sheet. Send ONLY the second sheet of fiche from the "sorted by filename" list and a return address to:

Judy Arsenault
DECUS BP02
249 Northboro Rd
Marlboro, MA 01752

A corrected fiche will be sent to you. You must send the sheet with the error to get a corrected sheet. This sheet can be exchanged at the DECUS store in New Orleans during the Spring '85 Symposium.

Thanks for your cooperation

Bob Freeborn

Digital's Response To Menu

Fall 1984 Symposium
Anaheim, California

The following fifteen menu items were responded to by DIGITAL RSX Development Engineering in Anaheim during the Fall symposium:

1. Save worst case pool stats when changing RMD pages.

   Answer: We think this is a fairly easy task. If so, it will be done in future update.

2. Provide Virtual Disk Support.

   Answer: Support for this will be in V3.0.

3. ACS needs to be enhanced to list task checkpointed to a file.

   Answer: This may be addressed in the next release after V3.0

4. Document tuning on tasks?

   Answer: We will investigating the work required for a subsequent release.

5. Command line editor

   Answer: we believe that we need a command line editor. It will receive serious consideration for the release after V3.0

6. Settable upper and lower limits for an RMD page.

   Answer: BORING! If someone has the code for this "feature" forward it to RSX Engineering and we will attempt to integrate it in a future release.

7. Sysgen including NL: and CO:

   Answer: Seems like a reasonable request. We will put this on the list of items to consider for the next release after V3.0.

8. Do a double check with the user before BAD or INI switch.

   Answer: We are planning this for V3.0, but it may be delayed until the release after V3.0.

9. Allow system name to be changed for startup by VMR

   Answer: Good idea - it will be on the list of items for the release after V3.0.

10. Fortran 77 incompatibilities with ANSI standard.

    Answer: This has been brought to the attention of the 16 bit technical languages group and a response will be forthcoming.

11. Transparent DecNet spooling

    Answer: There will be a feature that will provide identical functionality at some point in time, namely a LAT Print Server. It will not implemented as DECNET spooling therefore.

12. Allow alteration of FLAG Page Printout.

    Answer: The sources are available now on the RSX-11M-PLUS Kit. This will not be a utility or switch.

13. Add a trailing FF option to the queue manager.

    Answer: Good idea - we will consider it for the next release after V3.0

14. Put DTE on the RSX-11M-PLUS kit.

    Answer: We will be doing that as soon as possible, probably in Update B or C to V3.0.

15. Ascii Queue Manager names

    Answer: This will not be done in the current implementation of the Queue Manager. We are considering an overhaul of the queue manager.

The RSX System Manager

This is the first article of a new, regular column in the Multi-Tasker, and as the name indicates, the column will address system management issues for RSX systems.

The impetus for this column originated at the Fall Symposium in Anaheim, during the System Managers Activities panel. A large majority of attendees at that panel session wanted to see more system management articles in the Multi-Tasker. I decided to use my RSX experience and begin a column that many readers might find useful.

Since I do not consider myself as being a definitive expert on managing computer systems, the format of this column will be driven by you, the readership. I will entertain any and all questions or ideas for articles relating to RSX system management issues. When I cannot come up with the best answer to a question, I will broadcast the question in this column, and ask other readers to send in their solutions. In the absence of reader-submitted ideas or questions to this column, I will publish a "System Manager's Short Note," a short article intended to make RSX system management an easier task.

This month, I would like to outline what "system management" means, especially with regards to RSX systems. Obviously, a manager's primary job is to maintain one or more RSX systems, with all the responsibilities and tasks that are required to keep RSX systems healthy. However, there are other functions attached to this position. Some of these functions are: acting as a applications consultant to users to help plan new applications or to improve existing ones; recommending the purchase or actually purchasing hardware and software to improve system capacity; and certainly not least in importance, the day-to-day "event driven" tasks that every manager must handle.

All of us who work with RSX systems are either users or programmers with some system management concerns or actual system responsible for the day-to-day upkeep of the system. It is my belief that all

of us are system managers to some degree, and that it is important to understand the major management tasks. With regard to RSX, these are:

1. System Generation (SYSGEN) - Copying, patching, building the operating system.

2. System Setup - Creating accounts and directories, using VMR and STARTUP.CMD to create the system and applications environment.

3. Mass Storage Maintenence - Backup and restoration of volumes, validation of volume integrity, enforcing disk space quotas.

4. Hardware and Software Maintenence - Examination of error logging reports, inspection of crash dumps, updating layered products, "keeping an eye out" for reports and rumors of hardware and software bugs.

5. Applications Maintenance - Keeping locally written software updated, managing system-wide libraries and files, installing and maintaining software development tools.

6. Event-driven Operations - Miscellaneous operations performed daily for the upkeep of the system, and responding to various user requests, such as "I can't read this tape," "I can't open this file," "My terminal's dead," and "Why can't I do so-and-so with EDT?"

Based on the various tasks listed above, the number of topics relevant to RSX system management is extremely large, too large to try and list here. The intent of this column is to present any issues that are relevant and current. If you have topics to be addressed in this column, whether it is a problem with BRU, a question about the queue manager, or a situation that hinders the efficient operation of your RSX system, please submit them either to the address listed below or to the Editor of the Multi-Tasker.

* * * * NEXT MONTH * * * *

So, you've deleted a file, and you absolutely have to get it back. Next month, we will see how files can be "undeleted," a tricky but occasionally necessary operation.

Sessions for System Managers
Spring 1985 DECUS Symposium, New Orleans

The following is a brief description of sessions relating to system management activities for RSX that have been scheduled for the Spring 1985 DECUS Symposium in New Orleans. Although this list is

accurate as of March 1, 1985, be sure to check your Symposium registration materials for any scheduling changes.

RSX SYSTEM TUNING AND PERFORMANCE OPTIMIZATION PANEL
Monday, May 27, 1985
12:00 noon to 1:30 p.m.

A panel of experienced RSX-11M and RSX-11M-Plus users will present practical information relating to measuring and improving RSX system performance. Time will be provided for performance-related questions from the audience.

RSX SYSTEM MANAGER'S ACTIVITIES PANEL
Monday, May 27, 1985
3:00 p.m. to 4:00 p.m.

This is a repeat of a session presented at the Fall 1984 Symposium in Anaheim. A panel of RSX-11M and RSX-11M-Plus system managers will make short presentations about the various activities performed by an RSX system manager. The emphasis will be on making system management more efficient. Time will be provided for management-related questions from the audience (but no BRU questions...read on).

RSX BRU PANEL AND FORUM
Monday, May 27, 1985
4:00 p.m. to 5:00 p.m.

A panel of users will give short presentations on how users and system managers can use BRU efficiently. Topics that may be covered include incremental backups, BRU behaviour with specific switches, available DECUS software that is useful with BRU tapes, etc. Time will be allocated for BRU-related questions from the audience.

RSX-11M V4.2 AND RSX-11M-PLUS V3.0 FIELD TEST PANEL
Monday, May 27, 1985
6:30 p.m. to 7:30 p.m.

A panel consisting of DEC developers and Field Test site users will report on experiences with the new versions of the operating systems. Topics presented may include: experiences with new features and new hardware support, software quality issues, and any problems experienced and workarounds developed during the Field Test. The panel will entertain questions relating to the new release from the audience.

LOADABLE DEVICE DRIVER BASES IN RSX11M SYSGEN

Thursday, May 30, 1985
6:00 p.m. to 6:30 p.m.

In order to support multiple and differing configurations of peripherals on different computer systems, a method for modifying RSX11M Sysgen is discussed that allows device drivers to be built with loadable, rather than resident, device data bases. System tailoring is done at VMR time when the particular devices for each configuration are included in the Bootable System image. The method allows the addition of new devices without a new Sysgen.

ADDING FOREIGN DISK DEVICES TO RSX-11M-PLUS
Friday, May 31, 1985
9:00 a.m. to 10:00 a.m.

An user will present his experiences with adding foreign disk device support to RSX-11M-Plus. Topics will include: modifications to the Executive and SAVe to make such devices bootable, how to add foreign disk support to system utilities, and how to add support to SYSGEN for foreign disk devices

Remember, the Symposium begins on Memorial Day. Be sure to get your hotel reservation form in early!

Please send questions, suggestions and submissions for this column to the following address:

Gary Maxwell
U.S.G.S. M/S 977
345 Middlefield Road
Menlo Park, CA 94025

The Software Clinic

Conducted by Ed Cetron

QUESTION 5 - We have an 11/23+ processor, 256kw memory and a third party vendor disk controller driving two 5-1/4" Winchesters. The drives are formatted to emulate RK07's so that they can be driven by the controller. The system works fine on/for an 18-bit RSX-11M V4.1B sysgen but crashes on an 22-bit system (O-bus). The actual DMDRV is from the third party vendor since DEC doesn't support

RK07's on the Q-bus. We would appreciate any information from anyone who has made the appropriate hacks to enable RK07's on the Q-bus in a 22-bit environment.

-- Michael P. Morrow

ANSWER 5 - First off, if anyone else has done this please let me know. I am only printing questioners' names to enable all responses to flow through me. This will prevent each questioner from being deluged with identical responses and will also guarantee that the Multi-Tasker will get a copy of the answer.

I have not looked at the code of the DMDRV extensively. I have spent some little time looking at the code of DYDRV in the past. If you will recall, the RX02 is an 18-bit only device. It has the exact same problem with 22-bits as does the RK07. From a cursory examination of the DMDRV code (since it does work in the 18-bit mode) is to force it to use 18-bit mode even when the rest of the system is 22-bit.

This can easily be accomplished by making use of the same approach as was done with DYDRV:

1. create a resident common buffer (DYCOM -> DKCOM)
2. patch the driver to do DMA only to DKCOM
3. get/put information to the task through the common.

This should be straight forward (if you have the old DYDRV as well as the new, patched version.) If not, let me know and I will send you copies.

This same technique MIGHT work for other 18-bit only devices on a Q22-bus. Usually it is worth a try.

QUESTION 6 - I just recently deleted a critical logging file. How do I get it back???

-- Ed Cetron (so I make mistakes too....)

ANSWER 6 - The first thing to do is to run a program call LAZARUS. It is available from the RSX SIG tapes. It searches through all the free space on the disk and attempts to 'resurrect' complete files which have been deleted. It does this by searching for the headers of deleted files and scanning down the list of blocks assigned to the file. Restrictions:

1. NO FURTHER ACTIVITY IS ALLOWED ON THE DISK - if you create another file, this could wipe out the header of the newly deleted file and you are totally out of luck.

2. You must be in the same UFD as the deleted file.

3. You must have another disk to write out the deleted files, otherwise see 1.

4. It will recover ALL deleted files of the current UFD so plan on a lot of extraneous, properly deleted files also showing up.

QUESTION 7 - I tried LAZRUS but due to restriction 1., it didn't work. This file is worth about $3,000.00 in billing costs. It can't be backed up since it is being generated continually in real-time. HELP!!!!!!

ANSWER 7 - There is another way. I will warn ALL potential users that it is very dangerous and neither I, nor my employer, nor the RSX SIG, nor the Multi-Tasker is responsible if you destroy the rest of the disk.

The steps are:

1. Make an immediate copy of the disk - this will save all other files on the disk.

2. Create a new file using pip with an initial allocation equal to you free space on the disk:

```
>pip DDnn:/fr
          (get the free blocks here)
>pip free.spc/bl:xxxx.=nl:
          (xxxx is the free space)
```

3. You should now have 0. blocks free and a file called free.spc with an size of 0./xxxx. in you directory.

4. Dismount the disk and remount it with the /UNLOCK qualifieer. This enables the index file for write access. This is the point at which you can really destroy you disk - be careful from here on. I would consider copying the disk again.

   (In my case, the disk was my currently running system disk - and i had to keep it running through this whole procedure.....)

5. Using PIP, find out the file id and sequence number of free.spc:

```
>pip free.spc/fu
```

6. subtract 10. from the file id and use it to be the starting block number for DMP:

```
>DMP ti:/lc/as=DDnn:[0,0]indexf.sys/hf/bl:yyy:zzz
```

where yyy is (fileid-10) and zzz is (fileid+10.)

7. Search the header list until you find the header for free.spc. An example header list is shown on page 11-11 of the RSX-11M/M-Plus Utilities Manual. The line in the identification area after I.FVER should say:

```
I.FVER      FREE      .SPC;1
```

8. Record the block number within indexf.sys that is the header for free.spc. DMP to a hard-copy device that block in two forms:

```
>dmp cl:=DDnn:[0,0]indexf.sys/bl:lll:lll
>dmp cl:=DDnn:[0,0]indexf.sys/bl:lll:lll/hf
```

where lll is the block number of free.spc in octal.

9. The next step is to find the lines that says:

```
F.HIBK H:d L:mmmmmm = xxxx.
F.EFBK H:0 L:000001 = 1.
```

where xxxx. is the size of free.spc that we allocated previously.

10. F.EFBK sets the end-of-file block. In our case since we have a 0. block file, it is set to block 1. We now have to go in and change it to xxxx. First convert xxxx. to octal in two word format (H:d L:mmmmmm). This should be identical to the two word format of F.HIBK. Then run ZAP to actually go in and change the word:

```
>zap                              (you type)
zap>DDnn:[0,0]indexf.sys/ab       (you type)
    -lll:26/                      (you type)
    lll:000026/ 000000            (it types)
    -d                            (you type)
    -                             (you type <cr>)
    lll:000030/ 000001            (it types)
    -mmmmmm+1                     (you type)
    -x                            (you type)
    >
```

note: d and mmmmmm are octal and the same as in F.HIBK
- are prompts from ZAP
all of the (you type) lines are ended with <cr>

```
lll is the octal block of the free.spc header
The mmmmmm+1 is to set the end-of-file block
one past the last block.
```

11. The file is know as big as free space and is listed on a PIP/LI as xxxx./xxxx. in size. Or it would be except that H.CKSM is now an invalid checksum.

12. Run DMP again and it will generate the new checksum:

```
>DMP ti:=DDnn:[0,0]indexf.sys/hf/bl:lll:lll
```

The checksum can also be calculated (IN OCTAL) as the old H.CKSM + (xxxx. converted to octal) + 1.

13. Run ZAP again to modify the checksum:

```
>zap                              (you type)
zap>DDnn:[0,0]indexf.sys/ab       (you type)
-lll:776/                         (you type)
lll:000776/ cccccc                (it types)
-nnnnnn                           (you type)
-x                                (you type)
>

note: cccccc is the old octal checksum
      nnnnnn is the new octal checksum
      - are prompts from ZAP
      all of the (you type) lines are ended with <cr>
```

14. Run pip again to verify correct checksum:

```
>PIP free.spc/fu
```

15. The file is now "perfect". The only problem is to get your information out of it. The first step is to dump the entire file and determine which blocks are needed.

```
>DMP ti:/as/lc=free.spc
```

16. After figuring out which blocks that you need, you must reshuffle the map area pointers to rearrange the blocks in free.spc to match the file that you deleted. This process involves reading the map pointers and manipulating them. There format is:

```
byte 1 - (block count - 1)
byte 2 - high order part of logical block number
word 2 - low order part of logical block number
```

17. After determing how to shuffle the blocks, regenerate the pointer table. Then run ZAP to replace the old table with the new table (sorry for the simplification - the shuffling is much harder, but is also very file specific and difficult to explain in any more detail). Determine the last block that you need and and one to it to get the new end-of file block. Replace E.EFBK with this value. Again run DMP to get the new checksum and replace it with ZAP.

18. At this point do a full directory of free.spc. The size should be bbb./xxxx. (where bbb. is the size that you want). Truncate the file with pip and rename the file to whatever the name of the deleted file was. You have know succesfully undeleted your file.

I hope the above information is valuable. More can be found in Appendix F for the file header format, and Appendix E for the index file format. I was able to successfully undelete a 600. block file from 3500. blocks of free space. It took about 8. hours of sweating and groaning.

The Bag of Tricks: MACRO-11

Bruce R. Mitchell
Machine Intelligence and Industrial Magic
PO Box 601
Hudson, WI 54016

This column covers MACRO-11 bag-of-tricks routines, as stated in last month's issue of the Multi-Tasker. It will appear as space permits. All MACRO programmers are encouraged to submit their favorite routines to the Multi-Tasker so that these useful, interesting, or just plain bizarre tricks can be put out before the SIG in general for the admiration and edification of all.

In this month's column, we have something which I'm not so sure I want to give away - a routine to identify the host processor type.

"Well, I can get that by mapping the Exec or doing a GIN$!", says the smart guy in the back row. Yes indeed, VAXbreath, but you can't map the Exec unless you're a privileged task, and you can't do a GIN$ unless you're under M-Plus. This requires neither.

This routine lets those of us who do commercial coding ride herd on persons of questionable morality who would make unauthorized copies of our children who are out in the big cold world. If you license a program to run on an 11/34, then by damn, you can use this to make sure it runs on an 11/34 and nothing else.

It is also very good for amazing the peons.

This routine was taken from the M-Plus SAV source, with heavy modification to make it run without having to be in kernel mode. As such, it cannot be quite as specific to the host CPU as the original code was, which mapped the I/O page to look at several CPU-specific registers. However, it gets in fairly close and, in most cases, gets down the specific CPU. In any case, it will at least get to the CPU class (e.g. KDJ11 chip). As in last month's feature, those who feel that this is a severe drawback are invited to look at the original code.

It should be noted that this routine sits on top of the trap vectors; hence, any attempt to use it when the mother program is linked to a debugging aid such as ODT will produce very strange results.

Because the author tends to separate data and code structures in his programs, the labels have been made so that the data and code can be readily separated within a single source file.

The output from IDHOST is an integer in R0 which gives the host CPU type. This can be used directly in the mother program to compare against a stored value.

```
   .SBTTL  IDHOST  Identify Host


;       IIIIIIII  DDDDDDDD    HH      HH   OOOOOOOO     SSSSSSSS  TTTTTTTTTT
;       IIIIIIII  DDDDDDDD    HH      HH   OOOOOOOO    SSSSSSSSS  TTTTTTTTTT
;          II     DD     DD   HH      HH  OO      OO  SS              TT
;          II     DD     DD   HH      HH  OO      OO  SS              TT
;          II     DD      DD  HHHHHHHHHH  OO      OO   SSSSSSS        TT
;          II     DD      DD  HHHHHHHHHH  OO      OO    SSSSSSS       TT
;          II     DD      DD  HH      HH  OO      OO          SS      TT
;          II     DD      DD  HH      HH  OO      OO          SS      TT
;          II     DD     DD   HH      HH  OO      OO          SS      TT
;          II     DD     DD   HH      HH  OO      OO          SS      TT
;       IIIIIIII  DDDDDDDD    HH      HH   OOOOOOOO    SSSSSSSS       TT
;       IIIIIIII  DDDDDDDD    HH      HH   OOOOOOOO    SSSSSSS        TT


;       IDHOST - Identify Host System's Processor Type
;
;       This subroutine attempts to identify the type of processor on
;       which the host system is running via various instructions unique
;       to each PDP-11 processor and special Exec calls.
```

```
;
;       NOTE.  It is assumed that no PDP-11/45 uses 22 bit addressing, and
;              all PDP-11/70s use 22 bit addressing.
;
;       Inputs:  None
;
;       Outputs:  R0 - Processor type, one of:
;                      0    - Unknown
;                      2324 - LSI-11/23, 23-Plus, PDP-11/24
;                      34   - PDP-11/34
;                      3540 - PDP-11/35, PDP-11/40
;                      44   - PDP-11/44
;                      4555 - PDP-11/45, PDP-11/55
;                      60   - PDP-11/60
;                      7384 - LSI-11/73, LSI-11/83, PDP-11/84
;                      70   - PDP-11/70
;
;       Register dispositions:  R0, R1 destroyed
;
;       Variable dispositions:  None modified


;       GPRT$ data buffer

GPRBUF: .WORD   0, 0, 0


;       SST trap table

TRPTBL: .WORD   0, 0, 0, 0, RSRVED, 0, 0, 0
TRTLEN = . - TRPTBL


;       GEN partition name

GENPAR: .RAD50  ®GEN    ®


;       Trap catcher

RSRVED: INC     R1
        RTT


;       Subroutine code

IDHOST: MOV     R1, -(SP)               ; Save R1 on the stack
        CLR     R0                      ; Clear processor type register
        CLR     R1                      ; Clear the trap flag register
        SVTK$S  #TRPTBL, #TRTLEN        ; Set up trapping

;       We start hoping that it's an easy one, and do a MFPT
```

```
        MFPT                            ; Move from processor type
        TST     R1                      ; Did it trap?
        BNE     10$                     ; If so, on to next text

;       It didn't trap; it must be a processor which supports MFPT

        CMPB    R0, #1                  ; Is it a PDP-11/44?
        BNE     1$                      ; If not, go see next one

        MOV     #44., R0                ; It's an 11/44
        BR      100$                    ; Go clear trapping and return


;       Not a PDP-11/44; is it a 23 or 24?
;       hi byte 0 for 24

1$:     CMPB    R0, #3                  ; Is it a 11/23B or a 11/24?
        BNE     2$                      ; If not, go see next one

        MOV     #2324., R0              ; It's an 11/23 or 11/24
        BR      100$                    ; Go clear trapping and return


;       Not a PDP-11/44, 11/23 or 11/24; is it a J11 based CPU?

2$:     CMPB    R0, #5                  ; Is it J11 based?
        BNE     100$                    ; If not, we don't know what it is

        MOV     #7384., R0              ; It's an 11/73, 11/83 or 11/84
        BR      100$                    ; Go clear trapping and return


;       Check for PDP-11/60 using MED

10$:    CLR     R1                      ; Clear the trapping flag
        .WORD   076600                  ; First half of an 11/60 MED.  On 11/60s
                                        ;  a CPU internal register is read into
                                        ;  into R0.  If not a 60, trap catcher
                                        ;  gets it.  The second half of the MED
                                        ;  is then be executed as a:
        .WORD   000400                  ;  BR .+2 (effectively, a NOP)
        TST     R1                      ; Did it trap?
        BNE     20$                     ; If so, on to next text

;       It didn't trap; it must be an 11/60

        MOV     #60., R0                ; It's an 11/60
        BR      100$                    ; Go clear trapping and return


;       Check for PDP-11/34 using an MFPS

20$:    CLR     R1                      ; Clear the trapping flag
```

```
        MFPS    R0                      ; Try to move from processor status
        TST     R1                      ; Did it trap?
        BNE     30$                     ; If so, on to next text

;       It trapped; it must be an 11/34

        MOV     #34., R0                ; It's an 11/34
        BR      100$                    ; Go clear trapping and return


;       Check for PDP-11/35 or PDP-11/40 using an SPL

30$:    CLR     R1                      ; Clear the trapping flag
        SPL     0                       ; Try to set priority level
        TST     R1                      ; Did it trap?
        BEQ     40$                     ; If so, on to next text

;       It trapped; it must be an 11/35 or 11/40

        MOV     #3540., R0              ; It's an 11/35 or 40
        BR      100$                    ; Go clear trapping and return


;       45/55 and 70 architecture is identical.  Assume that all 70s run
;       in 22-bit addressing mode and do a GPRT$.

40$:    GPRT$S  #GENPAR, #GPRBUF        ; Get partition parameters on GEN
        BCS     100$                    ; If it failed, exit now

        MOV     #70., R0                ; Assume it's an 11/70 (or 74)
        ADD     GPRBUF, GPRBUF+2        ; Add base of GEN to size of GEN
        CMP     GPRBUF+2, #7700         ; 126 Kwords or greater?
        BGE     100$                    ; If not, it's a 45 "for sure" (ha)

        MOV     #4555., R0              ; It's an 11/45, 50 or 55

;       Clear the trap catcher and return to the caller

100$:   CLR     TRPTBL+8.               ; Clear reserved instruction vector
        SVTK$S  #TRPTBL, #TRTLEN        ; Clear trapping
        MOV     (SP)+, R1               ; Restore R1 from the stack

        RETURN                          ; Return to the caller


        .END
```

Quick and Dirty Disk Recovery

J. W. Vaughn
U.C.S.D. Medical Center
225 Dickinson St. H-772-C
San Diego, CA. 92103-9981
(619)-697-0042

During an audit by the sponser of a drug study, a user brought me a floppy and said "it doesn't work". I tried mounting the disk with the command MOU DY0:/OVR/VI and alas, HOME BLOCK IO ERROR. It was imperative to recover the data as the disk contained over 200 files from the study that would be nearly impossible to repeat.

I examined physical blocks 1 and 2 with DMP and found the boot block intact but the home block garbage. Since we have a set of disks formatted exactly the same, (with a little help from IND) I used another disk to find the physical block where the rest of INDEXF.SYS was located. Examination showed this apparently unscathed and now the question was, how to recover quickly and easily.

I suspected that MOU was the only roadblock and that once mounted, the file utilities would probably work. Following that reasoning, I formatted a disk exactly as the damaged disk with the same number of ufd's, same number of files, etc. I MOUnted that disk in DY0: and then replaced it with the damaged disk. PIP DY0:*.*/LI listed the directory of the damaged disk. I then MOUnted the duplicate disk in DY1: and FLXed the files the good disk. Only one file not recovered in the process and it was the last file written to the disk. I suspect that it caused the damage but have been unable to prove so.

I used FLX because it has proven to be somewhat more robust on our system than PIP. Later, when speed was no longer a criteria, I repeated the recovery using PIP with the same results.

This trick is safe because the damaged disk is never written to and may allow a quick recovery if only the home block is damaged. It should work as long as the two disks have INDEXF.SYS files that match both physically and logically with the exception of the damaged home block.

Interfacing with the FORTRAN Run Time System

Terry Medlin
Survey Sampling, Inc.

Many of you undoubtedly still develop software using FORTRAN. DEC sells two distinct versions of FORTRAN known as "FOR" and "F77". The purpose of this article is to furnish you with some macro code that can be used to embellish your software and hopefully make your life easier. Although the two versions of FORTRAN support run time systems that are similar, my examples will be geared toward F77 and they may not work under FOR.

F77 can be installed on your system to provide either FCS file support or RMS file support (or both if you place the OTS code in separate libraries). In my examples, you need to be aware of which OTS you are using since they are not even close to being the same: one uses a typical FCS File Desriptor Block (FDB) while he other uses the typical RMS RAB and FAB.

One of the typical applications in our shop is the need to process multiple versions of a file: usually starting from lower versions to higher and also in a non-destructive fashion. The following shell FORTRAN program demonstrates this usage:

```
            PROGRAM SHOWUM
C
C THIS EXAMPLE CALLS ROUTINE GETV WHICH IS FOR FCS FILES
C
            INTEGER*2 LOWVER
            INTEGER*2 HIVER
            INTEGER*2 I
   C
   C PROCESS ALL VERSIONS OF A FILE
   C
            OPEN(UNIT=1,NAME='TESTCASE.DAT;-1',...
            CALL GETV( 1 , LOWVER )
            CLOSE(UNIT=1)
   C
   C GET THE HIGH VERSION
   C
            OPEN(UNIT=1,NAME='TESTCASE.DAT;0,...
            CALL GETV( 1 , HIVER )
            CLOSE(UNIT=1)
   C
   C PROCESS THE FILE
   C
            DO 100 I = LOWVER , HIVER
```

```
            ETC.
```

For each open file, F77 keeps a block of information called a LUB and associated with each LUB is the FDB which the FCS routines use:

```
   .TITLE GETV - ROUTINE TO GET VERSION NUMBER OF OPEN FCS FILE
   .IDENT /X01/
;+
;  CALL GETV ( INTEGER*2  LUN , INTEGER*2 VER )
;
;  LUN      LOGICAL UNIT NUMBER WHERE FILE IS ALREADY OPEN
;  VER      RETURNED VERSION NUMBER
;-
GETV::
   MOV      @2(R5),R2        ;SET CHNL NO.
   CLR      @4(R5)           ;CLEAR VERSION IN CASE OF ERROR
   JSR      PC,$FCHNL        ;GET LUB IN R0
   BCS      ERROR
   MOV      R0,R1            ;LUB TO R1
   ADD      #12.,R1          ;MAKE R1 POINT TO FDB
   MOV      F.FNB+N.FVER(R1),@4(R5) ; RETURN VERSION NUMBER
ERROR:
   RETURN
   .END
```

If the above code looks mysterious, then review the F77 OTS guide and review the description of an FDB in the FCS manual. Note that the $FCHNL is a compiler routine that is specific to FORTRAN.

Now since some of you may link to the RMS style OTS, here is a routine that accomplishes the same thing:

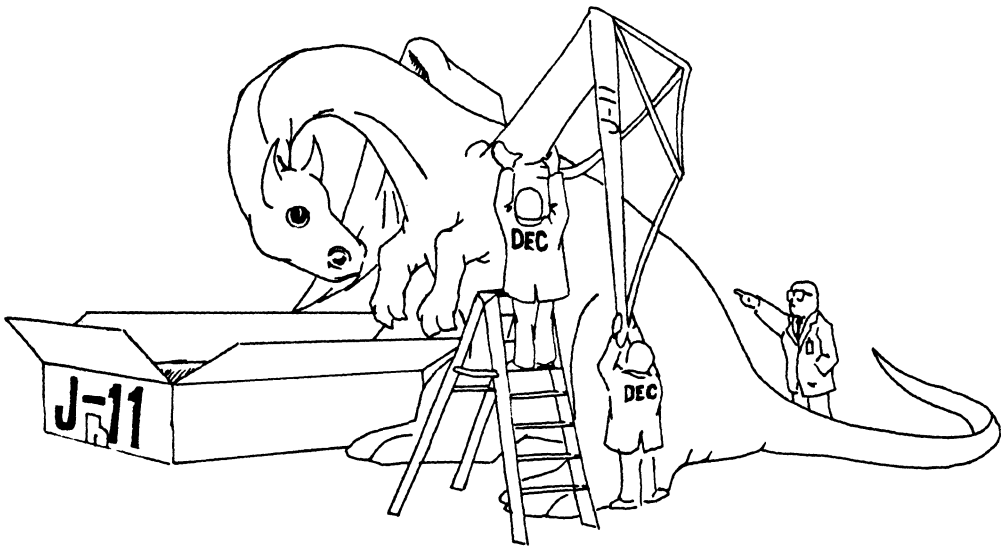
```
   .TITLE GETRV - ROUTINE TO GET VERSION NUMBER OF OPEN RMS FILE
   .IDENT /X01/
;+
;  CALL GETRV ( INTEGER*2  LUN , INTEGER*2 VER )
;
;  LUN      LOGICAL UNIT NUMBER WHERE FILE IS ALREADY OPEN
;  VER      RETURNED VERSION NUMBER
;-
GETRV::
   MOV      @2(R5),R2        ;SET CHNL NO.
   CLR      @4(R5)           ;CLEAR VERSION NUMBER IN CASE OF ERROR
   JSR      PC,$FCHNL        ;GET LUB IN R0
   BCS      ERROR
D.NAMC=5
   MOVB     D.NAMC(R0),R2    ;SIZE OF NAME STRING
D.NAM=32.+120
   ADD      #D.NAM,R0        ;START OF NAME
LOOP:
   CMPB     (R0)+,#';        ;FIND SEMICOLON IN STRING
```

```
    BEQ     GOTIT           ;BR IF HIT
    SOB     R2,LOOP         ;KEEP GOING OTHERWISE
    BR      EXIT            ;NEVER FOUND IT
GOTIT:
    CALL    $COTB           ;CONVERT TO BINARY - RETURN IN R1
    MOV     R1,@4(R5)       ;RETURN TO USER
ERROR:
EXIT:
    RETURN
    .END
```

In the above code, the D.xxx style symbols are defined as part of FORTRAN. The definitions of the symbols will NOT normally be in your library. They can be (and should be defined) by using the module in the FORTRAN installation. Also, the $COTB routine is a standard RSX routine that is described in the manual on system routines (NOT system services).

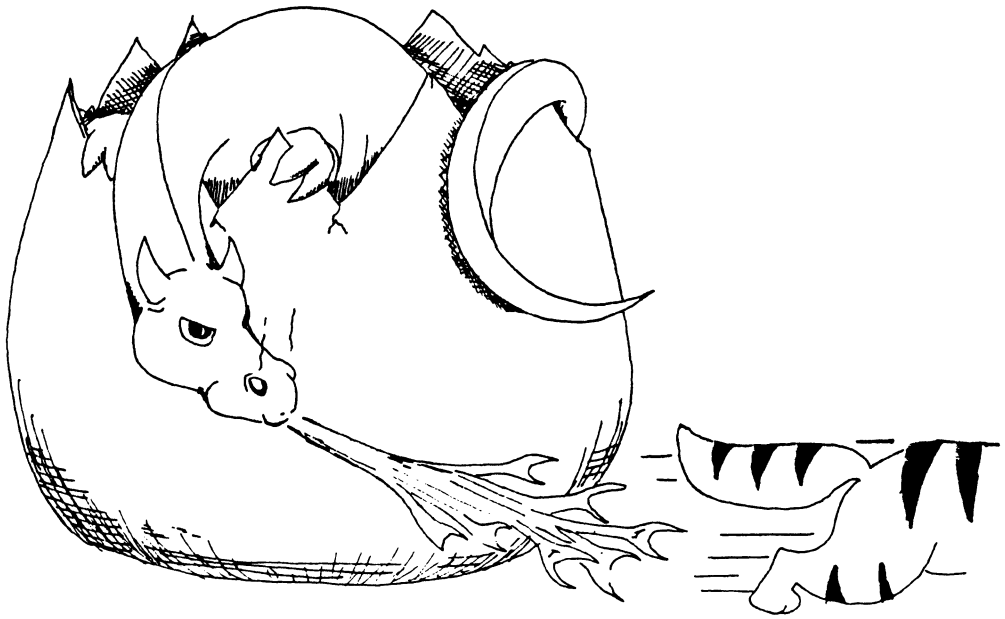
The RMS case is more difficult because you have to decode the ASCII file string. I cannot believe that the version is not stuck somewhere but I could not find it and you will find scant documentation on RMS file structures in the RSX documentation package. That is one reason the SIG is sponsoring an RMS pre-symposia seminar in New Orleans!

In future articles, we will address some other neat and useful things to pick up from the OTS.



Michael D'Angelo
© February 19, 1985


RSX
Michael D'Angelo
1/85



Michael D'Angelo
© February 19, 1985

Page 1 of ____

RSX Menu Item Submission Form

Name:_____ Firm:_____

Address:_____ Phone:_____

_____

_____

Operating System: RSX-11M RSX-11M+ Micro-RSX VAX-RSX POS-RSX

Describe the capability you would like to see available. Be as specific as possible. Please don't assume we know how it's done on the XYZ system. Explain how the capability would be useful and give an example of its use. If you wish, suggest a possible implementation of your request

Return forms to: Allen Bennett
L.S.I. Rapistan
555 Plymouth Ave., N.E.
Grand Rapids, MI 49505

Printed in the U.S.A.

DECUS SUBSCRIPTION SERVICE
DIGITAL EQUIPMENT COMPUTER SOCIETY
249 NORTHBORO ROAD, (BP02)
MARLBORO, MA 01752

Bulk Rate
U.S. Postage
PAID
Permit No. 18
Leominster, MA
01453

STATUS CHANGE

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

( ) Change of Address
( ) Please Delete My Membership Record
(I Do Not Wish To Remain A Member)

DECUS Membership No: _____

Name: _____

Company: _____

Address: _____

_____

_____

State/Country: _____

Zip/Postal Code: _____

Mail to: **DECUS - ATTN: Subscription Service**
249 Northboro Road, BP02
Marlboro, Massachusetts 01752 USA

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